aws

Panduan Pengguna

# Amazon ElastiCache untuk Redis

Versi API 2015-02-02

# Amazon ElastiCache untuk Redis: Panduan Pengguna

# Table of Contents

# Apa itu Amazon ElastiCache untuk Redis?

Selamat datang di Panduan Pengguna Amazon ElastiCache for Redis. Amazon ElastiCache adalah layanan web yang memudahkan pengaturan, pengelolaan, dan skala penyimpanan data dalam memori terdistribusi atau lingkungan cache di cloud. Layanan ini menyediakan solusi caching berkinerja tinggi, dapat diskalakan, dan hemat biaya. Layanan ini juga membantu menghilangkan kompleksitas yang terkait deployment dan pengelolaan lingkungan cache terdistribusi.

Anda dapat mengoperasikan Amazon ElastiCache dalam dua format. Anda dapat memulai dengan cache nirserver atau memilih untuk merancang klaster cache Anda sendiri.

> **Note**
>
> Amazon ElastiCache bekerja dengan mesin Redis dan Memcached. Gunakan panduan untuk mesin yang sesuai. Jika Anda tidak yakin mesin yang ingin digunakan, lihat Membandingkan cache yang dirancang sendiri dari Memcached dan Redis pada panduan ini.

## Caching nirserver

ElastiCache untuk Redis menawarkan caching tanpa server, yang menyederhanakan penambahan dan pengoperasian cache berbasis Redis untuk aplikasi Anda. ElastiCache untuk Redis Tanpa Server memungkinkan Anda membuat cache yang sangat tersedia dalam waktu kurang dari satu menit, dan menghilangkan kebutuhan untuk menyediakan instance atau mengkonfigurasi node atau cluster. Pengembang dapat membuat cache Tanpa Server dengan menentukan nama cache menggunakan ElastiCache konsol, SDK atau CLI.

ElastiCache untuk Redis Serverless juga menghilangkan kebutuhan untuk merencanakan dan mengelola kapasitas caching. ElastiCache untuk Redis terus memantau memori cache, komputasi, dan bandwidth jaringan yang digunakan oleh aplikasi Anda, dan skala untuk memenuhi kebutuhan aplikasi Anda. ElastiCache untuk Redis menawarkan pengalaman titik akhir sederhana bagi pengembang, dengan mengabstraksi infrastruktur cache dan desain cluster yang mendasarinya. ElastiCache untuk Redis mengelola penyediaan perangkat keras, pemantauan, penggantian node, dan penambalan perangkat lunak secara otomatis dan transparan, sehingga Anda dapat fokus pada pengembangan aplikasi, daripada mengoperasikan cache.

ElastiCache untuk Redis Serverless kompatibel dengan Redis 7.1 dan di atasnya.

# Merancang sendiri ElastiCache untuk cluster Redis

Jika Anda memerlukan kontrol halus atas cluster Redis Anda, Anda dapat memilih ElastiCache untuk mendesain cluster Redis Anda sendiri. ElastiCache ElastiCache memungkinkan Anda untuk mendesain cluster Anda, dengan memilih tipe node, jumlah node, dan penempatan node di seluruh AWS Availability Zones untuk cluster Anda. Karena ElastiCache merupakan layanan yang dikelola sepenuhnya, secara otomatis mengelola penyediaan perangkat keras, pemantauan, penggantian node, dan penambalan perangkat lunak untuk cluster Anda.

Merancang sendiri ElastiCache untuk cluster Redis menawarkan fleksibilitas dan kontrol yang lebih besar atas cluster Anda. Misalnya, Anda dapat memilih untuk mengoperasikan klaster dengan ketersediaan Single-AZ atau ketersediaan multi-AZ tergantung pada kebutuhan Anda. Anda juga dapat memilih untuk menjalankan Redis dalam mode klaster yang memungkinkan penskalaan horizontal, atau tanpa mode klaster hanya untuk penskalaan secara vertikal. Saat merancang klaster Anda sendiri, Anda bertanggung jawab untuk memilih jenis dan jumlah simpul dengan benar untuk memastikan bahwa cache Anda memiliki kapasitas yang cukup seperti yang dipersyaratkan oleh aplikasi Anda. Anda juga dapat memilih kapan harus menerapkan patch perangkat lunak baru ke klaster Redis Anda.

Saat merancang sendiri ElastiCache untuk cluster Redis, Anda dapat memilih untuk menjalankan Redis 3.0 ke atas.

# Layanan terkait

Amazon MemoryDB for Redis

Saat memutuskan untuk menggunakan ElastiCache for Redis atau Amazon MemoryDB for Redis, pertimbangkan perbandingan berikut:

- ElastiCache for Redis adalah layanan yang biasa digunakan untuk cache data dari basis data lain dan penyimpanan data menggunakan Redis. Anda harus mempertimbangkan ElastiCache for Redis untuk caching beban kerja di mana Anda ingin mempercepat akses data dengan basis data primer atau penyimpanan data yang ada (kinerja baca dan tulis mikrodetik). Anda juga harus mempertimbangkan ElastiCache for Redis untuk kasus penggunaan di mana Anda ingin menggunakan struktur data dan API Redis untuk mengakses data yang disimpan dalam basis data perimer atau penyimpanan data.
- Amazon MemoryDB for Redis adalah basis data dalam memori yang tahan lama untuk beban kerja yang memerlukan basis data primer yang sangat cepat. Anda harus mempertimbangkan

untuk menggunakan MemoryDB jika beban kerja Anda memerlukan basis data tahan lama yang memberikan kinerja sangat cepat (latensi baca mikrodetik dan penulisan satu digit milidetik). MemoryDB mungkin juga cocok untuk kasus penggunaan Anda jika Anda ingin membangun aplikasi menggunakan struktur data dan API Redis dengan basis data primer yang tahan lama. Terakhir, Anda harus mempertimbangkan untuk menggunakan MemoryDB untuk menyederhanakan arsitektur aplikasi Anda dan menurunkan biaya dengan mengganti penggunaan basis data dengan cache untuk ketahanan dan kinerja.

[Amazon RDS]

ElastiCache for Redis dapat membantu Anda menghemat biaya database dengan menyimpan data yang sering diakses dalam cache. Jika aplikasi Anda memiliki persyaratan throughput baca yang tinggi, Anda dapat mencapai skala tinggi, kinerja cepat, dan biaya penyimpanan data yang lebih rendah dengan menggunakan ElastiCache, alih-alih menskalakan basis data Anda.

# Cara kerjanya

Di sini Anda dapat menemukan ikhtisar komponen utama penerapan ElastiCache for Redis.

## Mesin cache dan caching

Cache adalah penyimpanan data dalam memori yang dapat Anda gunakan untuk menyimpan data yang di-cache. Biasanya, aplikasi Anda akan menyimpan data yang sering diakses dalam cache untuk mengoptimalkan waktu respons. ElastiCache untuk Redis menawarkan dua opsi penerapan: Cluster tanpa server dan yang dirancang sendiri. Lihat Memilih antara opsi deployment

> **Note**
>
> Amazon ElastiCache bekerja dengan mesin Redis dan Memcached. Gunakan panduan untuk mesin yang sesuai. Jika Anda tidak yakin mesin yang ingin digunakan, lihat Membandingkan cache yang dirancang sendiri dari Memcached dan Redis pada panduan ini.

Topik

- Bagaimana cara ElastiCache kerja Redis
- Dimensi harga
- ElastiCache untuk cadangan Redis

# Bagaimana cara ElastiCache kerja Redis

## ElastiCache untuk Redis Tanpa Server

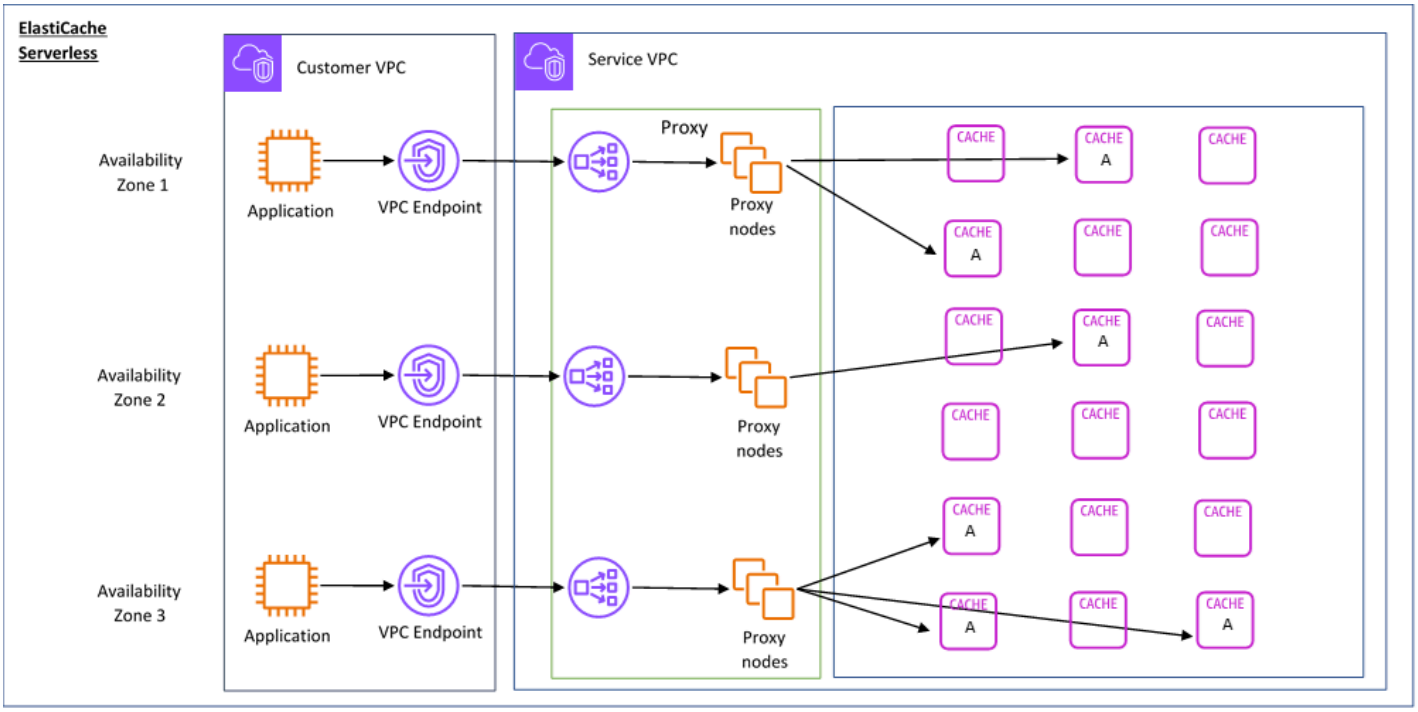
ElastiCache untuk Redis Tanpa Server memungkinkan Anda membuat cache tanpa khawatir tentang perencanaan kapasitas, manajemen perangkat keras, atau desain cluster. Anda cukup memberikan nama untuk cache Anda dan Anda menerima satu titik akhir yang dapat Anda konfigurasi di klien Redis Anda untuk mulai mengakses cache Anda.

> **Note**
>
> ElastiCache untuk Redis Serverless menjalankan Redis dalam mode cluster dan hanya kompatibel dengan klien Redis yang mendukung TLS dan protokol cluster Redis.

### Manfaat Utama

- Tidak ada perencanaan kapasitas: ElastiCache Tanpa server menghilangkan kebutuhan Anda untuk merencanakan kapasitas. ElastiCache Tanpa server terus memantau memori, komputasi, dan pemanfaatan bandwidth jaringan cache Anda dan menskalakan baik secara vertikal maupun horizontal. Hal ini memungkinkan simpul cache untuk tumbuh dalam ukuran, sekaligus secara paralel memulai operasi menskalakan ke luar untuk memastikan bahwa cache dapat diskalakan untuk memenuhi persyaratan aplikasi Anda setiap saat.
- Pay-per-use: Dengan ElastiCache Tanpa Server, Anda membayar untuk data yang disimpan dan menghitung yang digunakan oleh beban kerja Anda pada cache. Lihat Dimensi harga.
- Ketersediaan tinggi: ElastiCache Tanpa server secara otomatis mereplikasi data Anda di beberapa Availability Zone (AZ) untuk ketersediaan tinggi. Secara otomatis memantau simpul cache yang mendasarinya dan menggantikannya jika terjadi kegagalan. Hal ini menawarkan SLA ketersediaan 99,99% untuk setiap cache.
- Upgrade perangkat lunak otomatis: ElastiCache Tanpa server secara otomatis meningkatkan cache Anda ke versi perangkat lunak minor dan patch terbaru tanpa dampak ketersediaan apa pun pada aplikasi Anda. Ketika versi utama Redis baru tersedia, ElastiCache akan mengirimkan pemberitahuan kepada Anda.
- Keamanan: Nirserver selalu mengenkripsi data bergerak dan saat tidak aktif. Anda dapat menggunakan kunci yang dikelola layanan atau menggunakan Kunci Dikelola Pelanggan Anda sendiri untuk mengenkripsi data diam.

Diagram berikut menggambarkan bagaimana ElastiCache Serverless bekerja.




Saat Anda membuat cache tanpa server baru, ElastiCache buat Titik Akhir Virtual Private Cloud (VPC) Virtual Private Cloud (VPC) di subnet pilihan Anda di VPC Anda. Aplikasi Anda dapat terhubung ke cache melalui Titik Akhir VPC ini.

Dengan ElastiCache Tanpa Server Anda menerima satu titik akhir DNS yang terhubung dengan aplikasi Anda. Saat Anda meminta koneksi baru ke titik akhir, ElastiCache Tanpa Server menangani semua koneksi cache melalui lapisan proxy. Lapisan proxy membantu mengurangi konfigurasi klien yang kompleks, karena klien tidak perlu menemukan kembali topologi klaster jika terjadi perubahan pada klaster yang mendasarinya. Lapisan proksi adalah sekumpulan simpul proksi yang menangani koneksi menggunakan penyeimbang beban jaringan. Saat aplikasi Anda membuat koneksi cache baru, permintaan dikirim ke simpul proxy oleh penyeimbang beban jaringan. Ketika aplikasi Anda mengeksekusi perintah cache, simpul proxy yang terhubung ke aplikasi Anda mengeksekusi permintaan di simpul cache di cache Anda. Lapisan proxy mengabstraksi topologi klaster cache dan simpul dari klien Anda. Hal ini memungkinkan ElastiCache untuk secara cerdas memuat keseimbangan, skala keluar dan menambahkan node cache baru, mengganti node cache ketika mereka gagal, dan memperbarui perangkat lunak pada node cache, semua tanpa dampak ketersediaan ke aplikasi Anda atau harus mengatur ulang koneksi.

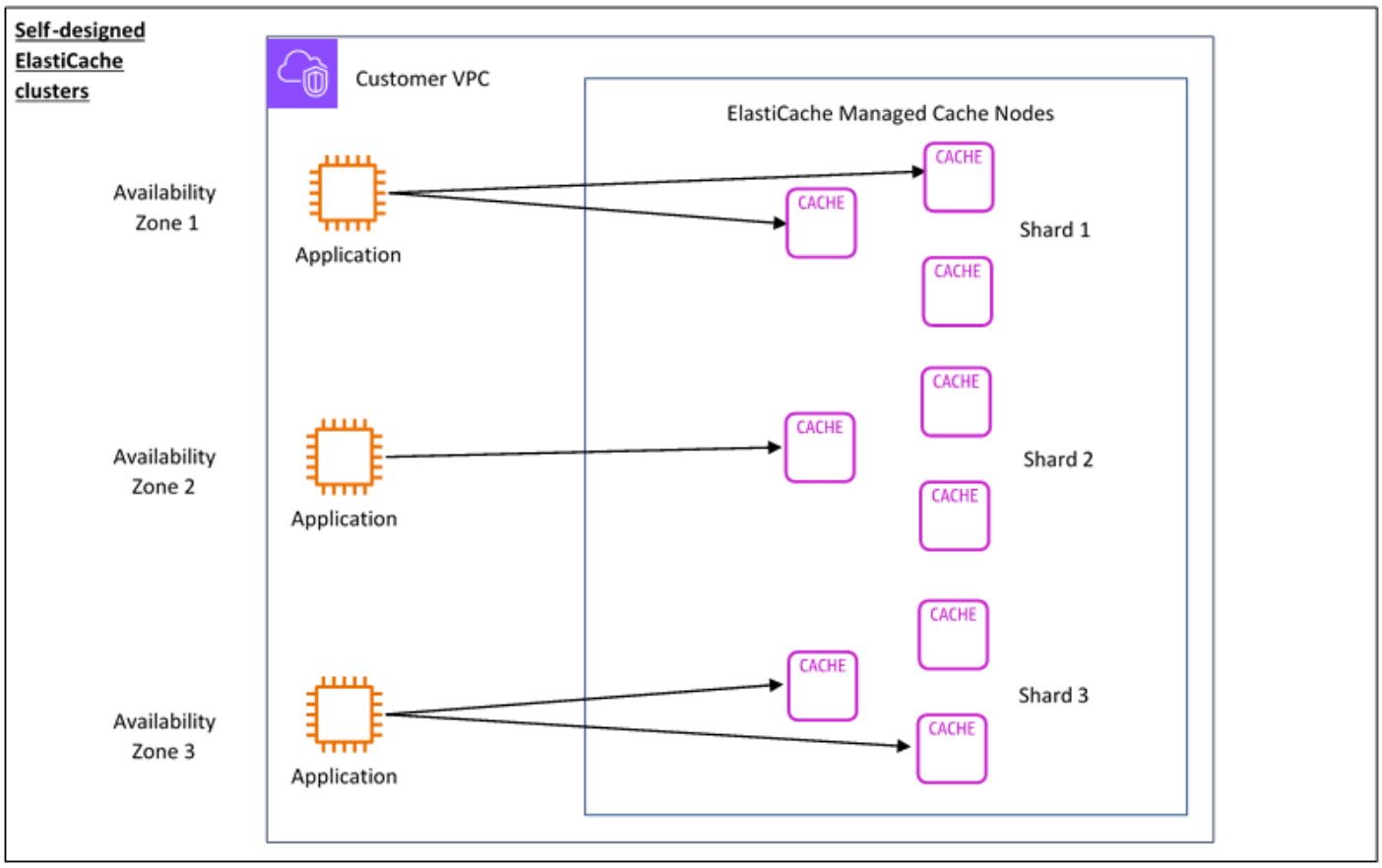
Cluster yang dirancang sendiri ElastiCache

Anda dapat memilih untuk mendesain ElastiCache cluster Anda sendiri dengan memilih keluarga node cache, ukuran, dan jumlah node untuk cluster Anda. Merancang klaster Anda sendiri memberi Anda kontrol berbutir lebih halus dan memungkinkan Anda memilih jumlah serpihan di cache Anda dan jumlah simpul (primer dan replika) di setiap serpihan. Anda dapat memilih untuk mengoperasikan Redis dalam mode klaster dengan membuat klaster dengan beberapa serpihan, atau dalam mode non-klaster dengan serpihan tunggal.

Manfaat Utama

- Desain cluster Anda sendiri: Dengan ElastiCache, Anda dapat mendesain cluster Anda sendiri dan memilih di mana Anda ingin menempatkan node cache Anda. Misalnya, jika Anda memiliki aplikasi yang ingin menukar ketersediaan tinggi dengan latensi rendah, Anda dapat memilih untuk menerapkan simpul cache Anda dalam satu AZ. Anda juga dapat mendesain klaster Anda dengan simpul di beberapa AZ untuk mencapai ketersediaan tinggi.
- Kontrol terperinci: Saat mendesain klaster sendiri, Anda memiliki kontrol yang lebih besar dalam menyempurnakan pengaturan cache Anda. Misalnya, Anda dapat menggunakan Parameter spesifik Redis untuk mengonfigurasi mesin cache.
- Menskalakan secara vertikal dan horizontal: Anda dapat memilih untuk menskalakan klaster Anda secara manual dengan menambah atau mengurangi ukuran simpul cache saat diperlukan. Anda juga dapat menskalakan secara horizontal dengan menambahkan serpihan baru atau menambahkan lebih banyak replika ke serpihan Anda. Anda juga dapat menggunakan fitur Auto-Scaling untuk mengonfigurasi penskalaan berdasarkan jadwal atau penskalaan berdasarkan metrik seperti penggunaan CPU dan Memori pada cache.

Diagram berikut menggambarkan bagaimana cluster yang ElastiCache dirancang sendiri bekerja.




## Dimensi harga

Anda dapat menerapkan ElastiCache dalam dua opsi penerapan. Saat menerapkan ElastiCache Tanpa Server, Anda membayar penggunaan untuk data yang disimpan dalam GB-jam dan menghitung di ElastiCache Unit Pemrosesan (ECPU). Saat memilih untuk mendesain sendiri ElastiCache untuk kluster Redis, Anda membayar per jam penggunaan node cache. Lihat detail harga di sini.

### Penyimpanan data

Anda membayar data yang disimpan dalam ElastiCache Serverless yang ditagih dalam gigabyte-hours (GB-HRs). ElastiCache Tanpa server terus memantau data yang disimpan dalam cache Anda, mengambil sampel beberapa kali per menit, dan menghitung rata-rata per jam untuk menentukan penggunaan penyimpanan data cache dalam GB-jam. Setiap cache ElastiCache Tanpa Server diukur untuk minimal 1 GB data yang disimpan.

### ElastiCache Unit Pemrosesan (ECPU)

Anda membayar permintaan Redis yang dijalankan aplikasi Anda di ElastiCache Tanpa Server di Unit ElastiCache Pemrosesan (ECPU), unit yang mencakup waktu vCPU dan data yang ditransfer.

- Bacaan dan penulisan sederhana membutuhkan 1 ECPU untuk setiap kilobyte (KB) data yang ditransfer. Misalnya, perintah GET yang mentransfer hingga 1 KB data mengonsumsi 1 ECPU. Permintaan SET yang mentransfer 3,2 KB data akan mengonsumsi 3,2 ECPU.
- Perintah yang membutuhkan waktu vCPU tambahan akan mengonsumsi lebih banyak ECPU secara proporsional. Sebagai contoh, jika aplikasi Anda menggunakan perintah HMGET Redis, dan menghabiskan 3 kali waktu vCPU sebagai perintah SET/GET sederhana, maka aplikasi akan menghabiskan 3 ECPU.
- Perintah yang mengonsumsi lebih banyak waktu vCPU dan mentransfer lebih banyak data mengonsumsi ECPU berdasarkan yang lebih tinggi dari dua dimensi. Misalnya, jika aplikasi Anda menggunakan perintah HMGET, mengonsumsi 3 kali waktu vCPU sebagai perintah SET/GET sederhana, dan mentransfer 3,2 KB data, perintah akan mengonsumsi 3,2 ECPU. Atau, jika data yang yang ditransfer hanya berukuran 2 KB, perintah akan mengonsumsi 3 ECPU.

ElastiCache Tanpa server memancarkan metrik baru yang disebut yang membantu Anda memahami ECPU `ElastiCacheProcessingUnits` yang dikonsumsi oleh beban kerja Anda.

Jam simpul

Anda dapat memilih untuk mendesain klaster cache Redis Anda sendiri dengan memilih keluarga simpul EC2, ukuran, jumlah simpul, dan penempatan di seluruh Zona Ketersediaan. Saat mendesain sendiri klaster Anda, Anda membayar per jam untuk setiap simpul cache.

### ElastiCache untuk cadangan Redis

Cadangan adalah point-in-time salinan cache Redis. ElastiCache memungkinkan Anda untuk mengambil cadangan data Anda kapan saja atau mengatur cadangan otomatis. Backup dapat digunakan untuk memulihkan cache yang telah ada atau menyemai cache baru. Cadangan terdiri dari semua data dalam cache ditambah beberapa metadata. Untuk informasi selengkapnya, lihat Melakukan snapshot dan pemulihan.

## Memilih antara opsi deployment

Amazon ElastiCache memiliki dua opsi penerapan:

- Caching nirserver

- Klaster yang dirancang sendiri

Untuk daftar perintah yang didukung untuk keduanya, lihatPerintah Redis yang didukung dan dibatasi.

Caching nirserver

Amazon ElastiCache Serverless menyederhanakan pembuatan cache dan menskalakan secara instan untuk mendukung aplikasi pelanggan yang paling menuntut. Dengan ElastiCache Tanpa Server, Anda dapat membuat cache yang sangat tersedia dan dapat diskalakan dalam waktu kurang dari satu menit, menghilangkan kebutuhan untuk menyediakan, merencanakan, dan mengelola kapasitas cluster cache. ElastiCache Tanpa server secara otomatis menyimpan data secara berlebihan di tiga Availability Zone dan menyediakan Perjanjian Tingkat Layanan (SLA) ketersediaan 99,99%. Cadangan kompatibel silang, dan dapat diekspor ke dan dipulihkan dari cluster yang dirancang sendiri.

Klaster yang dirancang sendiri

Jika Anda memerlukan kontrol halus atas cluster Redis Anda, Anda dapat memilih ElastiCache untuk mendesain cluster Redis Anda sendiri. ElastiCache ElastiCache memungkinkan Anda untuk mengoperasikan cluster berbasis node, dengan memilih tipe node, jumlah node, dan penempatan node di seluruh AWS Availability Zones untuk cluster Anda. Karena ElastiCache merupakan layanan yang dikelola sepenuhnya, ini membantu mengelola penyediaan perangkat keras, pemantauan, penggantian node, dan penambalan perangkat lunak untuk cluster Anda. Cluster yang dirancang sendiri dapat dirancang untuk memberikan SLA ketersediaan hingga 99,99%. Cadangan kompatibel silang, dan dapat diekspor ke dan dipulihkan dari cache Serveless.

Memilih antara opsi deployment

Pilih caching nirserver jika:

- Anda membuat cache untuk beban kerja yang baru atau sulit diprediksi.
- Anda memiliki lalu lintas aplikasi yang tak terduga.
- Anda ingin cara termudah untuk memulai dengan cache.

Pilih untuk mendesain ElastiCache cluster Anda sendiri jika:

- Anda sudah menjalankan ElastiCache Tanpa Server dan menginginkan kontrol yang lebih halus atas jenis node yang menjalankan Redis, jumlah node, dan penempatan node.

- Anda mengharapkan lalu lintas aplikasi Anda relatif dapat diprediksi, dan Anda ingin kontrol yang baik atas kinerja, ketersediaan, dan biaya.
- Anda dapat memperkirakan persyaratan kapasitas Anda untuk mengontrol biaya.

## Membandingkan caching tanpa server dan cluster yang dirancang sendiri

| Fitur | Caching nirserver | Klaster yang dirancang sendiri |
|---|---|---|
| Penyiapan cache | Buat cache hanya dengan nama dalam waktu kurang dari satu menit | Menyediakan kontrol halus atas desain cluster cache. Pengguna dapat memilih tipe node, jumlah node, dan penempatan di seluruh zona ketersediaan AWS |
| Didukung ElastiCache untuk versi Redis | ElastiCache untuk Redis versi 7.1 dan lebih tinggi | ElastiCache untuk Redis versi 4.0 dan lebih tinggi |
| Mode Cluster | Mengoperasikan Redis `cluster mode enabled` hanya di. Klien Redis harus mendukung `cluster mode enabled` untuk terhubung ke Tanpa ElastiCache Server. | Dapat dikonfigurasi untuk beroperasi dalam mode cluster diaktifkan atau mode cluster dinonaktifkan. |
| Penskalaan | Secara otomatis menskalakan baik secara vertikal maupun horizontal tanpa manajemen kapasitas apa pun. | Memberikan kontrol atas penskalaan, sementara juga membutuhkan pemantauan untuk memastikan kapasitas saat ini memenuhi permintaan secara memadai.<br><br>Anda dapat memilih untuk menskalakan secara vertikal, dengan menambah atau mengurangi ukuran node cache saat diperlukan. Anda |

| Fitur | Caching nirserver | Klaster yang dirancang sendiri |
| --- | --- | --- |
| | | juga dapat menskalakan secara horizontal, dengan menambahkan pecahan baru atau menambahkan lebih banyak replika ke pecahan Anda.<br><br>Dengan fitur Auto-Scaling, Anda juga dapat mengonfig urasi penskalaan berdasarkan jadwal, atau skala berdasark an metrik seperti penggunaan CPU dan Memori pada cache. |
| Koneksi klien | Klien terhubung ke satu titik akhir. Ini memungkinkan topologi node cache yang mendasarinya (penskalaan, penggantian, dan peningkat an) berubah tanpa memutuska n sambungan klien. | Klien terhubung ke setiap node cache individu. Jika sebuah node diganti, klien menemukan kembali topologi cluster dan membangun kembali koneksi. |
| Konfigurasi | Tidak ada konfigurasi berbutir halus yang tersedia. Pelanggan dapat mengkonfi gurasi pengaturan dasar termasuk subnet yang dapat mengakses cache, apakah backup otomatis diaktifkan atau dimatikan, dan batas penggunaan cache maksimum. | Cluster yang dirancang sendiri menyediakan opsi konfigura si berbutir halus. Pelanggan dapat menggunakan grup parameter untuk kontrol berbutir halus. Untuk tabel nilai parameter berdasarkan jenis simpul, lihat Parameter khusus jenis simpul Redis. |

| Fitur | Caching nirserver | Klaster yang dirancang sendiri |
| --- | --- | --- |
| Multi-AZ | Data direplikasi secara asinkron di beberapa Availability Zone untuk ketersediaan yang lebih tinggi dan latensi baca yang lebih baik. | Menyediakan opsi untuk mendesain cluster dalam satu Availability Zone atau di beberapa Availability Zone (AZ). Untuk klaster multi-AZ, data direplikasi secara asinkron di beberapa Availability Zone untuk ketersediaan yang lebih tinggi dan latensi baca yang lebih baik. |
| Enkripsi diam | Selalu diaktifkan. Pelanggan dapat menggunakan Kunci yang dikelola AWS atau kunci yang dikelola pelanggan AWS KMS. | Opsi untuk mengaktifkan atau menonaktifkan enkripsi saat istirahat. Ketika diaktifkan, pelanggan dapat menggunakan Kunci yang dikelola AWS atau kunci yang dikelola pelanggan AWS KMS. |
| Enkripsi dalam perjalanan (TLS) | Selalu diaktifkan. Klien harus mendukung konektivitas TLS. | Opsi untuk mengaktifkan atau menonaktifkan. |

| Fitur | Caching nirserver | Klaster yang dirancang sendiri |
| --- | --- | --- |
| Cadangan | Mendukung pencadangan cache otomatis dan manual tanpa dampak kinerja.<br><br>Cadangan kompatibel silang, dan dapat dikembalikan ke cache ElastiCache Tanpa Server atau cluster yang dirancang sendiri. | Mendukung backup otomatis dan manual. Cluster mungkin melihat beberapa dampak kinerja tergantung pada memori cadangan yang tersedia. Untuk informasi selengkapnya, lihat Mengelola Memori Terpesan.<br><br>Cadangan kompatibel silang, dan dapat dikembalikan ke cache ElastiCache Tanpa Server atau cluster yang dirancang sendiri. |
| Pemantauan | Mendukung metrik tingkat cache termasuk tingkat hit cache, tingkat kehilanga n cache, ukuran data, dan ECPU yang dikonsumsi.<br><br>ElastiCache Tanpa server mengirim peristiwa menggunakan EventBridge saat peristiwa penting terjadi di cache Anda. Anda dapat memilih untuk memantau, menelan, mengubah, dan menindaklanjuti ElastiCache acara menggunakan Amazon EventBridge. Untuk informasi selengkapnya, lihat Peristiwa cache nirserver. | ElastiCache Cluster yang dirancang sendiri memancark an metrik di setiap tingkat node, termasuk metrik tingkat host dan metrik cache.<br><br>Cluster yang dirancang sendiri memancarkan pemberitahuan SNS untuk acara penting. Lihat Metrik untuk Redis. |

| Fitur | Caching nirserver | Klaster yang dirancang sendiri |
| --- | --- | --- |
| Ketersediaan | 99,99% ketersediaan Perjanjian Tingkat Layanan (SLA) | Cluster yang dirancang sendiri dapat dirancang untuk mencapai ketersediaan hingga 99,99% Perjanjian Tingkat Layanan (SLA), tergantung pada konfigurasi. |
| Peningkatan dan penambalan perangkat lunak | Secara otomatis memutakhirkan perangkat lunak cache ke versi minor dan patch terbaru, tanpa dampak aplikasi. Pelanggan menerima pemberitahuan untuk peningkatan versi utama, dan pelanggan dapat meningkatkan ke versi utama terbaru kapan pun mereka mau. | Cluster yang dirancang sendiri menawarkan layanan mandiri yang diaktifkan pelanggan untuk peningkatan versi minor dan patching, serta peningkatan versi utama. Pembaruan terkelola diterapkan secara otomatis selama jendela pemeliharaan yang ditentukan pelanggan. Pelanggan juga dapat memilih untuk menerapkan upgrade versi minor atau patch sesuai permintaan. |
| Toko Data Global | Tidak didukung | Mendukung Global Data Store, yang memungkinkan replikasi lintas wilayah dengan penulisan wilayah tunggal dan pembacaan multi-wilayah |

| Fitur | Caching nirserver | Klaster yang dirancang sendiri |
| --- | --- | --- |
| Tingkat Data | Tidak didukung | Cluster yang dirancang menggunakan node dari keluarga r6gd memiliki data berjenjang antara memori dan penyimpanan SSD lokal (solid state drive). Tiering data menyediakan opsi harga-kin erja untuk beban kerja Redis dengan memanfaatkan solid state drive (SSD) berbiaya rendah di setiap node cluster, selain menyimpan data dalam memori. |
| Model penetapan harga | Pay-per-use, berdasarkan data yang disimpan dalam GB-jam dan permintaan di Unit ElastiCache Pemrosesa n (ECPU). Lihat detail harga di sini. | Pay-per-hour, berdasarkan penggunaan node cache. Lihat detail harga di sini. |

Topik terkait:

- Merancang dan mengelola klaster ElastiCache Anda sendiri

## Sumber daya Amazon ElastiCache

Sebaiknya Anda memulai dengan membaca bagian berikut, dan merujuk pada bagian tersebut saat diperlukan:

- Sorotan dan harga layanan – Halaman detail produk memberikan gambaran produk ElastiCache secara umum, sorotan layanan, dan harga.
- Video ElastiCache — Bagian Video ElastiCache memiliki video yang memperkenalkan Anda ke Amazon ElastiCache. Video meliputi kasus penggunaan umum untuk ElastiCache dan demo cara

menggunakan ElastiCache untuk mengurangi latensi dan meningkatkan throughput untuk aplikasi Anda.

- Memulai – Bagian Memulai dengan Amazon ElastiCache untuk Redis mencakup informasi tentang membuat klaster cache. Ini juga mencakup cara memberikan otorisasi akses ke klaster cache, menyambung ke simpul cache, dan menghapus klaster cache.
- Kinerja sesuai skala – Kinerja sesuai skala dengan Amazon ElastiCache laporan resmi yang membahas strategi caching yang memungkinkan aplikasi Anda bekerja dengan baik pada skalanya.

Jika Anda ingin menggunakan AWS Command Line Interface (AWS CLI), Anda dapat menggunakan dokumen ini untuk membantu Anda memulai:

- Dokumentasi AWS Command Line Interface

  Bagian ini memberikan informasi tentang mengunduh AWS CLI, membuat AWS CLI bekerja pada sistem Anda, dan memberikan kredensial AWS Anda.
- Dokumentasi AWS CLI untuk ElastiCache

  Dokumen terpisah ini mencakup semua hal mengenai AWS CLI untuk perintah ElastiCache, termasuk sintaks dan contoh.

Anda dapat membuat program aplikasi menggunakan API ElastiCache dengan berbagai bahasa pemrograman yang populer. Berikut adalah beberapa sumber daya:

- Alat untuk Amazon Web Services

  Amazon Web Services menyediakan sejumlah kit pengembangan perangkat lunak (SDK) dengan dukungan untuk ElastiCache. Anda dapat membuat kode untuk ElastiCache menggunakan Java, .NET, PHP, Ruby, dan bahasa lainnya. SDK ini dapat secara signifikan menyederhanakan pengembangan aplikasi Anda dengan memformat permintaan Anda untuk ElastiCache, melakukan parsing tanggapan, serta memberikan logika coba lagi dan penanganan kesalahan.
- Menggunakan API ElastiCache

  Jika Anda tidak ingin menggunakan SDK AWS, Anda dapat berinteraksi dengan ElastiCache secara langsung menggunakan API Kueri. Pada bagian ini, Anda dapat menemukan tips pemecahan masalah dan informasi tentang membuat dan melakukan autentikasi atas permintaan serta menangani tanggapan.

- Referensi API Amazon ElastiCache

  Dokumen terpisah ini mencakup semua hal mengenai operasi API ElastiCache, termasuk sintaks dan contoh.

# Wilayah AWS dan Zona Ketersediaan

Sumber daya komputasi cloud Amazon berlokasi di fasilitas pusat data dengan ketersediaan tinggi di berbagai wilayah di dunia (misalnya, Amerika Utara, Eropa, atau Asia). Setiap lokasi pusat data disebut Wilayah AWS.

Setiap Wilayah AWS berisi beberapa lokasi berbeda yang disebut Zona Ketersediaan, atau AZs. Setiap Zona Ketersediaan dirancang agar terisolasi dari kegagalan di Zona Ketersediaan yang lain. Setiap AZ direkayasa untuk menyediakan konektivitas jaringan latensi rendah yang murah ke Zona Ketersediaan lainnya di dalam Wilayah AWS yang sama. Dengan meluncurkan instans di Zona Ketersediaan yang terpisah, Anda dapat melindungi aplikasi Anda dari kegagalan di satu lokasi. Untuk informasi selengkapnya, lihat Memilih wilayah dan zona ketersediaan.

Anda dapat menjalankan klaster Anda di beberapa Zona Ketersediaan, yang merupakan opsi yang disebut deployment Multi-AZ. Saat Anda memilih opsi ini, Amazon secara otomatis menyediakan dan memelihara instans simpul siaga sekunder pada Zona Ketersediaan yang berbeda. Instans simpul primer Anda direplikasi secara asinkron di seluruh Zona Ketersediaan ke instans sekunder. Pendekatan ini membantu menyediakan redundansi data dan dukungan failover, menghilangkan macetnya I/O, dan meminimalkan lonjakan latensi selama pencadangan sistem. Untuk informasi selengkapnya, lihat Meminimalkan waktu henti di ElastiCache for Redis dengan Multi-AZ.

# Kasus Penggunaan ElastiCache Umum dan Cara ElastiCache Dapat Membantu

Baik ketika menyajikan berita terkini, papan peringkat 10 teratas, katalog produk, maupun menjual tiket suatu acara, kecepatan adalah aspek yang terpenting. Keberhasilan situs web dan bisnis Anda sangat dipengaruhi oleh kecepatan Anda dalam menyediakan konten.

Dalam artikel "For Impatient Web Users, an Eye Blink Is Just Too Long to Wait", New York Times menjelaskan bahwa pengguna dapat merasakan perbedaan 250 milidetik (1/4 detik) di antara beberapa situs yang bersaing. Pengguna cenderung lebih memilih situs yang lebih cepat daripada situs yang lambat. Pengujian yang dilakukan di Amazon, yang dikutip dalam artikel How Webpage Load Time Is Related to Visitor Loss, mengungkapkan bahwa untuk setiap 100 milidetik (1/10 detik) penambahan waktu pemuatan, penjualan turun 1 persen.

Jika seseorang menginginkan data, Anda dapat mengirimkan data tersebut lebih cepat jika disimpan dalam cache. Hal tersebut berlaku baik untuk halaman web maupun laporan yang mendorong keputusan bisnis. Apakah bisnis Anda mampu menampilkan halaman web Anda dengan latensi yang sesingkat mungkin tanpa menyimpannya dalam cache?

Tanpa perlu dipikir lagi, Anda pasti ingin meng-cache item Anda yang paling banyak diminta. Namun, mengapa tidak meng-cache item yang kurang sering diminta? Bahkan kueri atau panggilan API jarak jauh ke basis data yang paling dioptimalkan sekalipun akan terasa lebih lambat daripada mengambil satu kunci datar dari cache dalam memori. Pemuatan yang terasa lebih lambat cenderung membuat pelanggan beralih ke bisnis lain.

Contoh berikut mengilustrasikan beberapa cara penggunaan ElastiCache untuk meningkatkan performa aplikasi Anda secara keseluruhan.

Topik

# Penyimpanan Data Dalam Memori

Tujuan utama dari penyimpanan nilai-kunci dalam memori adalah untuk menyediakan akses ultracepat (latensi submilidetik) dan murah ke salinan data, Kebanyakan penyimpanan data memiliki area data yang sering diakses namun jarang diperbarui. Selain itu, kueri basis data selalu lebih lambat dan lebih mahal daripada menemukan kunci dalam cache pasangan nilai kunci. Beberapa kueri basis data khususnya mahal untuk dijalankan. Contohnya adalah kueri yang memerlukan operasi join terhadap beberapa tabel atau kueri dengan penghitungan intensif. Dengan meng-cache hasil kueri tersebut, Anda membayar harga kueri hanya sekali. Kemudian, Anda dapat dengan cepat mengambil data beberapa kali tanpa harus menjalankan kembali kueri tersebut.

## Apa yang Harus Saya Simpan dalam Cache?

Saat menentukan data apa yang harus di-cache, pertimbangkan beberapa faktor ini:

Kecepatan dan biaya – Selalu akan lebih lambat dan lebih mahal untuk mendapatkan data dari basis data dibandingkan dari cache. Beberapa kueri basis data secara inheren lebih lambat dan lebih mahal daripada yang lain. Misalnya, kueri yang melakukan operasi join terhadap beberapa tabel akan jauh lebih lambat dan lebih mahal daripada kueri tabel tunggal sederhana. Jika data yang diperlukan hanya dapat diperoleh menggunakan kueri yang lambat dan mahal, berarti data ini kemungkinan cocok untuk caching. Jika data dapat diambil dengan kueri yang relatif cepat dan sederhana, data tersebut mungkin masih dapat menjadi kandidat untuk caching, tergantung pada beberapa faktor lainnya.

Data dan pola akses – Untuk menentukan apa yang perlu di-cache, diperlukan juga pemahaman terhadap data itu sendiri dan pola aksesnya. Misalnya, tidak masuk akal untuk meng-cache data yang berubah dengan cepat atau jarang diakses. Agar proses cache menyediakan manfaat nyata, data harus relatif statis dan sering diakses. Contohnya adalah profil pribadi di situs media sosial. Di sisi lain, Anda tidak ingin meng-cache data jika caching tidak menyediakan kecepatan atau keuntungan biaya. Misalnya, tidak masuk akal untuk meng-cache halaman web yang menampilkan hasil pencarian karena kueri dan hasilnya biasanya unik.

Staleness – Menurut definisi, data yang di-cache adalah data usang (stale). Meskipun dalam keadaan tertentu data tidak usang, namun data ini harus selalu dianggap dan diperlakukan sebagai data usang. Untuk mengetahui apakah data Anda adalah kandidat untuk caching, tentukan toleransi aplikasi Anda untuk data usang.

Aplikasi Anda mungkin dapat menoleransi data usang dalam satu konteks, tetapi tidak untuk yang lain. Misalnya, anggaplah situs Anda menyajikan harga saham yang diperdagangkan secara publik.

Pelanggan Anda mungkin bisa menerima beberapa data usang jika diberi klarifikasi bahwa data harga mungkin tertunda n menit. Namun, jika Anda menyajikan harga saham tersebut kepada pialang saham yang melakukan penjualan atau pembelian, maka Anda menginginkan data waktu nyata.

Pertimbangkan untuk meng-cache data Anda jika hal berikut ini berlaku:

- Data Anda lambat atau mahal untuk diperoleh jika dibandingkan dengan pengambilan dari cache.
- Pengguna sering mengakses data Anda.
- Data Anda tetap relatif sama, atau jika data berubah dengan cepat, "staleness" bukanlah masalah besar.

Untuk informasi selengkapnya tentang IAM, lihat hal berikut:

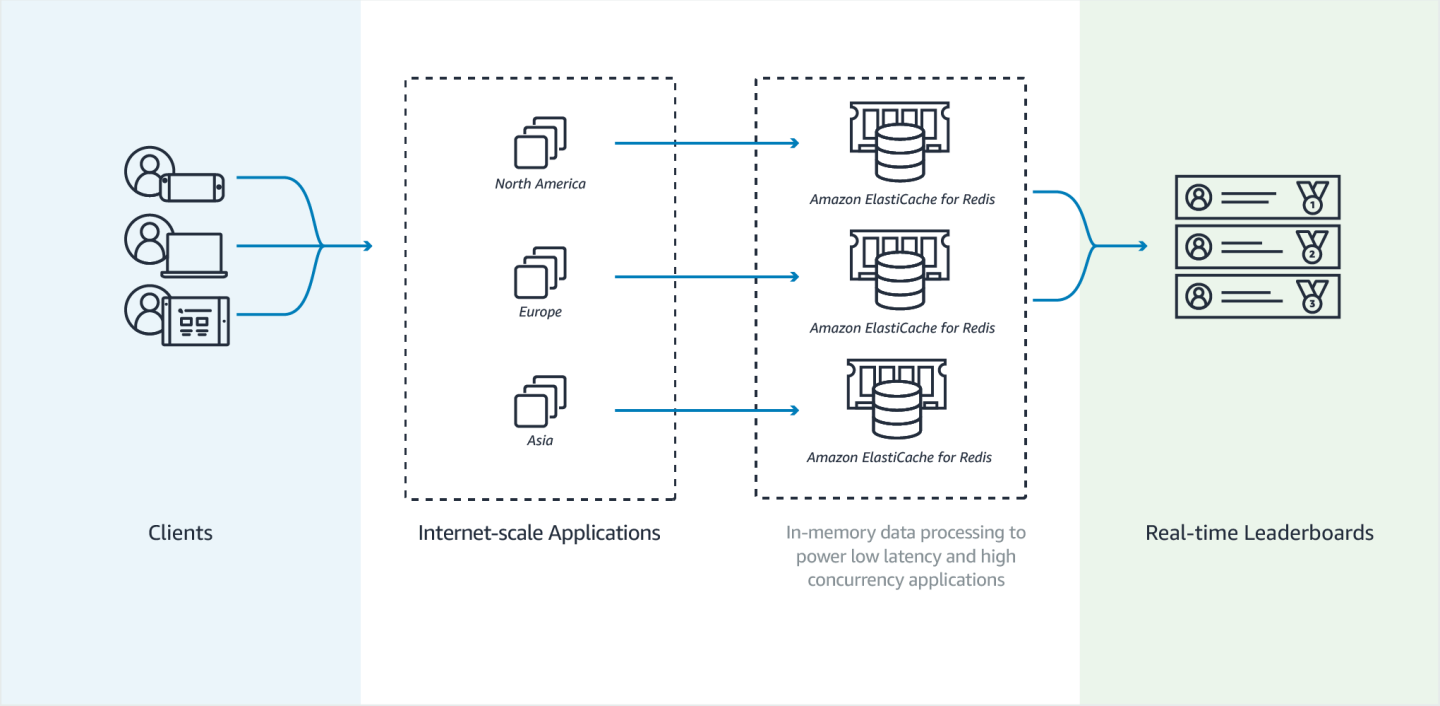
- Strategi Caching dalam Panduan Pengguna ElastiCache for Redis

## Papan Peringkat Game (Set Berurut Redis)

Set berurut Redis memindahkan kompleksitas komputasi papan peringkat dari aplikasi Anda ke klaster Redis Anda.

Papan peringkat, seperti 10 skor teratas untuk sebuah game memiliki komputasi yang rumit. Hal ini terutama terjadi jika ada banyak pemain sekaligus dan skor terus berubah. Set berurut Redis menjamin baik keunikan maupun pengurutan elemen. Saat menggunakan set berurut Redis, setiap kali elemen baru ditambahkan ke set berurut, maka peringkatnya akan diurutkan kembali secara waktu nyata. Elemen tersebut kemudian ditambahkan ke set dalam urutan numerik yang benar.

Pada diagram berikut, Anda dapat melihat cara kerja papan peringkat game ElastiCache for Redis.




Example - Papan Peringkat Redis

Dalam contoh ini, empat pemain game dan skor mereka dimasukkan ke dalam daftar berurut menggunakan ZADD. Perintah ZREVRANGEBYSCORE mengurutkan para pemain sesuai skor mereka, dari tinggi ke rendah. Selanjutnya, ZADD digunakan untuk memperbarui skor milik June dengan menimpa entri yang sudah ada. Terakhir, ZREVRANGEBYSCORE mengurutkan para pemain sesuai skor mereka, dari tinggi ke rendah. Daftar ini menunjukkan bahwa June telah naik dalam peringkat.

```
ZADD leaderboard 132 Robert
ZADD leaderboard 231 Sandra
ZADD leaderboard 32 June
ZADD leaderboard 381 Adam

ZREVRANGEBYSCORE leaderboard +inf -inf
1) Adam
2) Sandra
3) Robert
4) June

ZADD leaderboard 232 June

ZREVRANGEBYSCORE leaderboard +inf -inf
1) Adam
2) June
```

```
3) Sandra
4) Robert
```

Perintah berikut menunjukkan peringkat June di antara semua pemain. Karena peringkat adalah berbasis nol, ZREVRANK menampilkan nilai 1 untuk June, yang berada di posisi kedua.

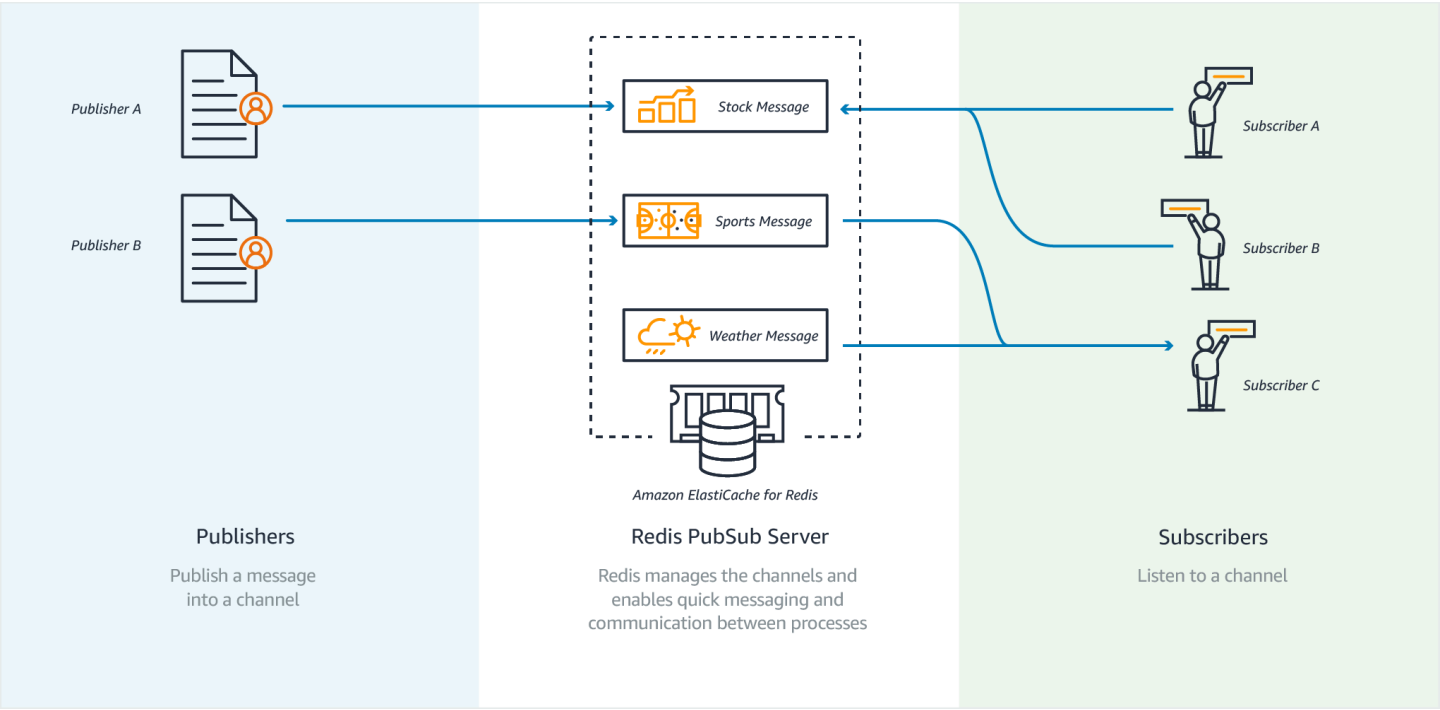
```
ZREVRANK leaderboard June
1
```

Untuk informasi selengkapnya, lihat Dokumentasi Redis tentang set berurut.

## Pesan (Pub/Sub Redis)

Ketika Anda mengirim pesan email, Anda mengirimkannya ke satu atau beberapa penerima tertentu. Dalam paradigma pub/sub, Anda mengirim pesan ke saluran tertentu tanpa diketahui siapa penerimanya, jika ada. Orang-orang yang mendapatkan pesan adalah mereka yang berlangganan saluran tersebut. Misalnya, anggaplah Anda berlangganan saluran news.sports.golf. Anda dan semua orang yang berlangganan saluran news.sports.golf mendapatkan semua pesan yang diterbitkan ke news.sports.golf.

Fungsionalitas pub/sub Redis tidak terkait dengan ruang kunci mana pun. Oleh karena itu, fungsi ini sama sekali tidak memberikan pengaruh. Dalam diagram berikut, Anda dapat menemukan ilustrasi pesan ElastiCache for Redis.



## Berlangganan

Untuk menerima pesan pada saluran, Anda berlangganan saluran tersebut. Anda dapat berlangganan satu saluran, beberapa saluran tertentu, atau semua saluran yang sesuai dengan pola. Untuk membatalkan langganan, Anda berhenti berlangganan dari saluran yang ditentukan saat Anda berlangganan. Atau, jika berlangganan menggunakan pencocokan pola, Anda berhenti berlangganan menggunakan pola yang sama seperti yang Anda gunakan sebelumnya.

Example - Langganan ke Satu Saluran

Untuk berlangganan satu saluran, gunakan perintah SUBSCRIBE yang menentukan saluran yang diinginkan. Pada contoh berikut, klien berlangganan saluran news.sports.golf.

```
SUBSCRIBE news.sports.golf
```

Setelah beberapa saat, klien ini membatalkan langganannya ke saluran tersebut menggunakan perintah UNSUBSCRIBE yang menentukan saluran yang akan dihentikan langganannya.

```
UNSUBSCRIBE news.sports.golf
```

Example - Berlangganan ke Beberapa Saluran Tertentu

Untuk berlangganan beberapa saluran tertentu, tampilkan daftar saluran dengan perintah SUBSCRIBE. Pada contoh berikut, klien berlangganan saluran news.sports.golf, news.sports.soccer, dan news.sports.skiing.

```
SUBSCRIBE news.sports.golf news.sports.soccer news.sports.skiing
```

Untuk membatalkan langganan ke saluran tertentu, gunakan perintah UNSUBSCRIBE dan tentukan saluran yang akan dihentikan langganannya.

```
UNSUBSCRIBE news.sports.golf
```

Untuk membatalkan langganan ke beberapa saluran, gunakan perintah UNSUBSCRIBE dan tentukan saluran-saluran yang akan dihentikan langganannya.

```
UNSUBSCRIBE news.sports.golf news.sports.soccer
```

Untuk membatalkan semua langganan, gunakan UNSUBSCRIBE dan tentukan setiap saluran. Atau gunakan UNSUBSCRIBE dan jangan menentukan saluran.

```
UNSUBSCRIBE news.sports.golf news.sports.soccer news.sports.skiing
```

atau

```
UNSUBSCRIBE
```

Example - Langganan Menggunakan Pencocokan Pola

Klien dapat berlangganan semua saluran yang cocok dengan sebuah pola dengan menggunakan perintah PSUBSCRIBE.

Pada contoh berikut, klien berlangganan semua saluran olahraga. Anda tidak menampilkan daftar semua saluran olahraga satu per satu, seperti yang Anda gunakan dengan SUBSCRIBE. Namun, dengan perintah PSUBSCRIBE Anda menggunakan pencocokan pola.

```
PSUBSCRIBE news.sports.*
```

Example Membatalkan Langganan

Untuk membatalkan langganan ke saluran tersebut, gunakan perintah PUNSUBSCRIBE.

```
PUNSUBSCRIBE news.sports.*
```

> **Important**
>
> String saluran yang dikirim ke perintah [P]SUBSCRIBE dan ke perintah [P]UNSUBSCRIBE harus cocok. Anda tidak dapat menjalankan PSUBSCRIBE ke news.* dan PUNSUBSCRIBE dari news.sports.* atau UNSUBSCRIBE dari news.sports.golf.

## Penerbitan

Untuk mengirim pesan ke semua pelanggan saluran, gunakan perintah PUBLISH, dengan menentukan saluran dan pesan. Contoh berikut menerbitkan pesan, "It's Saturday and sunny. I'm headed to the links." ke saluran news.sports.golf.

```
PUBLISH news.sports.golf "It's Saturday and sunny. I'm headed to the links."
```

Klien tidak dapat menerbitkan pesan ke saluran yang dilanggani klien tersebut.

Untuk informasi selengkapnya, lihat Pub/Sub dalam dokumentasi Redis.

## Data Rekomendasi (Hash Redis)

Penggunaan INCR atau DECR di Redis membuat kompilasi rekomendasi menjadi sederhana. Setiap kali pengguna "menyukai" produk, Anda menaikkan nilai penghitung item:productID:like. Setiap kali pengguna "tidak menyukai" produk, Anda menaikkan nilai penghitung item:productID:dislike. Dengan hash Redis, Anda juga dapat memelihara daftar semua orang yang telah menyukai atau tidak menyukai produk.

Example - Suka dan Tidak Suka

```
INCR item:38923:likes
HSET item:38923:ratings Susan 1
INCR item:38923:dislikes
HSET item:38923:ratings Tommy -1
```

## Penggunaan Redis Lainnya

Postingan blog How to take advantage of Redis just adding it to your stack oleh Salvatore Sanfilippo membahas sejumlah masalah umum basis data dan cara menyelesaikannya dengan mudah menggunakan Redis. Pendekatan ini menghapus beban dari basis data Anda dan meningkatkan performa.

## Testimoni Pelanggan ElastiCache

Untuk mempelajari bagaimana bisnis seperti Airbnb, PBS, Esri, dan lainnya menggunakan Amazon ElastiCache untuk mengembangkan bisnis mereka dengan pengalaman pelanggan yang disempurnakan, lihat Cara Orang Lain Menggunakan Amazon ElastiCache.

Anda juga dapat menonton Video tutorial ini untuk kasus penggunaan pelanggan ElastiCache lainnya.

# Memulai dengan Amazon ElastiCache untuk Redis

Gunakan tutorial langsung di bagian ini untuk membantu Anda memulai dan mempelajari lebih lanjut tentang Redis ElastiCache .

Topik



## Pengaturan

Untuk mengatur ElastiCache:

Topik



### Mendaftar untuk Akun AWS

Jika Anda tidak memiliki Akun AWS, selesaikan langkah-langkah berikut untuk membuatnya.

Untuk mendaftar untuk Akun AWS

1. Buka https://portal.aws.amazon.com/billing/signup.

2. Ikuti petunjuk secara online.

   Anda akan diminta untuk menerima panggilan telepon dan memasukkan kode verifikasi pada keypad telepon sebagai bagian dari prosedur pendaftaran.

   Saat Anda mendaftar untuk sebuah Akun AWS, sebuah Pengguna root akun AWSdibuat. Pengguna root memiliki akses ke semua Layanan AWS dan sumber daya dalam akun. Sebagai praktik terbaik keamanan, tetapkan akses administratif ke pengguna administratif, dan hanya gunakan pengguna root untuk melakukan tugas yang memerlukan akses pengguna root.

AWS mengirimi Anda email konfirmasi setelah proses pendaftaran selesai. Anda dapat melihat aktivitas akun saat ini dan mengelola akun dengan mengunjungi https://aws.amazon.com/ dan memilih Akun Saya.

## Membuat pengguna administratif

Setelah Anda mendaftar Akun AWS, amankan Pengguna root akun AWS, aktifkan AWS IAM Identity Center, dan buat pengguna administratif sehingga Anda tidak menggunakan pengguna root untuk tugas sehari-hari.

Amankan Anda Pengguna root akun AWS

1. Masuk ke AWS Management Consolesebagai pemilik akun dengan memilih pengguna Root dan memasukkan alamat Akun AWS email Anda. Di halaman berikutnya, masukkan kata sandi Anda.

   Untuk bantuan masuk menggunakan pengguna root, lihat Masuk sebagai pengguna root dalam Panduan Pengguna AWS Sign-In .
2. Aktifkan autentikasi multi-faktor (MFA) untuk pengguna root Anda.

   Untuk petunjuk, lihat Mengaktifkan perangkat MFA virtual untuk pengguna Akun AWS root (konsol) Anda di Panduan Pengguna IAM.

Membuat pengguna administratif

1. Aktifkan Pusat Identitas IAM.

   Untuk mendapatkan petunjuk, silakan lihat Mengaktifkan AWS IAM Identity Center di Panduan Pengguna AWS IAM Identity Center .
2. Di Pusat Identitas IAM, berikan akses administratif ke sebuah pengguna administratif.

Untuk tutorial tentang menggunakan Direktori Pusat Identitas IAM sebagai sumber identitas Anda, lihat Mengkonfigurasi akses pengguna dengan default Direktori Pusat Identitas IAM di Panduan AWS IAM Identity Center Pengguna.

Masuk sebagai pengguna administratif

- Untuk masuk dengan pengguna Pusat Identitas IAM, gunakan URL masuk yang dikirim ke alamat email Anda saat Anda membuat pengguna Pusat Identitas IAM.

  Untuk bantuan masuk menggunakan pengguna Pusat Identitas IAM, lihat Masuk ke portal AWS akses di Panduan AWS Sign-In Pengguna.

## Memberikan akses terprogram

Pengguna membutuhkan akses terprogram jika mereka ingin berinteraksi dengan AWS luar. AWS Management Console Cara untuk memberikan akses terprogram tergantung pada jenis pengguna yang mengakses AWS.

Untuk memberi pengguna akses terprogram, pilih salah satu opsi berikut.

| Pengguna mana yang membutuhkan akses terprogram? | Untuk | Oleh |
| --- | --- | --- |
| Identitas tenaga kerja<br><br>(Pengguna yang dikelola di Pusat Identitas IAM) | Gunakan kredensyal sementara untuk menandatangani permintaan terprogram ke AWS CLI, AWS SDK, atau API. AWS | Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan.<br><br>• Untuk AWS CLI, lihat Mengkonfigurasi yang akan AWS CLI digunakan AWS IAM Identity Center dalam Panduan AWS Command Line Interface Pengguna.<br>• Untuk AWS SDK, alat, dan AWS API, lihat autentikasi Pusat Identitas IAM di |

| Pengguna mana yang membutuhkan akses terprogram? | Untuk | Oleh |
| --- | --- | --- |
| | | Panduan Referensi AWS SDK dan Alat. |
| IAM | Gunakan kredensyal sementara untuk menandatangani permintaan terprogram ke AWS CLI, AWS SDK, atau API. AWS | Mengikuti petunjuk dalam Menggunakan kredensyal sementara dengan AWS sumber daya di Panduan Pengguna IAM. |
| IAM | (Tidak direkomendasikan) Gunakan kredensyal jangka panjang untuk menandatangani permintaan terprogram ke AWS CLI, AWS SDK, atau API. AWS | Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan.<br>• Untuk mengetahui AWS CLI, lihat Mengautentikasi menggunakan kredensyal pengguna IAM di Panduan Pengguna. AWS Command Line Interface<br>• Untuk AWS SDK dan alat bantu, lihat Mengautentikasi menggunakan kredensyal jangka panjang di Panduan Referensi AWS SDK dan Alat.<br>• Untuk AWS API, lihat Mengelola kunci akses untuk pengguna IAM di Panduan Pengguna IAM. |

Topik terkait:

- Apa itu IAM? dalam Panduan Pengguna IAM.

- AWS Kredensyal Keamanan dalam Referensi AWS Umum.

## Siapkan izin Anda (hanya ElastiCache pengguna baru)

Untuk memberikan akses, tambahkan izin ke pengguna, grup, atau peran Anda:

- Pengguna dan grup di AWS IAM Identity Center:

  Buat rangkaian izin. Ikuti petunjuk dalam Buat set izin dalam Panduan Pengguna AWS IAM Identity Center .
- Pengguna yang dikelola di IAM melalui penyedia identitas:

  Buat peran untuk federasi identitas. Ikuti petunjuk dalam Membuat peran untuk penyedia identitas pihak ketiga (federasi) di Panduan Pengguna IAM.
- Pengguna IAM:
  - Buat peran yang dapat diambil pengguna Anda. Ikuti petunjuk dalam Membuat peran untuk pengguna IAM dalam Panduan Pengguna IAM.
  - (Tidak disarankan) Pasang kebijakan langsung ke pengguna atau tambahkan pengguna ke grup pengguna. Ikuti petunjuk di Menambahkan izin ke pengguna (konsol) dalam Panduan Pengguna IAM.

Amazon ElastiCache membuat dan menggunakan peran terkait layanan untuk menyediakan sumber daya dan mengakses sumber AWS daya dan layanan lain atas nama Anda. ElastiCache Untuk membuat peran terkait layanan untuk Anda, gunakan kebijakan AWS -managed bernama. `AmazonElastiCacheFullAccess` Peran ini dilengkapi sebelumnya dengan izin yang diperlukan layanan untuk membuat peran tertaut layanan untuk Anda.

Anda mungkin memutuskan untuk tidak menggunakan kebijakan default dan sebagai gantinya menggunakan kebijakan terkelola khusus. Dalam hal ini, pastikan bahwa Anda memiliki izin untuk memanggil `iam:createServiceLinkedRole` atau Anda telah membuat peran tertaut layanan ElastiCache.

Untuk informasi lain, lihat yang berikut ini:

- Membuat Kebijakan Baru (IAM)
- Kebijakan terkelola AWS untuk Amazon ElastiCache
- Menggunakan Peran Tertaut Layanan untuk Amazon ElastiCache

# Menyiapkan EC2

Anda perlu menyiapkan instans EC2 dari mana Anda akan terhubung ke cache Anda.

- Jika Anda belum memiliki instans EC2, pelajari cara menyiapkan instans EC2 di sini: Memulai EC2.
- Instans EC2 Anda harus berada di VPC yang sama dan memiliki pengaturan grup keamanan yang sama dengan cache Anda. Secara default, Amazon ElastiCache membuat cache di VPC default Anda dan menggunakan grup keamanan default. Untuk mengikuti tutorial ini, pastikan bahwa instans EC2 Anda ada di VPC default dan memiliki grup keamanan default.

## Berikan akses jaringan dari grup keamanan Amazon VPC ke cache Anda

ElastiCache Cluster yang dirancang sendiri menggunakan port 6379 untuk perintah Redis, dan ElastiCache tanpa server menggunakan port 6379 dan port 6380. Agar berhasil menghubungkan dan menjalankan perintah Redis dari instans EC2 Anda, grup keamanan Anda harus mengizinkan akses ke port ini sesuai kebutuhan.

1. Masuk ke AWS Command Line Interface dan buka konsol Amazon EC2.
2. Pada panel navigasi, di bagian Jaringan & Keamanan, pilih Grup Keamanan.
3. Dari daftar grup keamanan, pilih grup keamanan untuk Amazon VPC Anda. Kecuali Anda membuat grup keamanan untuk ElastiCache digunakan, grup keamanan ini akan diberi nama default.
4. Pilih tab Inbound, dan kemudian:

    a. Pilih Edit.

    b. Pilih Tambahkan aturan.

    c. Pada kolom Jenis, pilih Aturan TCP khusus.

    d. Di kotak rentang Port, ketik`6379`.

    e. Pada kotak Sumber, pilih Dari mana pun yang memiliki rentang port (0.0.0.0/0) sehingga setiap instans Amazon EC2 yang Anda luncurkan di dalam Amazon VPC Anda dapat terhubung ke cache Anda.

    f. Jika Anda menggunakan ElastiCache tanpa server, tambahkan aturan lain dengan memilih Tambahkan aturan.

    g. Pada kolom Jenis, pilih Aturan TCP Khusus.

    h. Di kotak rentang Port, ketik `6380`.

i. Pada kotak Sumber, pilih Dari mana pun yang memiliki rentang port (0.0.0.0/0) sehingga setiap instans Amazon EC2 yang Anda luncurkan di dalam Amazon VPC Anda dapat terhubung ke cache Anda.

j. Pilih Simpan

## Unduh dan atur redis-cli

1. Hubungkan ke instans Amazon EC2 Anda menggunakan utilitas koneksi pilihan Anda. Untuk petunjuk tentang cara menghubungkan ke instans Amazon EC2, lihat Panduan Memulai Amazon EC2.
2. Unduh dan instal utilitas redis-cli dengan menjalankan perintah yang sesuai untuk pengaturan Anda.

   Amazon Linux 2023

   ```
   sudo yum install redis6 -y
   ```

   Amazon Linux 2

   ```
   sudo amazon-linux-extras install epel -y
   sudo yum install gcc jemalloc-devel openssl-devel tcl tcl-devel -y
   sudo wget http://download.redis.io/redis-stable.tar.gz
   sudo tar xvzf redis-stable.tar.gz
   cd redis-stable
   sudo make BUILD_TLS=yes
   ```

> **Note**
>
> - Ketika Anda menginstal paket redis6, paket ini akan menginstal redis6-cli dengan dukungan enkripsi default.
> - Penting untuk memiliki dukungan build untuk TLS saat menginstal redis-cli. ElastiCache Tanpa server hanya dapat diakses saat TLS diaktifkan.
> - Jika Anda terhubung ke klaster yang tidak terenkripsi, Anda tidak memerlukan opsi. `Build_TLS=yes`

# Langkah 1: Buat Cache

Pada langkah ini, Anda membuat cache baru di Amazon ElastiCache.

AWS Management Console

Untuk membuat cache menggunakan konsol ElastiCache:

1. Masuk ke AWS Management Console dan buka https://console.aws.amazon.com/connect/.
2. Di panel navigasi pada bagian kiri konsol. pilih Cache Redis.
3. Di bagian kanan konsol, pilih Buat cache Redis
4. Pada Pengaturan cache masukkan Nama. Anda juga dapat memasukkan deskripsi untuk cache.
5. Biarkan Pengaturan default dipilih.
6. Klik Buat untuk membuat cache.
7. Setelah cache berada dalam status “AKTIF”, Anda dapat mulai melakukan operasi baca-tulis ke cache.

AWS CLI

Contoh AWS CLI berikut membuat cache baru menggunakan create-serverless-cache.

Linux

```
aws elasticache create-serverless-cache \
    --serverless-cache-name CacheName \
    --engine redis
```

Windows

```
aws elasticache create-serverless-cache ^
    --serverless-cache-name CacheName ^
    --engine redis
```

Perhatikan bahwa nilai bidang Status diatur ke `CREATING`.

Untuk memverifikasi bahwa ElastiCache selesai membuat cache, gunakan perintah `describe-serverless-caches`.

Linux

```
aws elasticache describe-serverless-caches --serverless-cache-name CacheName
```

Windows

```
aws elasticache describe-serverless-caches --serverless-cache-name CacheName
```

Setelah membuat cache baru, lanjutkan ke Langkah 2: Baca dan tulis data ke cache.

# Langkah 2: Baca dan tulis data ke cache

Bagian ini mengasumsikan bahwa Anda telah membuat instans Amazon EC2 dan dapat tersambung ke instans tersebut. Untuk petunjuk cara melakukannya, lihat Panduan Memulai Amazon EC2.

Bagian ini juga mengasumsikan bahwa Anda memiliki akses pengaturan VPC dan pengaturan grup keamanan untuk instans EC2 dari tempat Anda tersambung ke cache, dan pengaturan redis-cli pada instans EC2 Anda. Untuk informasi selengkapnya mengenai langkah tersebut lihat Pengaturan.

Menemukan titik akhir cache Anda

AWS Management Console

Untuk menemukan titik akhir cache Anda menggunakan konsol ElastiCache:

1. Masuk ke AWS Management Console dan buka konsol Amazon ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Di panel navigasi pada bagian kiri konsol. pilih Cache Redis.
3. Di sisi kanan konsol, klik nama cache yang baru saja Anda buat.
4. Dalam detail Cache, cari dan salin titik akhir cache.

AWS CLI

Contoh AWS CLI berikut menunjukkan cara menemukan titik akhir untuk cache baru Anda menggunakan perintah describe-serverless-caches. Setelah Anda menjalankan perintah, cari kolom “Titik Akhir”.

Linux

```
aws elasticache describe-serverless-caches \
  --serverless-cache-name CacheName
```

Windows

```
aws elasticache describe-serverless-caches ^
  --serverless-cache-name CacheName
```

### Menyambungkan ke Cache Redis Anda (Linux)

Setelah memiliki titik akhir yang dibutuhkan, Anda dapat masuk ke instans EC2 dan menyambungkan ke cache. Pada contoh berikut, Anda menggunakan utilitas redis-cli untuk menyambung ke klaster. Perintah berikut menyambungkan ke cache (catatan: ganti cache-endpoint dengan titik akhir yang Anda ambil pada langkah sebelumnya).

```
src/redis-cli -h cache-endpoint --tls -p 6379
set a "hello"          // Set key "a" with a string value and no expiration
OK
get a                  // Get value for key "a"
"hello"
```

### Menyambungkan ke Cache Redis Anda (Windows)

Setelah memiliki titik akhir yang dibutuhkan, Anda dapat masuk ke instans EC2 dan menyambungkan ke cache. Pada contoh berikut, Anda menggunakan utilitas redis-cli untuk menyambung ke klaster. Perintah berikut menyambung ke cache. Buka Prompt Perintah dan ubah ke direktori Redis dan jalankan perintah (catatan: ganti Cache_Endpoint dengan titik akhir yang Anda ambil pada langkah sebelumnya).

```
c:\Redis>redis-cli -h Redis_Cluster_Endpoint --tls -p 6379
set a "hello"          // Set key "a" with a string value and no expiration
OK
get a                  // Get value for key "a"
"hello"
```

Sekarang Anda dapat melanjutkan ke Langkah 3: (Opsional) Hapus.

## Langkah 3: (Opsional) Hapus

Jika Anda tidak lagi memerlukan cache Amazon ElastiCache yang telah Anda buat, Anda dapat menghapusnya. Langkah ini membantu memastikan bahwa Anda tidak akan dikenakan biaya untuk sumber daya yang tidak Anda gunakan. Anda dapat menggunakan konsol ElastiCache, AWS CLI, atau API ElastiCache untuk menghapus cache Anda.

AWS Management Console

Untuk menghapus cache Anda menggunakan konsol:

1. Masuk ke AWS Management Console dan buka konsol Amazon ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Di panel navigasi pada bagian kiri konsol. pilih Cache Redis.
3. Pilih tombol radio di samping cache yang ingin Anda hapus.
4. Pilih Tindakan di kanan atas lalu pilih Hapus.
5. Anda dapat memilih untuk mengambil snapshot terakhir sebelum menghapus cache Anda.
6. Pada layar konfirmasi Hapus, masukkan kembali nama cache dan pilih Hapus untuk menghapus klaster, atau pilih Batal untuk mempertahankan klaster.

Setelah cache Anda masuk ke status PENGHAPUSAN, Anda tidak lagi dikenakan biaya untuk itu.

AWS CLI

Contoh AWS CLI berikut menghapus cache menggunakan perintah delete-serverless-cache.

Linux

```
aws elasticache delete-serverless-cache \
  --serverless-cache-name CacheName
```

Windows

```
aws elasticache delete-serverless-cache ^
  --serverless-cache-name CacheName
```

Perhatikan bahwa nilai bidang Status diatur ke DELETING.

Sekarang Anda dapat melanjutkan ke Langkah Berikutnya.

# Langkah Berikutnya

Untuk informasi selengkapnya ElastiCache lihat halaman berikut:

- Bekerja dengan ElastiCache

- Penskalaan ElastiCache untuk Redis
- Logging dan pemantauan di Amazon ElastiCache
- Praktik terbaik ElastiCache dan strategi caching
- Melakukan snapshot dan pemulihan
- Pemantauan Amazon SNS dari peristiwa ElastiCache

# Mulai Menggunakan ElastiCache dan SDK AWS

Bagian ini berisi tutorial langsung untuk membantu Anda mempelajari Amazon ElastiCache. Kami mendorong Anda untuk mengikuti salah satu tutorial dengan bahasa yang sesuai.

> **Note**
>
> SDK AWS tersedia dalam berbagai bahasa. Untuk daftar yang lengkapnya, lihat Alat untuk Amazon Web Services.

## Python dan ElastiCache

Dalam tutorial ini, Anda menggunakan AWS SDK for Python (Boto3) untuk menulis program sederhana untuk melakukan operasi ElastiCache berikut:

- Membuat klaster ElastiCache (mode klaster diaktifkan dan mode klaster dinonaktifkan)
- Periksa apakah pengguna atau grup pengguna sudah ada, jika tidak ada, buat pengguna dan grup pengguna (hanya Redis 6.0 dan yang lebih baru)
- Menghubungkan ke ElastiCache
- Lakukan operasi seperti mengatur dan mendapatkan string, membaca dari dan menulis ke aliran, serta menerbitkan dan berlangganan dari saluran Pub/Sub.

Dalam menggunakan tutorial ini, Anda dapat membaca dokumentasi AWS SDK for Python (Boto). Bagian berikut ini khusus untuk ElastiCache: klien tingkat rendah ElastiCache

### Prasyarat Tutorial

- Menyiapkan kunci akses AWS untuk menggunakan SDK AWS. Untuk informasi selengkapnya, lihat Pengaturan.

- Instal Python 3.0 atau yang lebih baru. Untuk informasi selengkapnya, lihat https://www.python.org/downloads. Untuk petunjuk, lihat Quickstart dalam dokumentasi Boto 3.

## Membuat ElastiCache cluster dan pengguna

Contoh berikut menggunakan boto3 SDK untuk operasi ElastiCache manajemen (kluster atau pembuatan pengguna) dan redis-py-cluster redis-py/ untuk penanganan data.

Topik

- Buat klaster dengan mode klaster dinonaktifkan
- Membuat klaster dengan mode klaster dinonaktifkan dengan TLS dan RBAC
- Membuat klaster dengan mode klaster diaktifkan
- Membuat klaster dengan mode klaster diaktifkan dengan TLS dan RBAC
- Periksa apakah pengguna/grup pengguna sudah ada, buat jika belum ada

### Buat klaster dengan mode klaster dinonaktifkan

Salin program berikut dan tempel ke dalam file bernama CreateClusterModeDisabledCluster.py.

```
import boto3
import logging

logging.basicConfig(level=logging.INFO)
client = boto3.client('elasticache')

def
 create_cluster_mode_disabled(CacheNodeType='cache.t3.small',EngineVersion='6.0',NumCacheCluste
 cache cluster',ReplicationGroupId=None):
    """Creates an ElastiCache Cluster with cluster mode disabled

    Returns a dictionary with the API response

    :param CacheNodeType: Node type used on the cluster. If not specified,
 cache.t3.small will be used
    Refer to https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/
CacheNodes.SupportedTypes.html for supported node types
    :param EngineVersion: Engine version to be used. If not specified, latest will be
 used.
```

```
    :param NumCacheClusters: Number of nodes in the cluster. Minimum 1 (just a primary
 node) and maximun 6 (1 primary and 5 replicas).
    If not specified, cluster will be created with 1 primary and 1 replica.
    :param ReplicationGroupDescription: Description for the cluster.
    :param ReplicationGroupId: Name for the cluster
    :return: dictionary with the API results

    """
    if not ReplicationGroupId:
        return 'ReplicationGroupId parameter is required'

    response = client.create_replication_group(
        AutomaticFailoverEnabled=True,
        CacheNodeType=CacheNodeType,
        Engine='redis',
        EngineVersion=EngineVersion,
        NumCacheClusters=NumCacheClusters,
        ReplicationGroupDescription=ReplicationGroupDescription,
        ReplicationGroupId=ReplicationGroupId,
        SnapshotRetentionLimit=30,
    )
    return response


if __name__ == '__main__':

    # Creates an ElastiCache Cluster mode disabled cluster, based on cache.m6g.large
 nodes, Redis 6, one primary and two replicas
    elasticacheResponse = create_cluster_mode_disabled(
        #CacheNodeType='cache.m6g.large',
        EngineVersion='6.0',
        NumCacheClusters=3,
        ReplicationGroupDescription='Redis cluster mode disabled with replicas',
        ReplicationGroupId='redis202104053'
        )

    logging.info(elasticacheResponse)
```

Untuk menjalankan program ini, masukkan perintah berikut:

```
python CreateClusterModeDisabledCluster.py
```

Untuk informasi selengkapnya, lihat Mengelola klaster.

### Membuat klaster dengan mode klaster dinonaktifkan dengan TLS dan RBAC

Untuk memastikan keamanan, Anda dapat menggunakan Keamanan Lapisan Pengangkutan (TLS) dan Kontrol Akses Berbasis Peran (RBAC) saat membuat klaster dengan mode klaster dinonaktifkan. Tidak seperti AUTH Redis yang memungkinkan semua klien terautentikasi memiliki akses grup replikasi penuh jika tokennya terautentikasi, RBAC memungkinkan Anda mengontrol akses klaster melalui grup pengguna. Grup pengguna ini dirancang sebagai cara untuk mengatur akses ke grup replikasi. Untuk informasi selengkapnya, lihat Kontrol Akses Berbasis Peran (RBAC).

Salin program berikut dan tempel ke file bernama ClusterModeDisabledWithRBAC.py.

```
import boto3
import logging

logging.basicConfig(level=logging.INFO)
client = boto3.client('elasticache')

def
 create_cluster_mode_disabled_rbac(CacheNodeType='cache.t3.small',EngineVersion='6.0',NumCacheC
 cache cluster',ReplicationGroupId=None, UserGroupIds=None,
 SecurityGroupIds=None,CacheSubnetGroupName=None):
    """Creates an ElastiCache Cluster with cluster mode disabled and RBAC

    Returns a dictionary with the API response

    :param CacheNodeType: Node type used on the cluster. If not specified,
 cache.t3.small will be used
    Refer to https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/
CacheNodes.SupportedTypes.html for supported node types
    :param EngineVersion: Engine version to be used. If not specified, latest will be
 used.
    :param NumCacheClusters: Number of nodes in the cluster. Minimum 1 (just a primary
 node) and maximun 6 (1 primary and 5 replicas).
    If not specified, cluster will be created with 1 primary and 1 replica.
    :param ReplicationGroupDescription: Description for the cluster.
    :param ReplicationGroupId: Mandatory name for the cluster.
    :param UserGroupIds: The ID of the user group to be assigned to the cluster.
    :param SecurityGroupIds: List of security groups to be assigned. If not defined,
 default will be used
    :param CacheSubnetGroupName: subnet group where the cluster will be placed. If not
 defined, default will be used.
    :return: dictionary with the API results
```

```
    """
    if not ReplicationGroupId:
        return {'Error': 'ReplicationGroupId parameter is required'}
    elif not isinstance(UserGroupIds,(list)):
        return {'Error': 'UserGroupIds parameter is required and must be a list'}

    params={'AutomaticFailoverEnabled': True,
            'CacheNodeType': CacheNodeType,
            'Engine': 'redis',
            'EngineVersion': EngineVersion,
            'NumCacheClusters': NumCacheClusters,
            'ReplicationGroupDescription': ReplicationGroupDescription,
            'ReplicationGroupId': ReplicationGroupId,
            'SnapshotRetentionLimit': 30,
            'TransitEncryptionEnabled': True,
            'UserGroupIds':UserGroupIds
        }

    # defaults will be used if CacheSubnetGroupName or SecurityGroups are not explicit.
    if isinstance(SecurityGroupIds,(list)):
        params.update({'SecurityGroupIds':SecurityGroupIds})
    if CacheSubnetGroupName:
        params.update({'CacheSubnetGroupName':CacheSubnetGroupName})

    response = client.create_replication_group(**params)
    return response

if __name__ == '__main__':

    # Creates an ElastiCache Cluster mode disabled cluster, based on cache.m6g.large
 nodes, Redis 6, one primary and two replicas.
    # Assigns the existent user group "mygroup" for RBAC authentication

    response=create_cluster_mode_disabled_rbac(
        CacheNodeType='cache.m6g.large',
        EngineVersion='6.0',
        NumCacheClusters=3,
        ReplicationGroupDescription='Redis cluster mode disabled with replicas',
        ReplicationGroupId='redis202104',
        UserGroupIds=[
            'mygroup'
        ],
        SecurityGroupIds=[
            'sg-7cc73803'
```

```
        ],
        CacheSubnetGroupName='default'
    )

    logging.info(response)
```

Untuk menjalankan program ini, masukkan perintah berikut:

```
python ClusterModeDisabledWithRBAC.py
```

Untuk informasi selengkapnya, lihat Mengelola klaster.

Membuat klaster dengan mode klaster diaktifkan

Salin program berikut dan tempel ke dalam file bernama ClusterModeEnabled.py.

```
import boto3
import logging

logging.basicConfig(level=logging.INFO)
client = boto3.client('elasticache')

def
 create_cluster_mode_enabled(CacheNodeType='cache.t3.small',EngineVersion='6.0',NumNodeGroups=1
 ReplicationGroupDescription='Sample cache with cluster mode
 enabled',ReplicationGroupId=None):
    """Creates an ElastiCache Cluster with cluster mode enabled

    Returns a dictionary with the API response

    :param CacheNodeType: Node type used on the cluster. If not specified,
 cache.t3.small will be used
    Refer to https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/
CacheNodes.SupportedTypes.html for supported node types
    :param EngineVersion: Engine version to be used. If not specified, latest will be
 used.
    :param NumNodeGroups: Number of shards in the cluster. Minimum 1 and maximun 90.
    If not specified, cluster will be created with 1 shard.
    :param ReplicasPerNodeGroup: Number of replicas per shard. If not specified 1
 replica per shard will be created.
    :param ReplicationGroupDescription: Description for the cluster.
    :param ReplicationGroupId: Name for the cluster
    :return: dictionary with the API results
```

```
    """
    if not ReplicationGroupId:
        return 'ReplicationGroupId parameter is required'

    response = client.create_replication_group(
        AutomaticFailoverEnabled=True,
        CacheNodeType=CacheNodeType,
        Engine='redis',
        EngineVersion=EngineVersion,
        ReplicationGroupDescription=ReplicationGroupDescription,
        ReplicationGroupId=ReplicationGroupId,
    #   Creates a cluster mode enabled cluster with 1 shard(NumNodeGroups), 1 primary
 node (implicit) and 2 replicas (replicasPerNodeGroup)
        NumNodeGroups=NumNodeGroups,
        ReplicasPerNodeGroup=ReplicasPerNodeGroup,
        CacheParameterGroupName='default.redis6.0.cluster.on'
    )

    return response


# Creates a cluster mode enabled
response = create_cluster_mode_enabled(
    CacheNodeType='cache.m6g.large',
    EngineVersion='6.0',
    ReplicationGroupDescription='Redis cluster mode enabled with replicas',
    ReplicationGroupId='redis20210',
#   Creates a cluster mode enabled cluster with 1 shard(NumNodeGroups), 1 primary
 (implicit) and 2 replicas (replicasPerNodeGroup)
    NumNodeGroups=2,
    ReplicasPerNodeGroup=1,
)

logging.info(response)
```

Untuk menjalankan program ini, masukkan perintah berikut:

```
python ClusterModeEnabled.py
```

Untuk informasi selengkapnya, lihat Mengelola klaster.

Membuat klaster dengan mode klaster diaktifkan dengan TLS dan RBAC

Untuk memastikan keamanan, Anda dapat menggunakan Keamanan Lapisan Pengangkutan (TLS) dan Kontrol Akses Berbasis Peran (RBAC) saat membuat klaster dengan mode klaster diaktifkan. Tidak seperti AUTH Redis yang memungkinkan semua klien terautentikasi memiliki akses grup replikasi penuh jika tokennya terautentikasi, RBAC memungkinkan Anda mengontrol akses klaster melalui grup pengguna. Grup pengguna ini dirancang sebagai cara untuk mengatur akses ke grup replikasi. Untuk informasi selengkapnya, lihat Kontrol Akses Berbasis Peran (RBAC).

Salin program berikut dan tempel ke file bernama ClusterModeEnabledWithRBAC.py.

```
import boto3
import logging

logging.basicConfig(level=logging.INFO)
client = boto3.client('elasticache')

def
 create_cluster_mode_enabled(CacheNodeType='cache.t3.small',EngineVersion='6.0',NumNodeGroups=1
 ReplicationGroupDescription='Sample cache with cluster
 mode enabled',ReplicationGroupId=None,UserGroupIds=None,
 SecurityGroupIds=None,CacheSubnetGroupName=None,CacheParameterGroupName='default.redis6.0.clus
    """Creates an ElastiCache Cluster with cluster mode enabled and RBAC

    Returns a dictionary with the API response

    :param CacheNodeType: Node type used on the cluster. If not specified,
 cache.t3.small will be used
    Refer to https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/
CacheNodes.SupportedTypes.html for supported node types
    :param EngineVersion: Engine version to be used. If not specified, latest will be
 used.
    :param NumNodeGroups: Number of shards in the cluster. Minimum 1 and maximun 90.
    If not specified, cluster will be created with 1 shard.
    :param ReplicasPerNodeGroup: Number of replicas per shard. If not specified 1
 replica per shard will be created.
    :param ReplicationGroupDescription: Description for the cluster.
    :param ReplicationGroupId: Name for the cluster.
    :param CacheParameterGroupName: Parameter group to be used. Must be compatible with
 the engine version and cluster mode enabled.
    :return: dictionary with the API results

    """
```

```
    if not ReplicationGroupId:
        return 'ReplicationGroupId parameter is required'
    elif not isinstance(UserGroupIds,(list)):
        return {'Error': 'UserGroupIds parameter is required and must be a list'}

    params={'AutomaticFailoverEnabled': True,
            'CacheNodeType': CacheNodeType,
            'Engine': 'redis',
            'EngineVersion': EngineVersion,
            'ReplicationGroupDescription': ReplicationGroupDescription,
            'ReplicationGroupId': ReplicationGroupId,
            'SnapshotRetentionLimit': 30,
            'TransitEncryptionEnabled': True,
            'UserGroupIds':UserGroupIds,
            'NumNodeGroups': NumNodeGroups,
            'ReplicasPerNodeGroup': ReplicasPerNodeGroup,
            'CacheParameterGroupName': CacheParameterGroupName
        }

    # defaults will be used if CacheSubnetGroupName or SecurityGroups are not explicit.
    if isinstance(SecurityGroupIds,(list)):
        params.update({'SecurityGroupIds':SecurityGroupIds})
    if CacheSubnetGroupName:
        params.update({'CacheSubnetGroupName':CacheSubnetGroupName})

    response = client.create_replication_group(**params)
    return response

if __name__ == '__main__':
    # Creates a cluster mode enabled cluster
    response = create_cluster_mode_enabled(
        CacheNodeType='cache.m6g.large',
        EngineVersion='6.0',
        ReplicationGroupDescription='Redis cluster mode enabled with replicas',
        ReplicationGroupId='redis2021',
    #   Creates a cluster mode enabled cluster with 1 shard(NumNodeGroups), 1 primary
 (implicit) and 2 replicas (replicasPerNodeGroup)
        NumNodeGroups=2,
        ReplicasPerNodeGroup=1,
        UserGroupIds=[
            'mygroup'
        ],
        SecurityGroupIds=[
            'sg-7cc73803'
```

```
        ],
        CacheSubnetGroupName='default'

    )

    logging.info(response)
```

Untuk menjalankan program ini, masukkan perintah berikut:

```
python ClusterModeEnabledWithRBAC.py
```

Untuk informasi selengkapnya, lihat Mengelola klaster.

Periksa apakah pengguna/grup pengguna sudah ada, buat jika belum ada

Dengan RBAC, Anda membuat pengguna dan memberi mereka izin tertentu menggunakan string akses. Anda menetapkan pengguna ke grup pengguna yang selaras dengan peran tertentu (administrator, sumber daya manusia) yang kemudian disebarkan ke satu atau beberapa grup replikasi Redis ElastiCache . Dengan melakukan hal ini, Anda dapat menetapkan batas keamanan antara klien menggunakan grup replikasi atau grup Redis yang sama dan mencegah klien mengakses data masing-masing. Untuk informasi selengkapnya, lihat Kontrol Akses Berbasis Peran (RBAC).

Salin program berikut dan tempel ke dalam file bernama UserAndUserGroups.py. Perbarui mekanisme untuk memberikan kredensi. Kredensial dalam contoh ini ditampilkan sebagai dapat diganti dan diberikan item yang tidak dideklarasikan. Hindari kredensial hard-coding.

```
import boto3
import logging

logging.basicConfig(level=logging.INFO)
client = boto3.client('elasticache')

def check_user_exists(UserId):
    """Checks if UserId exists

    Returns True if UserId exists, otherwise False
    :param UserId: ElastiCache User ID
    :return: True|False
    """
    try:
        response = client.describe_users(
            UserId=UserId,
```

```
        )
        if response['Users'][0]['UserId'].lower() == UserId.lower():
            return True
    except Exception as e:
        if e.response['Error']['Code'] == 'UserNotFound':
            logging.info(e.response['Error'])
            return False
        else:
            raise

def check_group_exists(UserGroupId):
    """Checks if UserGroupID exists

    Returns True if Group ID exists, otherwise False
    :param UserGroupId: ElastiCache User ID
    :return: True|False
    """

    try:
        response = client.describe_user_groups(
            UserGroupId=UserGroupId
        )
        if response['UserGroups'][0]['UserGroupId'].lower() == UserGroupId.lower():
            return True
    except Exception as e:
        if e.response['Error']['Code'] == 'UserGroupNotFound':
            logging.info(e.response['Error'])
            return False
        else:
            raise

def create_user(UserId=None,UserName=None,Password=None,AccessString=None):
    """Creates a new user

    Returns the ARN for the newly created user or the error message
    :param UserId: ElastiCache user ID. User IDs must be unique
    :param UserName: ElastiCache user name. ElastiCache allows multiple users with the
 same name as long as the associated user ID is unique.
    :param Password: Password for user. Must have at least 16 chars.
    :param AccessString: Access string with the permissions for the user. For
 details refer to https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/
Clusters.RBAC.html#Access-string
    :return: user ARN
    """
```

```
    try:
        response = client.create_user(
            UserId=UserId,
            UserName=UserName,
            Engine='Redis',
            Passwords=[Password],
            AccessString=AccessString,
            NoPasswordRequired=False
        )
        return response['ARN']
    except Exception as e:
        logging.info(e.response['Error'])
        return e.response['Error']

def create_group(UserGroupId=None, UserIds=None):
    """Creates a new group.
    A default user is required (mandatory) and should be specified in the UserIds list

    Return: Group ARN
    :param UserIds: List with user IDs to be associated with the new group. A default
 user is required
    :param UserGroupId: The ID (name) for the group
    :return: Group ARN
    """
    try:
        response = client.create_user_group(
            UserGroupId=UserGroupId,
            Engine='Redis',
            UserIds=UserIds
        )
        return response['ARN']
    except Exception as e:
        logging.info(e.response['Error'])


if __name__ == '__main__':

    groupName='mygroup2'
    userName = 'myuser2'
    userId=groupName+'-'+userName

    # Creates a new user if the user ID does not exist.
    for tmpUserId,tmpUserName in [ (userId,userName), (groupName+'-
default','default')]:
```

```
        if not check_user_exists(tmpUserId):
            response=create_user(UserId=tmpUserId,
 UserName=EXAMPLE,Password=EXAMPLE,AccessString='on ~* +@all')
            logging.info(response)
        # assigns the new user ID to the user group
    if not check_group_exists(groupName):
        UserIds = [ userId , groupName+'-default']
        response=create_group(UserGroupId=groupName,UserIds=UserIds)
        logging.info(response)
```

Untuk menjalankan program ini, masukkan perintah berikut:

```
python UserAndUserGroups.py
```

## Menghubungkan ke ElastiCache

Contoh berikut menggunakan klien Redis untuk terhubung ke ElastiCache.

Topik

- Menghubungkan ke klaster yang menonaktifkan mode klaster
- Menghubungkan ke klaster yang mengaktifkan mode klaster

### Menghubungkan ke klaster yang menonaktifkan mode klaster

Salin program berikut dan tempelkan ke file bernama ConnectClusterModeDisabled.py. Perbarui mekanisme untuk memberikan kredensi. Kredensial dalam contoh ini ditampilkan sebagai dapat diganti dan diberikan item yang tidak dideklarasikan. Hindari kredensial hard-coding.

```
from redis import Redis
import logging

logging.basicConfig(level=logging.INFO)
redis = Redis(host='primary.xxx.yyyyyy.zzz1.cache.amazonaws.com', port=6379,
 decode_responses=True, ssl=True, username=example, password=EXAMPLE)

if redis.ping():
    logging.info("Connected to Redis")
```

Untuk menjalankan program ini, masukkan perintah berikut:

```
python ConnectClusterModeDisabled.py
```

Menghubungkan ke klaster yang mengaktifkan mode klaster

Salin program berikut dan tempelkan ke file bernama ConnectClusterModeEnabled.py.

```
from rediscluster import RedisCluster
import logging

logging.basicConfig(level=logging.INFO)
redis = RedisCluster(startup_nodes=[{"host":
 "xxx.yyy.clustercfg.zzz1.cache.amazonaws.com","port": "6379"}],
 decode_responses=True,skip_full_coverage_check=True)

if redis.ping():
    logging.info("Connected to Redis")
```

Untuk menjalankan program ini, masukkan perintah berikut:

```
python ConnectClusterModeEnabled.py
```

## Contoh penggunaan

Contoh berikut menggunakan SDK boto3 untuk ElastiCache bekerja dengan ElastiCache.

Topik

- Menetapkan dan Mendapatkan string
- Tentukan dan Dapatkan hash dengan beberapa item
- Publikasikan (tulis) dan berlangganan (baca) dari saluran Pub/Sub
- Tulis dan baca dari aliran

### Menetapkan dan Mendapatkan string

Salin program berikut dan tempelkan ke file bernama SetAndGetStrings.py.

```
import time
import logging
logging.basicConfig(level=logging.INFO,format='%(asctime)s: %(message)s')

keyName='mykey'
currTime=time.ctime(time.time())
```

```
# Set the key 'mykey' with the current date and time as value.
# The Key will expire and removed from cache in 60 seconds.
redis.set(keyName, currTime, ex=60)

# Sleep just for better illustration of TTL (expiration) value
time.sleep(5)

# Retrieve the key value and current TTL
keyValue=redis.get(keyName)
keyTTL=redis.ttl(keyName)

logging.info("Key {} was set at {} and has {} seconds until expired".format(keyName,
 keyValue, keyTTL))
```

Untuk menjalankan program ini, masukkan perintah berikut:

```
python SetAndGetStrings.py
```

Tentukan dan Dapatkan hash dengan beberapa item

Salin program berikut dan tempelkan ke file bernama SetAndGetHash.py.

```
import logging
import time

logging.basicConfig(level=logging.INFO,format='%(asctime)s: %(message)s')

keyName='mykey'
keyValues={'datetime': time.ctime(time.time()), 'epochtime': time.time()}

# Set the hash 'mykey' with the current date and time in human readable format
 (datetime field) and epoch number (epochtime field).
redis.hset(keyName, mapping=keyValues)

# Set the key to expire and removed from cache in 60 seconds.
redis.expire(keyName, 60)

# Sleep just for better illustration of TTL (expiration) value
time.sleep(5)

# Retrieves all the fields and current TTL
keyValues=redis.hgetall(keyName)
keyTTL=redis.ttl(keyName)
```

```
logging.info("Key {} was set at {} and has {} seconds until expired".format(keyName,
 keyValues, keyTTL))
```

Untuk menjalankan program ini, masukkan perintah berikut:

```
python SetAndGetHash.py
```

Publikasikan (tulis) dan berlangganan (baca) dari saluran Pub/Sub

Salin program berikut dan tempelkan ke file bernama PubAndSub.py.

```
import logging
import time

def handlerFunction(message):
    """Prints message got from PubSub channel to the log output

    Return None
    :param message: message to log
    """
    logging.info(message)

logging.basicConfig(level=logging.INFO)
redis = Redis(host="redis202104053.tihewd.ng.0001.use1.cache.amazonaws.com", port=6379,
 decode_responses=True)


# Creates the subscriber connection on "mychannel"
subscriber = redis.pubsub()
subscriber.subscribe(**{'mychannel': handlerFunction})

# Creates a new thread to watch for messages while the main process continues with its
 routines
thread = subscriber.run_in_thread(sleep_time=0.01)

# Creates publisher connection on "mychannel"
redis.publish('mychannel', 'My message')

# Publishes several messages. Subscriber thread will read and print on log.
while True:
    redis.publish('mychannel',time.ctime(time.time()))
    time.sleep(1)
```

Untuk menjalankan program ini, masukkan perintah berikut:

```
python PubAndSub.py
```

Tulis dan baca dari aliran

Salin program berikut dan tempelkan ke file bernama ReadWriteStream.py.

```
from redis import Redis
import redis.exceptions as exceptions
import logging
import time
import threading

logging.basicConfig(level=logging.INFO)

def writeMessage(streamName):
    """Starts a loop writting the current time and thread name to 'streamName'

    :param streamName: Stream (key) name to write messages.
    """
    fieldsDict={'writerId':threading.currentThread().getName(),'myvalue':None}
    while True:
        fieldsDict['myvalue'] = time.ctime(time.time())
        redis.xadd(streamName,fieldsDict)
        time.sleep(1)

def readMessage(groupName=None,streamName=None):
    """Starts a loop reading from 'streamName'
    Multiple threads will read from the same stream consumer group. Consumer group is
 used to coordinate data distribution.
    Once a thread acknowleges the message, it won't be provided again. If message
 wasn't acknowledged, it can be served to another thread.

    :param groupName: stream group were multiple threads will read.
    :param streamName: Stream (key) name where messages will be read.
    """

    readerID=threading.currentThread().getName()
    while True:
        try:
            # Check if the stream has any message
            if redis.xlen(streamName)>0:
```

```
                # Check if if the messages are new (not acknowledged) or not (already
 processed)
                streamData=redis.xreadgroup(groupName,readerID,
{streamName:'>'},count=1)
                if len(streamData) > 0:
                    msgId,message = streamData[0][1][0]
                    logging.info("{}: Got {} from ID
 {}".format(readerID,message,msgId))
                    #Do some processing here. If the message has been processed
 sucessfuly, acknowledge it and (optional) delete the message.
                    redis.xack(streamName,groupName,msgId)
                    logging.info("Stream message ID {} read and processed successfuly
 by {}".format(msgId,readerID))
                    redis.xdel(streamName,msgId)
            else:
                pass
        except:
            raise

        time.sleep(0.5)

# Creates the stream 'mystream' and consumer group 'myworkergroup' where multiple
 threads will write/read.
try:
    redis.xgroup_create('mystream','myworkergroup',mkstream=True)
except exceptions.ResponseError as e:
    logging.info("Consumer group already exists. Will continue despite the error:
 {}".format(e))
except:
    raise

# Starts 5 writer threads.
for writer_no in range(5):
    writerThread = threading.Thread(target=writeMessage, name='writer-'+str(writer_no),
 args=('mystream',),daemon=True)
    writerThread.start()

# Starts 10 reader threads
for reader_no in range(10):
    readerThread = threading.Thread(target=readMessage, name='reader-'+str(reader_no),
 args=('myworkergroup','mystream',),daemon=True)
    readerThread.daemon = True
    readerThread.start()
```

```
# Keep the code running for 30 seconds
time.sleep(30)
```

Untuk menjalankan program ini, masukkan perintah berikut:

```
python ReadWriteStream.py
```

# Tutorial: Mengonfigurasi fungsi Lambda untuk mengakses Amazon di ElastiCache VPC Amazon

Dalam tutorial ini Anda dapat mempelajari cara membuat cache ElastiCache tanpa server, membuat fungsi Lambda, lalu menguji fungsi Lambda, dan secara opsional membersihkan setelahnya.

Topik



## Langkah 1: Buat cache tanpa server

Untuk membuat cache tanpa server, ikuti langkah-langkah ini.

Topik



### Langkah 1.1: Buat cache tanpa server

Pada langkah ini, Anda membuat cache tanpa server di VPC Amazon default di wilayah us-east-1 di akun Anda menggunakan (CLI). AWS Command Line Interface Untuk informasi tentang membuat cache tanpa server menggunakan ElastiCache konsol atau API, lihat. Langkah 1: Buat Cache

```
aws elasticache create-serverless-cache \
  --serverless-cache-name cache-01  \
--description "ElastiCache IAM auth application" \
--engine redis
```

Perhatikan bahwa nilai bidang Status diatur ke `CREATING`. Diperlukan waktu satu menit ElastiCache untuk menyelesaikan pembuatan cache Anda.

### Langkah 1.2: Salin titik akhir cache tanpa server

Verifikasi bahwa ElastiCache untuk Redis telah selesai membuat cache dengan `describe-serverless-caches` perintah.

```
aws elasticache describe-serverless-caches \
--serverless-cache-name cache-01
```

Salin Alamat Titik Akhir yang ditunjukkan pada output. Anda akan memerlukan alamat ini saat membuat paket deployment untuk fungsi Lambda Anda.

### Langkah 1.3: Buat Peran IAM

1. Buat dokumen kebijakan kepercayaan IAM, seperti yang ditunjukkan di bawah ini, untuk peran Anda yang memungkinkan akun Anda mengambil peran baru. Simpan kebijakan ini ke file bernama trust-policy.json.

   ```
   {
   "Version": "2012-10-17",
       "Statement": [{
   "Effect": "Allow",
           "Principal": { "AWS": "arn:aws:iam::123456789012:root" },
           "Action": "sts:AssumeRole"
       },
       {
         "Effect": "Allow",
         "Principal": {
           "Service": "lambda.amazonaws.com"
         },
         "Action": "sts:AssumeRole"
       }]
   }
   ```

2. Buat dokumen kebijakan IAM, seperti yang ditunjukkan di bawah ini. Simpan kebijakan ke file bernama policy.json.

```
{
"Version": "2012-10-17",
  "Statement": [
    {
"Effect" : "Allow",
      "Action" : [
        "elasticache:Connect"
      ],
      "Resource" : [
        "arn:aws:elasticache:us-east-1:123456789012:serverlesscache:cache-01",
        "arn:aws:elasticache:us-east-1:123456789012:user:iam-user-01"
      ]
    }
  ]
}
```

3. Buat peran IAM.

```
aws iam create-role \
--role-name "elasticache-iam-auth-app" \
--assume-role-policy-document file://trust-policy.json
```

4. Buat kebijakan IAM.

```
aws iam create-policy \
  --policy-name "elasticache-allow-all" \
  --policy-document file://policy.json
```

5. Lampirkan kebijakan IAM di peran tersebut.

```
aws iam attach-role-policy \
 --role-name "elasticache-iam-auth-app" \
 --policy-arn "arn:aws:iam::123456789012:policy/elasticache-allow-all"
```

## Langkah 1.4: Buat cache tanpa server

1. Buat pengguna default baru.

```
aws elasticache create-user \
  --user-name default \
--user-id default-user-disabled \
--engine redis \
--authentication-mode Type=no-password-required \
--access-string "off +get ~keys*"
```

2. Buat pengguna baru yang mendukung IAM.

```
aws elasticache create-user \
  --user-name iam-user-01 \
--user-id iam-user-01 \
--authentication-mode Type=iam \
--engine redis \
--access-string "on ~* +@all"
```

3. Buat grup pengguna dan lampirkan pengguna.

```
aws elasticache create-user-group \
  --user-group-id iam-user-group-01 \
--engine redis \
--user-ids default-user-disabled iam-user-01

aws elasticache modify-serverless-cache \
  --serverless-cache-name cache-01  \
--user-group-id iam-user-group-01
```

## Langkah 2: Buat fungsi Lambda

Untuk membuat fungsi Lambda, lakukan langkah-langkah ini.

Topik

## Langkah 2.1: Buat fungsi Lambda

Dalam tutorial ini, kami memberikan contoh kode dalam Python untuk fungsi Lambda Anda.

Python

Contoh kode Python berikut membaca dan menulis item ke cache Anda ElastiCache .
Salin kode tersebut dan simpan ke dalam file bernama `app.py`. Pastikan untuk mengganti `elasticache_endpoint` nilai dalam kode dengan alamat titik akhir yang Anda salin pada langkah 1.2.

```
from typing import Tuple, Union
from urllib.parse import ParseResult, urlencode, urlunparse

import botocore.session
import redis
from botocore.model import ServiceId
from botocore.signers import RequestSigner
from cachetools import TTLCache, cached
import uuid

class ElastiCacheIAMProvider(redis.CredentialProvider):
    def __init__(self, user, cache_name, is_serverless=False, region="us-east-1"):
        self.user = user
        self.cache_name = cache_name
        self.is_serverless = is_serverless
        self.region = region

        session = botocore.session.get_session()
        self.request_signer = RequestSigner(
            ServiceId("elasticache"),
            self.region,
            "elasticache",
            "v4",
            session.get_credentials(),
            session.get_component("event_emitter"),
        )

    # Generated IAM tokens are valid for 15 minutes
    @cached(cache=TTLCache(maxsize=128, ttl=900))
    def get_credentials(self) -> Union[Tuple[str], Tuple[str, str]]:
        query_params = {"Action": "connect", "User": self.user}
        if self.is_serverless:
```

```
            query_params["ResourceType"] = "ServerlessCache"
        url = urlunparse(
            ParseResult(
                scheme="https",
                netloc=self.cache_name,
                path="/",
                query=urlencode(query_params),
                params="",
                fragment="",
            )
        )
        signed_url = self.request_signer.generate_presigned_url(
            {"method": "GET", "url": url, "body": {}, "headers": {}, "context": {}},
            operation_name="connect",
            expires_in=900,
            region_name=self.region,
        )
        # RequestSigner only seems to work if the URL has a protocol, but
        # Elasticache only accepts the URL without a protocol
        # So strip it off the signed URL before returning
        return (self.user, signed_url.removeprefix("https://"))

def lambda_handler(event, context):
    username = "iam-user-01" # replace with your user id
    cache_name = "cache-01" # replace with your cache name
    elasticache_endpoint = "cache-01-xxxxx.serverless.use1.cache.amazonaws.com" #
 replace with your cache endpoint
    creds_provider = ElastiCacheIAMProvider(user=username, cache_name=cache_name,
 is_serverless=True)
    redis_client = redis.Redis(host=elasticache_endpoint, port=6379,
 credential_provider=creds_provider, ssl=True, ssl_cert_reqs="none")

    key='uuid'
    # create a random UUID - this will be the sample element we add to the cache
    uuid_in = uuid.uuid4().hex
    redis_client.set(key, uuid_in)
    result = redis_client.get(key)
    decoded_result = result.decode("utf-8")
    # check the retrieved item matches the item added to the cache and print
    # the results
    if decoded_result == uuid_in:
        print(f"Success: Inserted {uuid_in}. Fetched {decoded_result} from Redis.")
    else:
```

```
        raise Exception(f"Bad value retrieved. Expected {uuid_in}, got
 {decoded_result}")

    return "Fetched value from Redis"
```

Kode ini menggunakan pustaka redis-py Python untuk memasukkan item ke dalam cache Anda dan mengambilnya. Kode ini menggunakan cachetools untuk menyimpan token Auth IAM yang dihasilkan selama 15 menit. Untuk membuat paket penerapan yang berisi redis-py dan cachetools, lakukan langkah-langkah berikut.

Di direktori proyek Anda yang berisi file kode sumber app.py, buat paket folder untuk menginstal pustaka redis-py dan cachetools ke dalamnya.

```
mkdir package
```

Instal redis-py, cachetools menggunakan pip.

```
pip install --target ./package redis
pip install --target ./package cachetools
```

Buat file.zip yang berisi pustaka redis-py dan cachetools. Di Linux atau macOS, jalankan perintah CLI berikut. Di Windows, gunakan utilitas zip pilihan Anda untuk membuat file.zip dengan pustaka redis-py dan cachetools di root.

```
cd package
zip -r ../my_deployment_package.zip .
```

Tambahkan kode fungsi Anda ke file .zip. Di Linux atau macOS, jalankan perintah CLI berikut. Di Windows, gunakan utilitas zip pilihan Anda untuk menambahkan app.py ke root file.zip Anda.

```
cd ..
zip my_deployment_package.zip app.py
```

## Langkah 2.2: Buat peran IAM (peran eksekusi)

Lampirkan kebijakan AWS terkelola yang diberi nama `AWSLambdaVPCAccessExecutionRole` ke peran.

```
aws iam attach-role-policy \
```

```
--role-name "elasticache-iam-auth-app" \
--policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaVPCAccessExecutionRole"
```

### Langkah 2.3: Unggah paket deployment (buat fungsi Lambda)

Pada langkah ini, Anda membuat fungsi Lambda (AccessRedis) menggunakan perintah AWS CLI create-function.

Dari direktori proyek yang berisi file paket deployment Anda .zip, jalankan perintah Lambda CLI berikut. `create-function`

Untuk opsi peran, gunakan ARN dari peran eksekusi yang Anda buat di langkah 2.2. Untuk vpc-config masukkan daftar yang dipisahkan koma dari subnet VPC default Anda dan ID grup keamanan VPC default Anda. Anda dapat menemukan nilai-nilai ini di konsol Amazon VPC. Untuk menemukan subnet VPC default Anda, pilih VPC Anda, lalu pilih VPC default akun Anda AWS . Untuk menemukan grup keamanan untuk VPC ini, buka Keamanan dan pilih Grup keamanan. Pastikan Anda memilih wilayah us-east-1.

```
aws lambda create-function \
--function-name AccessRedis  \
--region us-east-1 \
--zip-file fileb://my_deployment_package.zip \
--role arn:aws:iam::123456789012:role/elasticache-iam-auth-app \
--handler app.lambda_handler \
--runtime python3.12  \
--timeout 30 \
--vpc-config SubnetIds=comma-separated-vpc-subnet-ids,SecurityGroupIds=default-
security-group-id
```

## Langkah 3: Uji fungsi Lambda

Pada langkah ini, Anda menjalankan fungsi Lambda secara manual menggunakan perintah pemanggilan. Ketika fungsi Lambda dijalankan, ia menghasilkan UUID dan menulisnya ke ElastiCache cache yang Anda tentukan dalam kode Lambda Anda. Fungsi Lambda kemudian mengambil item dari cache.

1. Memanggil fungsi Lambda AccessRedis () menggunakan perintah pemanggilan AWS Lambda .

   ```
   aws lambda invoke \
   --function-name AccessRedis  \
   ```

```
--region us-east-1 \
output.txt
```

2. Verifikasikan bahwa fungsi Lambda berhasil dijalankan sebagai berikut:

   - Tinjau file output.txt.
   - Verifikasi hasil di CloudWatch Log dengan membuka CloudWatch konsol dan memilih grup log untuk fungsi Anda (AccessRedis/aws/lambda/). Log stream akan berisi output seperti yang berikut ini:

     ```
     Success: Inserted 826e70c5f4d2478c8c18027125a3e01e. Fetched
      826e70c5f4d2478c8c18027125a3e01e from Redis.
     ```

   - Tinjau hasilnya di AWS Lambda konsol.

## Langkah 4: Bersihkan (Opsional)

Untuk membersihkan, ambil langkah-langkah ini.

Topik



### Langkah 4.1: Hapus fungsi Lambda

```
aws lambda delete-function \
 --function-name AccessRedis
```

### Langkah 4.2: Hapus cache Tanpa Server

Hapus cache.

```
aws elasticache delete-serverless-cache \
 --serverless-cache-name cache-01
```

Hapus pengguna dan grup pengguna.

```
aws elasticache delete-user \
 --user-id default-user-disabled

aws elasticache delete-user \
 --user-id iam-user-01

aws elasticache delete-user-group \
 --user-group-id iam-user-group-01
```

## Langkah 4.3: Hapus Peran dan kebijakan IAM

```
aws iam detach-role-policy \
 --role-name "elasticache-iam-auth-app" \
 --policy-arn "arn:aws:iam::123456789012:policy/elasticache-allow-all"

aws iam detach-role-policy \
--role-name "elasticache-iam-auth-app" \
--policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaVPCAccessExecutionRole"

aws iam delete-role \
 --role-name "elasticache-iam-auth-app"

 aws iam delete-policy \
  --policy-arn "arn:aws:iam::123456789012:policy/elasticache-allow-all"
```

# Merancang dan mengelola klaster ElastiCache Anda sendiri

Jika Anda memerlukan kontrol terperinci atas klaster ElastiCache, Anda dapat memilih untuk mendesain klaster Anda sendiri. ElastiCache memungkinkan Anda mengoperasikan klaster berbasis simpul, dengan memilih tipe simpul, jumlah simpul, dan penempatan simpul di seluruh Zona Ketersediaan AWS untuk klaster Anda. Karena ElastiCache adalah layanan yang terkelola sepenuhnya, ElastiCache secara otomatis mengelola penyediaan perangkat keras, pemantauan, penggantian simpul, dan penambalan perangkat lunak untuk klaster Anda.

Untuk informasi tentang cara menyiapkannya, lihat Pengaturan. Untuk detail tentang cara mengelola, memperbarui, atau menghapus simpul atau klaster, lihat Mengelola simpul. Untuk gambaran umum tentang komponen utama deployment Amazon ElastiCache saat Anda mendesain klaster ElastiCache Anda sendiri, lihat konsep utama ini.

Topik



## ElastiCache untuk komponen dan fitur Redis

Berikut ini, Anda dapat menemukan ikhtisar komponen utama ElastiCache penyebaran Amazon.

Topik

- ElastiCache untuk titik akhir Redis
- ElastiCache kelompok parameter
- ElastiCache untuk keamanan Redis
- ElastiCache kelompok subnet
- ElastiCache untuk cadangan Redis
- ElastiCache acara

## ElastiCache simpul

Node adalah blok bangunan terkecil dari sebuah ElastiCache deployment. Simpul dapat berdiri sendiri dari atau terkait dengan simpul lainnya.

Simpul adalah potongan RAM berukuran tetap yang terhubung ke jaringan secara aman. Setiap simpul menjalankan sebuah instans mesin dan versi yang dipilih pada saat Anda membuat klaster Anda. Jika diperlukan, Anda dapat menaikkan atau menurunkan skala simpul dalam klaster ke jenis instans yang berbeda. Untuk informasi selengkapnya, lihat Penskalaan ElastiCache untuk Redis .

Setiap simpul dalam klaster adalah tipe instans yang sama dan menjalankan mesin cache yang sama. Setiap simpul cache mempunyai nama dan port Layanan Nama Domain (DNS) sendiri. Beberapa jenis simpul cache didukung, masing-masing dengan jumlah yang bervariasi dari memori yang terkait. Untuk daftar tipe instans simpul yang didukung, lihat Tipe simpul yang didukung.

Anda dapat membeli node pay-as-you-go berdasarkan, di mana Anda hanya membayar untuk penggunaan node. Anda juga dapat membeli simpul terpesan dengan tarif per jam yang jauh lebih murah. Jika tingkat penggunaan Anda tinggi, pembelian simpul direservasi dapat menghemat uang Anda. Misalkan klaster Anda hampir setiap saat digunakan, dan Anda terkadang menambahkan simpul untuk menangani lonjakan penggunaan. Dalam kasus ini, Anda dapat membeli sejumlah simpul terpesan untuk bekerja pada hampir semua kesempatan. Anda kemudian dapat membeli pay-as-you-go node untuk saat-saat Anda sesekali perlu menambahkan node. Untuk informasi lain tentang simpul direservasi, lihat Simpul terpesan ElastiCache.

Untuk informasi lain tentang simpul, lihat Mengelola simpul.

## ElastiCache untuk pecahan Redis

Serpihan Redis (disebut grup simpul dalam API dan CLI) adalah pengelompokan dari satu hingga enam simpul yang berhubungan. Cluster Redis (mode cluster dinonaktifkan) selalu memiliki setidaknya satu pecahan.

Sharding adalah metode partisi database yang memisahkan database besar menjadi bagian yang lebih kecil, lebih cepat, dan lebih mudah dikelola yang disebut pecahan data. Hal ini dapat meningkatkan efisiensi database dengan mendistribusikan operasi di beberapa bagian terpisah. Menggunakan pecahan dapat menawarkan banyak manfaat termasuk peningkatan kinerja, skalabilitas, dan efisiensi biaya.

Klaster Redis (mode klaster diaktifkan) dapat memiliki hingga 500 serpihan, dengan data Anda dipartisi di seluruh serpihan. Batas simpul atau serpihan dapat ditingkatkan hingga maksimum 500 per klaster jika mesin Redis yang digunakan memiliki versi 5.0.6 atau yang lebih tinggi. Sebagai contoh, Anda dapat memilih untuk membuat konfigurasi dari sebuah klaster dengan 500 simpul yang berkisar antara 83 serpihan (satu primer dan 5 replika per serpihan) dan 500 serpihan (primer tunggal dan tidak ada replika). Pastikan alamat IP yang tersedia mencukupi untuk mengakomodasi peningkatan tersebut. Kesalahan umum termasuk subnet dalam grup subnet memiliki rentang CIDR yang terlalu kecil atau subnet digunakan bersama dan banyak digunakan oleh klaster lain. Untuk informasi selengkapnya, lihat Membuat grup subnet. Untuk versi di bawah 5.0.6, batasnya adalah 250 per klaster.

Untuk meminta penambahan batas, lihat Batas Layanan AWS dan pilih jenis batas Simpul per klaster per jenis instans.

Serpihan beberapa simpul mengimplementasikan replikasi dengan memiliki satu simpul primer baca/tulis dan 1–5 simpul replika. Untuk informasi selengkapnya, lihat Ketersediaan tinggi menggunakan grup replikasi.

Untuk informasi selengkapnya tentang serpihan, lihat Menggunakan serpihan.

## ElastiCache untuk kluster Redis

Klaster Redis adalah pengelompokan logis dari satu atau beberapa ElastiCache untuk pecahan Redis. Data dipartisi di seluruh serpihan di dalam klaster Redis (mode klaster diaktifkan).

Banyak ElastiCache operasi yang ditargetkan pada cluster:

- Membuat klaster
- Mengubah klaster
- Mengambil snapshot klaster (semua versi Redis)
- Menghapus klaster
- Melihat elemen di klaster

- Menambahkan atau menghapus tag alokasi biaya ke dan dari klaster

Untuk informasi selengkapnya, lihat topik terkait berikut:

- Mengelola klaster dan Mengelola simpul

  Informasi tentang klaster, simpul, dan operasi terkait.
- AWS batas layanan: Amazon ElastiCache

  Informasi tentang ElastiCache batas, seperti jumlah maksimum node atau cluster. Untuk melampaui batas tertentu, Anda dapat membuat permintaan menggunakan formulir permintaan node ElastiCache cache Amazon.
- Mitigasi Kegagalan

  Informasi tentang peningkatan toleransi kesalahan klaster dan grup replikasi Anda.

## Konfigurasi klaster yang khas

Berikut adalah konfigurasi klaster umum.

Klaster Redis

Klaster Redis (mode klaster dinonaktifkan) selalu berisi hanya satu serpihan (dalam API dan CLI, satu grup simpul). Serpihan Redis berisi satu hingga enam simpul. Jika terdapat lebih dari satu simpul dalam sebuah serpihan, serpihan tersebut mendukung replikasi. Dalam kasus ini, satu simpul adalah simpul primer baca/tulis dan yang lain adalah simpul replika baca-saja.

Untuk meningkatkan toleransi kesalahan, sebaiknya miliki setidaknya dua simpul dalam klaster Redis dan aktifkan Multi-AZ. Untuk informasi selengkapnya, lihat Mitigasi Kegagalan.

Seiring berubahnya permintaan atas klaster Redis (mode klaster), Anda dapat menaikkan atau menurunkan skala. Untuk melakukan hal ini, Anda memindahkan klaster Anda ke jenis instans simpul yang berbeda. Jika aplikasi Anda intensif baca, kami sarankan untuk menambahkan replika baca-saja klaster Redis (mode klaster dinonaktifkan). Dengan melakukan ini, Anda dapat menyebarkan pembacaan ke jumlah simpul yang lebih tepat.

Anda juga dapat menggunakan tingkatan data. Data yang lebih sering diakses disimpan dalam memori dan data yang lebih jarang diakses disimpan di disk. Keuntungan menggunakan tingkatan data adalah mengurangi kebutuhan memori. Untuk informasi selengkapnya, lihat Tingkatan data.

ElastiCache mendukung perubahan tipe node cluster Redis (mode cluster dinonaktifkan) ke tipe node yang lebih besar secara dinamis. Untuk informasi tentang kenaikan atau penurunan skala, lihat Menskalakan klaster simpul tunggal untuk Redis (Mode klaster Dinonaktifkan) atau Penskalaan Redis (Mode Cluster Dinonaktifkan) cluster dengan simpul replika.

## ElastiCache untuk replikasi Redis

Replikasi diimplementasikan dengan pengelompokan dua hingga simpul ke dalam serpihan (dalam API dan CLI, disebut grup simpul). Salah satu simpul ini adalah simpul primer baca/tulis. Semua simpul lain adalah simpul replika baca-saja.

Setiap replika baca berisi salinan data dari simpul primer. Simpul replika menggunakan mekanisme replikasi asinkron untuk tetap sinkron dengan simpul primer. Aplikasi dapat membaca dari simpul mana pun dalam klaster, tetapi hanya dapat menulis ke simpul primer. Replika baca meningkatkan skalabilitas dengan menyebarkan proses baca ke beberapa titik akhir. Replika baca juga meningkatkan toleransi kesalahan dengan mempertahankan beberapa salinan data. Menemukan replika baca di beberapa Zona Ketersediaan dapat meningkatkan toleransi kesalahan lebih baik lagi. Untuk informasi selengkapnya tentang toleransi kesalahan, lihat Mitigasi Kegagalan.

Klaster Redis (mode klaster dinonaktifkan) mendukung satu serpihan (dalam API dan CLI, disebut grup simpul).

Replikasi dari perspektif API dan CLI menggunakan terminologi yang berbeda untuk mempertahankan kompatibilitas dengan versi sebelumnya, tetapi hasilnya sama. Tabel berikut menunjukkan istilah API dan CLI untuk menerapkan replikasi.

Membandingkan Replikasi: Redis (mode klaster dinonaktifkan) dan Redis (mode klaster diaktifkan)

Dalam tabel berikut, Anda dapat menemukan perbandingan fitur grup replikasi Redis (mode klaster dinonaktifkan) dan Redis (mode klaster diaktifkan).

| | Redis (mode klaster dinonaktifkan) | Redis (mode klaster diaktifkan) |
|---|---|---|
| Serpihan (grup simpul) | 1 | 1–500 |
| Replika untuk setiap serpihan (grup simpul) | 0–5 | 0–5 |
| Pembuatan partisi data | Tidak | Ya |

| | Redis (mode klaster dinonaktifkan) | Redis (mode klaster diaktifkan) |
|---|---|---|
| Tambah/Hapus replika | Ya | Ya |
| Tambah/Hapus grup simpul | Tidak | Ya |
| Mendukung kenaikan skala | Ya | Ya |
| Mendukung peningkatan mesin | Ya | Ya |
| Naikkan replika menjadi primer | Ya | Otomatis |
| Multi-AZ | Opsional | Wajib |
| Cadangkan/Pulihkan | Ya | Ya |
| Catatan: | | |
| Jika primer tidak memiliki replika dan gagal, semua data primer akan hilang. | | |
| Anda dapat menggunakan cadangan dan pemulihan untuk bermigrasi ke Redis (mode klaster diaktifkan). | | |
| Anda dapat menggunakan cadangan dan pemulihan untuk mengatur ulang ukuran ke klaster Redis (mode klaster diaktifkan). | | |

Semua serpihan (dalam API dan CLI, grup simpul) dan simpul harus berada di Wilayah AWS yang sama. Namun, Anda dapat menyediakan node individual di beberapa Availability Zone dalam AWS Region tersebut.

Replika baca melindungi dari potensi kehilangan data karena data Anda direplikasi pada dua simpul atau lebih—primer dan satu atau beberapa replika baca. Untuk meningkatkan keandalan dan mempercepat pemulihan, kami sarankan Anda membuat satu atau beberapa replika baca di Zona Ketersediaan yang berbeda.

Anda juga dapat memanfaatkan penyimpanan data Global. Dengan menggunakan fitur Global Datastore for Redis, Anda dapat bekerja dengan replikasi yang dikelola sepenuhnya, cepat, andal,

dan aman di seluruh Wilayah. AWS Dengan menggunakan fitur ini, Anda dapat membuat klaster replika baca lintas wilayah ElastiCache untuk Redis guna mengaktifkan pembacaan latensi rendah dan pemulihan bencana di seluruh Wilayah. AWS Untuk informasi selengkapnya, lihat Replikasi lintas AWS Wilayah menggunakan datastores global.

Replikasi: Batas dan pengecualian

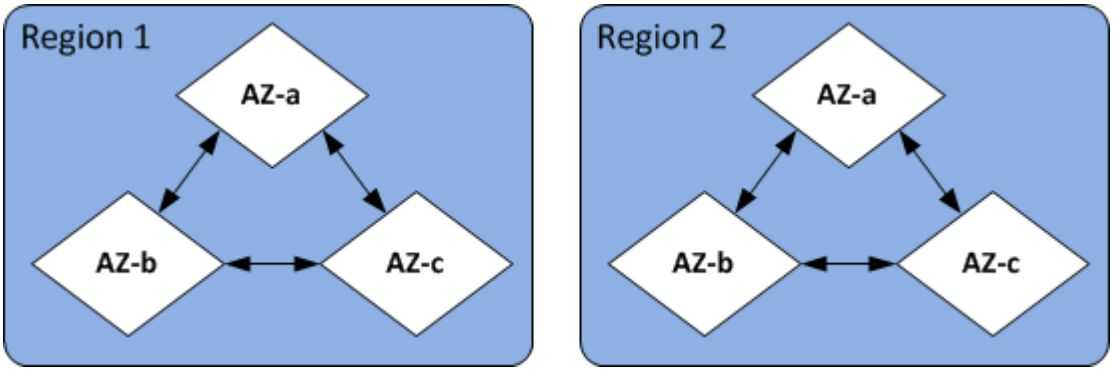
- Multi-AZ tidak didukung pada jenis simpul T1.

## AWS Wilayah dan zona ketersediaan

Amazon ElastiCache tersedia di beberapa AWS Wilayah di seluruh dunia. Dengan demikian, Anda dapat meluncurkan ElastiCache cluster di lokasi yang memenuhi persyaratan bisnis Anda. Misalnya, Anda dapat meluncurkan di AWS Wilayah terdekat dengan pelanggan Anda atau untuk memenuhi persyaratan hukum tertentu.

Secara default, AWS SDK, ElastiCache API AWS CLI, dan ElastiCache konsol mereferensikan Wilayah AS Barat (Oregon). Saat ElastiCache memperluas ketersediaan ke AWS Wilayah baru, titik akhir baru untuk AWS Wilayah ini juga tersedia. Anda dapat menggunakan ini dalam permintaan HTTP, AWS SDK AWS CLI, dan ElastiCache konsol.

Setiap AWS Wilayah dirancang untuk sepenuhnya terisolasi dari AWS Wilayah lain. Di dalam setiap wilayah terdapat beberapa Availability Zone. Dengan meluncurkan simpul Anda di Availability Zone yang berbeda, Anda dapat mencapai toleransi kesalahan sebesar mungkin. Untuk informasi selengkapnya tentang AWS Wilayah dan Availability Zone, lihatMemilih wilayah dan zona ketersediaan. Dalam diagram berikut, Anda dapat melihat tampilan tingkat tinggi tentang cara kerja AWS Wilayah dan Zona Ketersediaan.




Untuk informasi tentang AWS Wilayah yang didukung oleh ElastiCache dan titik akhirnya, lihatWilayah & titik akhir yang didukung.

# ElastiCache untuk titik akhir Redis

Endpoint adalah alamat unik yang digunakan aplikasi Anda untuk terhubung ke ElastiCache node atau cluster.

## Titik akhir satu simpul untuk Redis (Mode Klaster Dinonaktifkan)

Titik akhir untuk klaster Redis satu simpul digunakan untuk menyambung ke klaster untuk pembacaan dan penulisan.

## Titik akhir multi-simpul untuk Redis (Mode Klaster Dinonaktifkan)

Klaster Redis multi-simpul (mode klaster dinonaktifkan) memiliki dua jenis titik akhir. Titik akhir primer selalu tersambung ke simpul primer dalam klaster, bahkan jika simpul tertentu dalam peran primer berubah. Gunakan titik akhir primer untuk semua penulisan ke klaster.

Gunakan Titik Akhir Pembaca untuk membagi koneksi masuk ke titik akhir secara merata di antara semua replika baca. GunakanTitik Akhir Simpul individual untuk operasi baca (Dalam API/CLI, ini disebut sebagai Titik Akhir Baca).

## Titik akhir Redis (Mode Klaster Diaktifkan)

Sebuah klaster Redis (mode klaster diaktifkan) memiliki satu titik akhir konfigurasi. Dengan terhubung ke titik akhir konfigurasi, aplikasi Anda mampu menemukan titik akhir primer dan baca untuk setiap serpihan di klaster.

Untuk informasi selengkapnya, lihat Menemukan titik akhir koneksi.

# ElastiCache kelompok parameter

Grup parameter cache adalah cara mudah untuk mengelola pengaturan runtime untuk perangkat lunak mesin yang didukung. Parameter digunakan untuk mengontrol penggunaan memori, kebijakan pengosongan, ukuran item, dan lainnya. Grup ElastiCache parameter adalah kumpulan bernama parameter khusus mesin yang dapat Anda terapkan ke cluster. Dengan melakukan ini, Anda memastikan bahwa semua simpul dalam klaster dikonfigurasi dengan cara yang pasti sama.

Untuk daftar parameter yang didukung, nilai defaultnya, dan parameter mana yang dapat dimodifikasi, lihat DescribeEngineDefaultParameters (CLI: describe-engine-default-parameters).

Untuk informasi lebih rinci tentang grup ElastiCache parameter, lihatMengonfigurasi parameter mesin menggunakan grup parameter.

## ElastiCache untuk keamanan Redis

Untuk keamanan yang ditingkatkan, ElastiCache akses node Redis dibatasi untuk aplikasi yang berjalan di instans Amazon EC2 yang Anda izinkan. Anda dapat mengontrol instans Amazon EC2 yang dapat mengakses klaster Anda menggunakan grup keamanan.

Secara default, semua cluster Redis baru ElastiCache diluncurkan di lingkungan Amazon Virtual Private Cloud (Amazon VPC). Anda dapat menggunakan grup subnet untuk memberikan akses klaster dari instans Amazon EC2 yang berjalan di subnet tertentu.

Selain membatasi akses node, ElastiCache untuk Redis mendukung TLS dan enkripsi di tempat untuk node yang menjalankan versi tertentu untuk Redis. ElastiCache Untuk informasi selengkapnya, lihat hal berikut:

- Keamanan data di Amazon ElastiCache
- Autentikasi dengan perintah AUTH Redis

## ElastiCache kelompok subnet

Grup subnet adalah kumpulan subnet (biasanya privat) yang dapat Anda tetapkan untuk klaster Anda yang berjalan di lingkungan Amazon VPC.

Jika Anda membuat klaster di Amazon VPC, Anda harus menentukan grup subnet cache. ElastiCache menggunakan kelompok subnet cache itu untuk memilih subnet dan alamat IP dalam subnet itu untuk dikaitkan dengan node cache Anda.

Untuk informasi selengkapnya tentang penggunaan grup subnet cache di lingkungan Amazon VPC, lihat hal berikut.

- Amazon VPC dan keamanan ElastiCache
- Langkah 3: Mengizinkan akses ke klaster
- Subnet dan grup subnet

## ElastiCache untuk cadangan Redis

Cadangan adalah point-in-time salinan dari cluster Redis. Cadangan dapat digunakan untuk memulihkan klaster yang ada atau menyemai klaster baru. Cadangan terdiri dari semua data dalam klaster ditambah beberapa metadata.

Bergantung pada versi Redis yang berjalan di klaster Anda, proses pencadangan membutuhkan jumlah memori terpesan yang berbeda-beda agar berhasil. Untuk informasi selengkapnya, lihat hal berikut:

- Melakukan snapshot dan pemulihan
- Cara penerapan sinkronisasi dan pencadangan
- Dampak performa pencadangan klaster yang dirancang sendiri
- Memastikan bahwa Anda memiliki cukup memori untuk membuat snapshot Redis

## ElastiCache acara

Saat peristiwa penting terjadi di cluster cache, ElastiCache kirimkan pemberitahuan ke topik Amazon SNS tertentu. Peristiwa penting dapat mencakup hal seperti kegagalan atau keberhasilan penambahan simpul, perubahan grup keamanan, dan lainnya. Dengan memantau peristiwa penting, Anda dapat mengetahui status klaster terbaru Anda dan dalam banyak kasus dapat mengambil tindakan korektif.

Untuk informasi lebih lanjut tentang ElastiCache acara, lihatPemantauan Amazon SNS dari peristiwa ElastiCache.

# Terminologi ElastiCache for Redis

Pada bulan Oktober 2016, Amazon ElastiCache meluncurkan dukungan untuk Redis 3.2. Pada saat itu, kami menambahkan dukungan untuk membuat partisi data Anda hingga 500 serpihan (disebut grup simpul dalam API ElastiCache dan AWS CLI). Agar kompatibel dengan versi sebelumnya, kami memperluas operasi API versi 2015-02-02 untuk menyertakan fungsionalitas Redis yang baru.

Pada saat yang sama, kami mulai menggunakan terminologi di konsol ElastiCache yang digunakan dalam fungsionalitas baru ini dan bersifat umum di seluruh industri. Perubahan ini berarti bahwa pada beberapa tahap, terminologi yang digunakan di API dan CLI mungkin berbeda dari terminologi yang digunakan di konsol. Daftar berikut mengidentifikasi istilah yang mungkin berbeda antara API, CLI, dan konsol.

Klaster cache atau simpul vs. simpul

Terdapat hubungan satu-ke-satu antara simpul dan klaster cache jika tidak ada simpul replika. Oleh karena itu, konsol ElastiCache sering menggunakan istilah-istilah tersebut secara bergantian. Konsol sekarang menggunakan istilah simpul secara menyeluruh. Satu-satunya pengecualian adalah tombol Buat Klaster, yang memulai proses untuk membuat klaster dengan atau tanpa simpul replika.

API ElastiCache AWS CLI dan tetap menggunakan istilah lama.

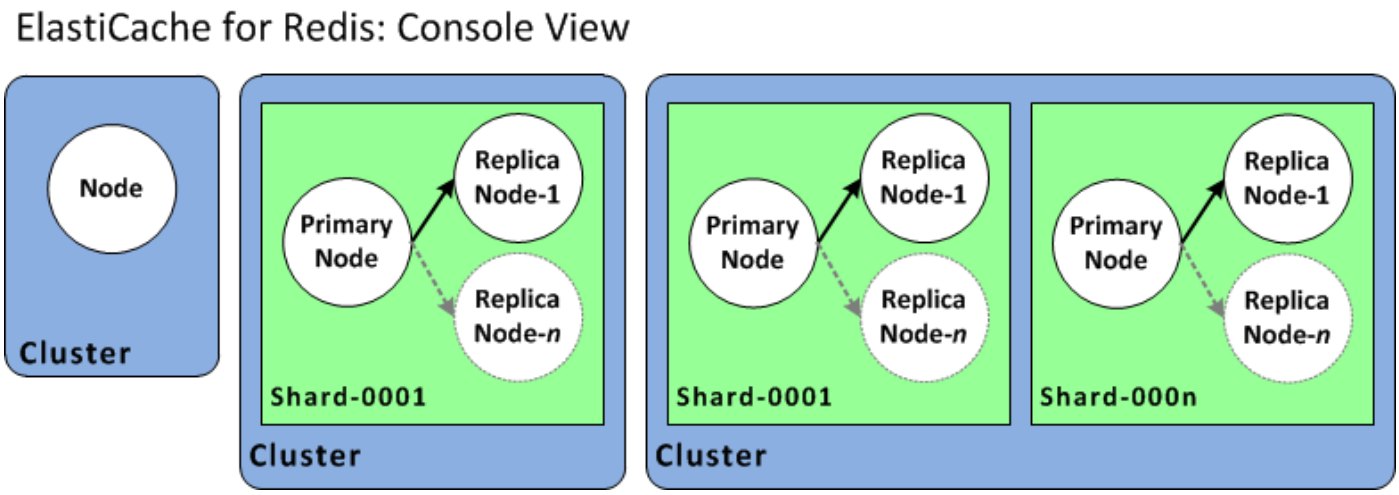
Klaster vs. grup replikasi

Konsol sekarang menggunakan istilah klaster untuk semua klaster ElastiCache for Redis. Konsol menggunakan istilah klaster dalam semua kondisi berikut:

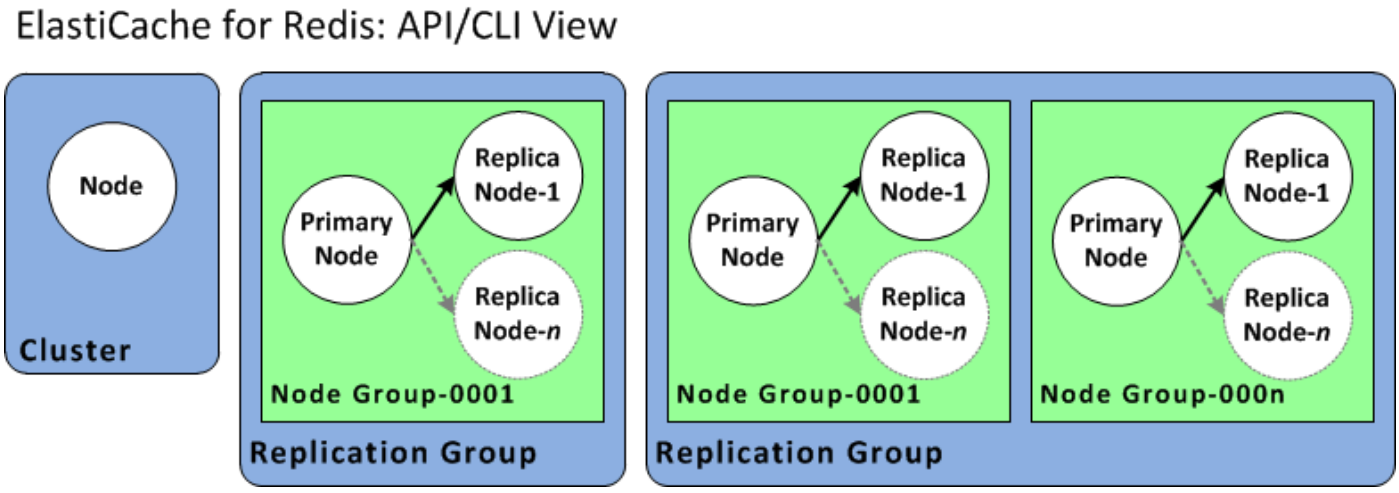
- Saat klaster adalah klaster Redis satu simpul.
- Saat klaster berupa klaster Redis (mode klaster dinonaktifkan) yang mendukung replikasi dalam satu serpihan (dalam API dan CLI, disebut grup simpul).
- Saat klaster berupa klaster Redis (mode klaster diaktifkan) yang mendukung replikasi dalam 1-90 serpihan atau hingga 500 dengan permintaan kenaikan batas. Untuk meminta kenaikan batas, lihat Batas layanan AWS dan pilih jenis batas Simpul per klaster per jenis instans.

Untuk informasi selengkapnya tentang grup replikasi, lihat Ketersediaan tinggi menggunakan grup replikasi.

Diagram berikut menggambarkan berbagai topologi klaster ElastiCache for Redis dari sudut pandang konsol.




API ElastiCache dan operasi AWS CLI masih membedakan klaster satu simpul ElastiCache for Redis dari grup replikasi multi-simpul. Diagram berikut menggambarkan berbagai topologi ElastiCache for Redis dari sudut pandang API ElastiCache dan AWS CLI.




Grup replikasi vs. penyimpanan data global

Penyimpanan data global adalah kumpulan dari satu atau beberapa klaster yang mereplikasi satu sama lain di seluruh Wilayah, sedangkan grup replikasi mereplikasi data melalui klaster yang mengaktifkan mode klaster dengan beberapa serpihan. Penyimpanan data global terdiri dari hal berikut:

- Klaster primer (aktif) – Klaster primer menerima proses tulis yang direplikasikan ke semua klaster di dalam penyimpanan data global. Klaster primer juga menerima permintaan baca.
- Klaster sekunder (pasif) – Klaster sekunder hanya menerima permintaan baca dan mereplikasikan pembaruan data dari klaster primer. Klaster sekunder harus berada di Wilayah AWS yang berbeda dari klaster primer.

Untuk informasi tentang penyimpanan data global, lihat Replikasi di seluruh Wilayah AWS menggunakan penyimpanan data global.

# Merancang klaster Anda sendiri

Berikut ini adalah tindakan satu kali yang harus Anda ambil untuk mulai mendesain ElastiCache cluster Anda.

Topik



## Pengaturan

Sebelum membuat klaster, grup subnet harus dibuat terlebih dahulu. Grup subnet cache adalah kumpulan subnet yang dapat ditetapkan untuk klaster cache Anda di dalam VPC. Saat meluncurkan klaster cache di VPC, Anda harus memilih grup subnet cache. Kemudian ElastiCache menggunakan grup subnet cache itu untuk memberikan alamat IP di dalam subnet itu ke setiap simpul cache di dalam klaster.

Saat Anda membuat grup subnet baru, perhatikan jumlah alamat IP yang tersedia. Jika subnet memiliki sangat sedikit alamat IP yang bebas, Anda akan dibatasi dalam hal jumlah simpul yang dapat ditambahkan ke klaster. Untuk mengatasi masalah ini, Anda dapat menetapkan satu atau beberapa subnet ke grup subnet sehingga Anda memiliki jumlah alamat IP yang cukup di dalam Zona Ketersediaan dari klaster Anda. Setelah itu, Anda dapat menambahkan lebih banyak simpul ke klaster Anda.

Untuk informasi lebih lanjut tentang pengaturan ElastiCache lihat Pengaturan.

## Langkah 1: Membuat grup subnet

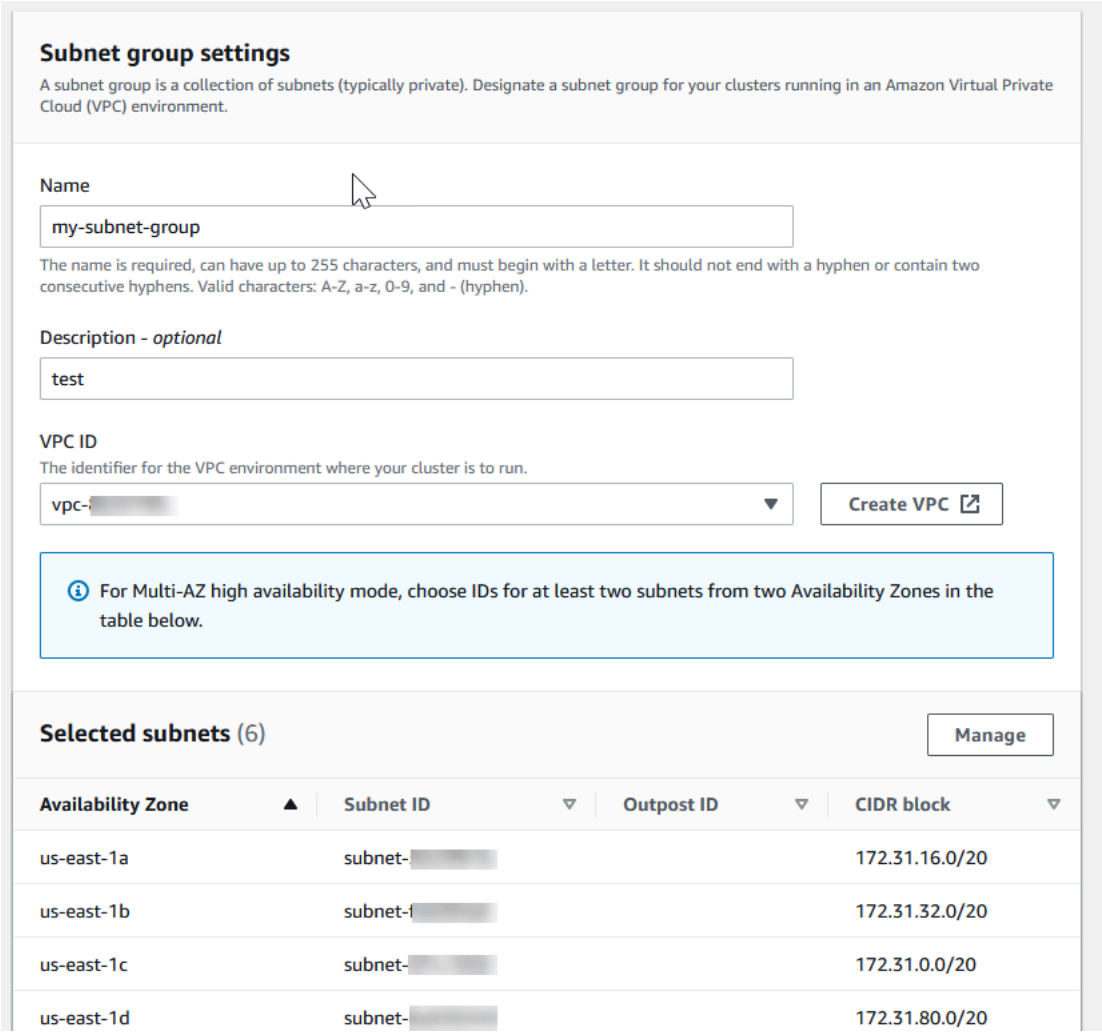
Prosedur berikut menunjukkan cara membuat grup subnet yang disebut `mysubnetgroup` (konsol), dan AWS CLI.

## Membuat grup subnet (Konsol)

Prosedur berikut menunjukkan cara membuat grup subnet (konsol).

Untuk membuat grup subnet (Konsol)

1. Masuk ke Konsol Manajemen AWS dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Di daftar navigasi, pilih Grup subnet.
3. Pilih Buat Grup Subnet.
4. Pada wizard Buat Grup Subnet, lakukan hal berikut. Jika semua pengaturan sudah sesuai keinginan Anda, pilih Ya, Buat.
    a. Pada kotak Nama, ketik nama grup subnet Anda.
    b. Di kotak Deskripsi, ketik deskripsi untuk grup subnet Anda.
    c. Pada kotak ID VPC, pilih Amazon VPC yang Anda buat.
    d. Pada daftar Zona Ketersediaan dan ID Subnet, pilih Zona Ketersediaan atau Zona Lokal dan ID subnet privat Anda, lalu pilih Tambahkan.




5. Pada pesan konfirmasi yang muncul, pilih Tutup.

Grup subnet baru Anda muncul pada daftar Grup Subnet dari konsol ElastiCache. Di bagian bawah jendela Anda dapat memilih grup subnet untuk melihat perincian, misalnya semua subnet yang terkait dengan grup ini.

Membuat grup subnet (AWS CLI)

Pada prompt perintah, gunakan perintah `create-cache-subnet-group` untuk membuat grup subnet.

Untuk Linux, macOS, atau Unix:

```
aws elasticache create-cache-subnet-group \
    --cache-subnet-group-name mysubnetgroup \
    --cache-subnet-group-description "Testing" \
```

```
    --subnet-ids subnet-53df9c3a
```

Untuk Windows:

```
aws elasticache create-cache-subnet-group ^
    --cache-subnet-group-name mysubnetgroup ^
    --cache-subnet-group-description "Testing" ^
    --subnet-ids subnet-53df9c3a
```

Perintah ini seharusnya menghasilkan keluaran yang serupa dengan yang berikut:

```
{
    "CacheSubnetGroup": {
        "VpcId": "vpc-37c3cd17",
        "CacheSubnetGroupDescription": "Testing",
        "Subnets": [
            {
                "SubnetIdentifier": "subnet-53df9c3a",
                "SubnetAvailabilityZone": {
                    "Name": "us-west-2a"
                }
            }
        ],
        "CacheSubnetGroupName": "mysubnetgroup"
    }
}
```

Untuk informasi selengkapnya, lihat AWS CLI topik create-cache-subnet-group.

## Langkah 2: Buat klaster

Sebelum membuat klaster untuk tujuan produksi, Anda tentu perlu mempertimbangkan pengaturan konfigurasi klaster untuk memenuhi kebutuhan bisnis Anda. Masalah terkait itu dibahas di bagian Menyiapkan klaster. Untuk tujuan latihan Memulai ini, Anda akan membuat klaster dengan mode klaster dinonaktifkan dan Anda dapat menggunakan nilai konfigurasi default jika sesuai.

Klaster yang Anda buat akan berjalan langsung, dan tidak berjalan di sandbox. Anda akan dikenakan biaya ElastiCache penggunaan standar untuk instans sampai Anda menghapusnya. Jumlah biayanya cukup kecil (biasanya kurang dari satu dolar) jika Anda menyelesaikan latihan yang dijelaskan di sini dalam satu sesi dan menghapus klaster itu ketika Anda sudah selesai. Untuk informasi selengkapnya tentang tarif ElastiCache penggunaan, lihat Amazon ElastiCache.

Klaster Anda diluncurkan dalam cloud privat virtual (VPC) berdasarkan layanan Amazon VPC.

## Membuat klaster Redis (Mode Klaster Dinonaktifkan) (Konsol)

Untuk membuat cluster Redis (mode cluster dinonaktifkan) menggunakan konsol ElastiCache

1. Masuk ke AWS Management Console dan buka ElastiCache konsol Amazon di https://console.aws.amazon.com/elasticache/.
2. Dari daftar di sudut kanan atas, pilih AWS Wilayah tempat Anda ingin meluncurkan cluster ini.
3. Pilih Mulai di panel navigasi.
4. Pilih Buat VPC dan ikuti langkah yang telah dijelaskan pada Membuat Cloud Privat Virtual (VPC).
5. Pada halaman ElastiCache dasbor, pilih Redis cache dan kemudian pilih Buat Redis cache.
6. Pada Pengaturan klaster, lakukan hal berikut:

   a. Pilih Konfigurasi dan buat klaster baru.

   b. Untuk Mode klaster, pilih Dinonaktifkan.

   c. Untuk Info klaster masukkan nilai untuk Nama.

   d. (Opsional) Masukkan nilai untuk Deskripsi.
7. Di bawah Lokasi:

   AWS Cloud

   1. Untuk AWS Cloud, sebaiknya Anda menerima pengaturan default untuk Multi-AZ dan Auto-failover. Untuk informasi selengkapnya, lihat Meminimalkan waktu henti ElastiCache untuk Redis dengan Multi-AZ.
   2. Di bawah Pengaturan klaster

      a. Untuk Versi mesin, pilih versi yang tersedia.

      b. Untuk Port, gunakan port default, 6379. Jika Anda ingin menggunakan port yang berbeda, tuliskan nomor port tersebut.

      c. Untuk Grup parameter, pilih grup parameter atau buat yang baru. Grup parameter mengontrol parameter runtime klaster Anda. Untuk informasi selengkapnya tentang grup parameter, lihat Parameter spesifik Redis dan Membuat grup parameter.

> **Note**
>
> Saat Anda memilih grup parameter untuk menetapkan nilai konfigurasi mesin, grup parameter tersebut diterapkan ke semua klaster di penyimpanan data global. Pada halaman Grup Parameter, atribut Global ya/tidak menunjukkan apakah grup parameter adalah bagian dari penyimpanan data global.

d. Untuk Jenis Simpul, pilih panah bawah
( ).
Pada kotak dialog Ubah jenis simpul, pilih nilai untuk Keluarga instans untuk jenis simpul yang Anda inginkan. Kemudian pilih jenis simpul yang ingin Anda gunakan untuk klaster ini, lalu pilih Simpan.

Untuk informasi selengkapnya, lihat Memilih ukuran simpul Anda.

Jika Anda memilih jenis simpul r6gd, tingkatan data diaktifkan secara otomatis. Untuk informasi selengkapnya, lihat Tingkatan data.

e. Untuk Jumlah replika, pilih jumlah replika baca yang Anda inginkan. Jika Anda mengaktifkan Multi-AZ, jumlahnya harus antara 1-5.

3. Di bawah Konektivitas

a. Untuk Jenis jaringan, pilih versi IP yang akan didukung oleh klaster ini.

b. Untuk grup Subnet, pilih subnet yang ingin Anda terapkan ke cluster ini. ElastiCache menggunakan grup subnet itu untuk memilih subnet dan alamat IP dalam subnet itu untuk dikaitkan dengan node Anda. ElastiCache cluster memerlukan subnet dual-stack dengan alamat IPv4 dan IPv6 yang ditetapkan untuk beroperasi dalam mode dual-stack dan subnet khusus IPv6 untuk beroperasi sebagai IPv6 saja.

Saat membuat grup subnet baru, masukkan ID VPC yang menaungi grup subnet tersebut.

Untuk informasi selengkapnya, lihat:

- Memilih jenis jaringan.
- Membuat subnet di VPC Anda.

Jika Anda adalah Menggunakan zona lokal dengan ElastiCache , Anda harus membuat atau memilih subnet yang berada di zona lokal.

Untuk informasi selengkapnya, lihat Subnet dan grup subnet.

4. Untuk Penempatan Zona Ketersediaan, Anda memiliki dua opsi:

   - Tidak ada preferensi — ElastiCache memilih Availability Zone.
   - Tentukan zona ketersediaan – Anda menentukan Zona Ketersediaan untuk setiap klaster.

   Jika Anda memilih untuk menentukan Zona Ketersediaan, untuk setiap klaster di setiap serpihan, pilih Zona Ketersediaan dari daftar.

   Untuk informasi selengkapnya, lihat Memilih wilayah dan zona ketersediaan.

5. Pilih Selanjutnya
6. Di bawah Pengaturan lanjutan Redis

   - Untuk Keamanan:

     i. Untuk mengenkripsi data Anda, Anda memiliki opsi berikut:

     - Enkripsi saat diam – Mengaktifkan enkripsi pada data yang disimpan di disk. Untuk informasi selengkapnya, lihat Enkripsi Data Diam.

       > **Note**
       >
       > Anda memiliki opsi untuk menyediakan kunci enkripsi yang berbeda dengan memilih kunci AWS KMS yang Dikelola Pelanggan dan memilih kunci. Untuk informasi selengkapnya, lihat Menggunakan kunci yang dikelola pelanggan dari AWS KMS.

     - Enkripsi Data Bergerak – Mengaktifkan enkripsi pada data selama pengiriman. Untuk informasi selengkapnya, lihat Enkripsi Data Bergerak. Untuk mesin Redis versi 6.0 dan di atasnya, jika Anda mengaktifkan Enkripsi data bergerak, Anda akan diminta untuk menentukan salah satu dari opsi Kontrol Akses berikut:

- Tidak ada Kontrol Akses – Ini adalah pengaturan default. Ini menunjukkan tidak ada pembatasan akses pengguna ke klaster.
- Daftar Kontrol Akses Grup Pengguna – Pilih grup pengguna dengan kelompok pengguna tertentu yang dapat mengakses klaster. Untuk informasi selengkapnya, lihat Mengelola Grup Pengguna dengan Konsol dan CLI.
- Pengguna Default Redis AUTH – Mekanisme autentikasi untuk server Redis. Untuk informasi selengkapnya, silakan lihat Redis AUTH.

- Redis AUTH - Mekanisme autentikasi untuk server Redis. Untuk informasi selengkapnya, silakan lihat Redis AUTH.

> **Note**
>
> Untuk versi Redis antara 3.2.6 dan seterusnya, kecuali versi 3.2.10, Redis AUTH adalah satu-satunya pilihan.

ii. Untuk Grup keamanan, pilih grup keamanan yang Anda inginkan untuk klaster ini. Grup keamanan bertindak sebagai firewall untuk mengontrol akses jaringan ke klaster Anda. Anda dapat menggunakan grup keamanan default untuk VPC Anda atau membuat yang baru.

Untuk informasi grup keamanan selengkapnya, lihat Grup Keamanan untuk VPC Anda di Panduan Pengguna Amazon VPC.

7. Untuk pencadangan otomatis terjadwal rutin, pilih Aktifkan pencadangan otomatis, lalu masukkan jumlah hari yang diinginkan untuk menyimpan cadangan otomatis sebelum dihapus secara otomatis. Jika Anda tidak ingin melakukan pencadangan otomatis terjadwal rutin, hapus kotak centang Aktifkan pencadangan otomatis. Apa pun pilihannya, Anda selalu bisa membuat pencadangan secara manual.

   Untuk informasi selengkapnya pencadangan dan pemulihan Redis, lihat Melakukan snapshot dan pemulihan.

8. (Opsional) Menentukan jendela pemeliharaan. Jendela pemeliharaan adalah waktu yang biasanya satu jam setiap minggu saat ElastiCache menjadwalkan pemeliharaan sistem untuk klaster Anda. Anda dapat mengizinkan ElastiCache untuk memilih hari dan waktu untuk jendela pemeliharaan Anda (Tidak ada preferensi), atau Anda dapat memilih hari, waktu, dan durasi sendiri (Tentukan jendela pemeliharaan). Jika Anda memilih Tentukan

jendela pemeliharaan dari daftar, pilih Hari mulai, Waktu mulai, dan Durasi (dalam jam) untuk jendela pemeliharaan. Waktu yang digunakan adalah UCT.

Untuk informasi selengkapnya, lihat Mengelola pemeliharaan.

9. (Opsional) Untuk Log:
   - Di bawah Format log, pilih salah satu dari Teks atau JSON.
   - Di bawah Jenis Tujuan, pilih CloudWatch Log atau Kinesis Firehose.
   - Di bawah Tujuan log, pilih Buat baru dan masukkan nama grup CloudWatch log Log atau nama aliran Firehose Anda, atau pilih Pilih yang ada, lalu pilih nama grup CloudWatch log Log atau nama aliran Firehose Anda,
10. Untuk Tag, untuk membantu mengelola kluster dan ElastiCache sumber daya lainnya, Anda dapat menetapkan metadata Anda sendiri ke setiap sumber daya dalam bentuk tag. Untuk informasi selengkapnya, lihat Menandai sumber daya ElastiCache Anda.
11. Pilih Selanjutnya.
12. Tinjau semua entri dan pilihan Anda, lalu lakukan koreksi yang diperlukan. Saat Anda siap, pilih Buat.

On premises

1. Untuk On-premis, sebaiknya Anda membiarkan Auto-failover tetap aktif. Untuk informasi selengkapnya, lihat Meminimalkan waktu henti ElastiCache untuk Redis dengan Multi-AZ
2. Untuk menyelesaikan pembuatan klaster, ikuti langkah-langkah di Menggunakan Outposts.

Setelah status klaster Anda menjadi tersedia, Anda dapat memberikan akses ke Amazon EC2, menyambungkannya, dan mulai menggunakannya. Lihat informasi yang lebih lengkap di Langkah 3: Mengizinkan akses ke klaster dan Langkah 4: Menyambung ke simpul klaster.

> **Important**
>
> Setelah klaster Anda tersedia, Anda akan ditagih untuk setiap jam atau durasi saat klaster aktif, meskipun Anda tidak sedang aktif menggunakannya. Untuk menghentikan tagihan biaya untuk klaster ini, Anda harus menghapusnya. Lihat Menghapus klaster.

### Membuat klaster Redis (Mode Klaster Dinonaktifkan) (AWS CLI)

#### Example

Kode CLI berikut membuat klaster cache Redis (mode klaster dinonaktifkan) tanpa replika.

Untuk Linux, macOS, atau Unix:

```
aws elasticache create-cache-cluster \
--cache-cluster-id my-cluster \
--cache-node-type cache.r4.large \
--engine redis \
--num-cache-nodes 1 \
--snapshot-arns arn:aws:s3:::my_bucket/snapshot.rdb
```

Untuk Windows:

```
aws elasticache create-cache-cluster ^
--cache-cluster-id my-cluster ^
--cache-node-type cache.r4.large ^
--engine redis ^
--num-cache-nodes 1 ^
--snapshot-arns arn:aws:s3:::my_bucket/snapshot.rdb
```

Untuk bekerja dengan mode klaster diaktifkan, lihat topik berikut:

- Untuk menggunakan konsol, lihat Membuat klaster Redis (mode klaster diaktifkan) (Konsol).
- Untuk menggunakan AWS CLI, lihatMembuat klaster Redis (modus klaster diaktifkan) (AWS CLI).

# Langkah 3: Mengizinkan akses ke klaster

Bagian ini mengasumsikan bahwa Anda telah memahami peluncuran dan penyambungan ke instans Amazon EC2. Untuk informasi selengkapnya, lihat Panduan Memulai Amazon EC2.

Semua klaster ElastiCache dirancang untuk diakses dari instans Amazon EC2. Skenario yang paling umum adalah untuk mengakses klaster ElastiCache dari instans Amazon EC2 di Amazon Virtual Private Cloud (Amazon VPC) yang sama, yang akan digunakan untuk latihan ini.

Secara default, akses jaringan ke klaster Anda dibatasi untuk akun yang digunakan untuk membuatnya. Sebelum Anda dapat menyambung ke klaster dari instans EC2, Anda harus memberikan otorisasi pada instans EC2 untuk mengakses klaster tersebut. Langkah yang diperlukan tergantung pada apakah Anda meluncurkan klaster Anda ke EC2-VPC atau EC2-Classic.

Kasus penggunaan yang paling umum adalah saat aplikasi yang disebarkan pada instans EC2 perlu terhubung ke klaster di VPC yang sama. Cara paling sederhana untuk mengelola akses antara instans EC2 dan klaster di VPC yang sama adalah dengan melakukan tindakan berikut:

1. Buat grup keamanan VPC untuk klaster Anda. Grup keamanan ini dapat digunakan untuk membatasi akses ke instans klaster. Sebagai contoh, Anda dapat membuat aturan khusus untuk grup keamanan ini yang mengizinkan akses TCP menggunakan port yang Anda tetapkan untuk klaster saat Anda membuatnya dan alamat IP yang Anda gunakan untuk mengakses klaster tersebut.

   Port default untuk klaster dan grup replikasi Redis adalah 6379.

   > **Important**
   >
   > Grup keamanan Amazon ElastiCache hanya dapat diaplikasikan untuk klaster yang tidak berjalan di lingkungan Amazon Virtual Private Cloud (Amazon VPC). Jika Anda menjalankannya di Amazon Virtual Private Cloud, Grup Keamanan tidak tersedia pada panel navigasi konsol.
   > Jika Anda menjalankan simpul ElastiCache Anda di Amazon VPC, Anda mengontrol akses ke klaster Anda dengan grup keamanan Amazon VPC, yang berbeda dari grup keamanan ElastiCache. Untuk informasi selengkapnya penggunaan ElastiCache di Amazon VPC, lihat Amazon VPC dan keamanan ElastiCache

2. Buat grup keamanan VPC untuk instans EC2 Anda (server web dan aplikasi). Jika diperlukan, grup keamanan ini dapat mengizinkan akses ke instans EC2 dari Internet dengan menggunakan

tabel perutean VPC. Sebagai contoh, Anda dapat menetapkan aturan pada grup keamanan ini untuk mengizinkan akses TCP ke instans EC2 melalui port 22.

3. Buat aturan kustom di grup keamanan untuk klaster Anda yang memungkinkan koneksi dari grup keamanan yang Anda buat untuk instans EC2 Anda. Hal ini akan mengizinkan semua anggota grup keamanan untuk mengakses klaster.

> **Note**
>
> Jika Anda berencana untuk menggunakan Zona Lokal, pastikan Anda telah mengaktifkannya. Saat Anda membuat grup subnet di zona lokal, VPC Anda diperluas ke Zona Lokal tersebut dan VPC Anda akan memperlakukan subnet itu seperti subnet lain di Zona Ketersediaan lainnya. Semua gateway dan tabel rute yang berkaitan akan disesuaikan secara otomatis.

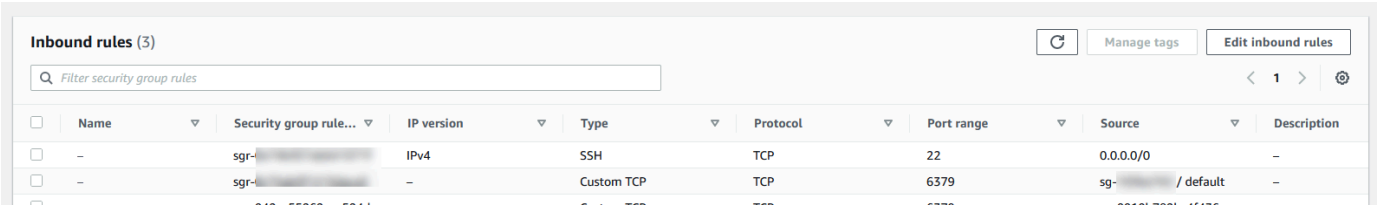
Untuk membuat aturan dalam grup keamanan VPC yang memungkinkan koneksi dari grup keamanan lain

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon VPC di https://console.aws.amazon.com/vpc.
2. Pada panel navigasi, pilih Grup Keamanan.
3. Pilih atau buat grup keamanan yang akan Anda gunakan untuk instans Klaster Anda. Di bawah Aturan Masuk, pilih Edit Aturan Masuk lalu pilih Tambahkan Aturan. Grup keamanan ini akan mengizinkan akses bagi anggota dari grup keamanan lain.
4. Dari Jenis, pilih Aturan TCP Khusus.

   a. Untuk Rentang Port, tentukan port yang Anda gunakan saat membuat klaster.

      Port default untuk klaster dan grup replikasi Redis adalah 6379.

   b. Pada kotak Sumber, masukkan ID dari grup keamanan. Dari daftar, pilih grup keamanan yang akan Anda gunakan untuk instans Amazon EC2 Anda.
5. Pilih Simpan jika selesai.



Setelah Anda mengaktifkan akses, Anda sekarang siap untuk menyambung ke simpul, yang dibahas pada bagian berikutnya.

Untuk informasi pengaksesan klaster ElastiCache Anda dari Amazon VPC yang berbeda, Wilayah AWS yang berbeda, atau bahkan jaringan perusahaan Anda, lihat hal berikut:

- Pola Akses untuk Mengakses Cache ElastiCache di Amazon VPC
- Mengakses sumber daya ElastiCache dari luar AWS

# Langkah 4: Menyambung ke simpul klaster

Sebelum melanjutkan, selesaikan Langkah 3: Mengizinkan akses ke klaster.

Bagian ini mengasumsikan bahwa Anda telah membuat instans Amazon EC2 dan dapat terhubung ke instans tersebut. Untuk petunjuk cara melakukannya, lihat Panduan Memulai Amazon EC2.

Instans Amazon EC2 dapat terhubung ke simpul klaster hanya setelah Anda memberikan otorisasi pada instans tersebut.

## Temukan titik akhir simpul Anda

Ketika klaster dalam status tersedia dan Anda telah memberikan otorisasi akses ke klaster, Anda dapat masuk ke instans Amazon EC2 dan terhubung ke klaster tersebut. Untuk melakukan itu, Anda perlu menentukan titik akhir terlebih dahulu.

Mencari Titik Akhir Klaster Redis (Mode Klaster Dinonaktifkan) (Konsol)

Jika klaster Redis (mode klaster dinonaktifkan) hanya memiliki satu simpul, maka titik akhir simpul tersebut digunakan untuk proses baca-tulis. Jika klaster memiliki beberapa simpul, terdapat tiga jenis titik akhir; titik akhir primer, titik akhir pembaca dan titik akhir simpul.

Titik akhir primer adalah nama DNS yang selalu dicocokkan ke simpul primer di klaster. Titik akhir primer tidak terpengaruh oleh perubahan klaster Anda, seperti promosi replika baca ke peran primer. Untuk aktivitas tulis, sebaiknya aplikasi Anda terhubung ke titik akhir primer.

Titik akhir pembaca akan membagi koneksi masuk ke titik akhir secara merata di antara semua replika baca di klaster ElastiCache for Redis. Faktor lain seperti saat aplikasi membuat koneksi atau cara aplikasi menggunakan atau menggunakan (ulang) koneksi akan menentukan distribusi lalu lintas. Titik akhir pembaca tetap mengikuti perubahan klaster dalam waktu nyata saat replika ditambahkan atau dihapus. Anda dapat menempatkan beberapa replika baca dari klaster ElastiCache for Redis pada Zona Ketersediaan (AZ) AWS yang berbeda untuk memastikan ketersediaan tinggi titik akhir pembaca.

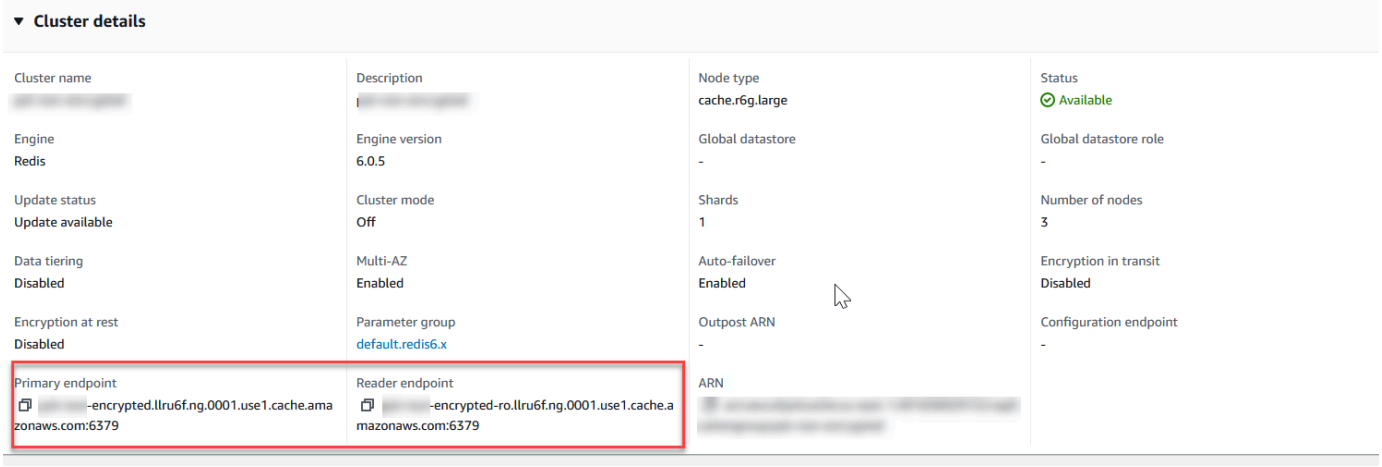
> **Note**
>
> Titik akhir pembaca bukan penyeimbang beban. Ini adalah catatan DNS yang akan diresolusi sebagai alamat IP dari salah satu simpul replika dengan metode round robin.

Untuk aktivitas baca, aplikasi juga dapat menghubungkan ke simpul mana pun di klaster. Tidak seperti titik akhir primer, titik akhir simpul diresolusi ke titik akhir tertentu. Jika Anda membuat perubahan di dalam klaster Anda, seperti menambahkan atau menghapus replika, Anda harus memperbarui titik akhir simpul di aplikasi Anda.

Menemukan titik akhir klaster Redis (mode klaster dinonaktifkan)

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Dari panel navigasi, pilih Cache Redis.

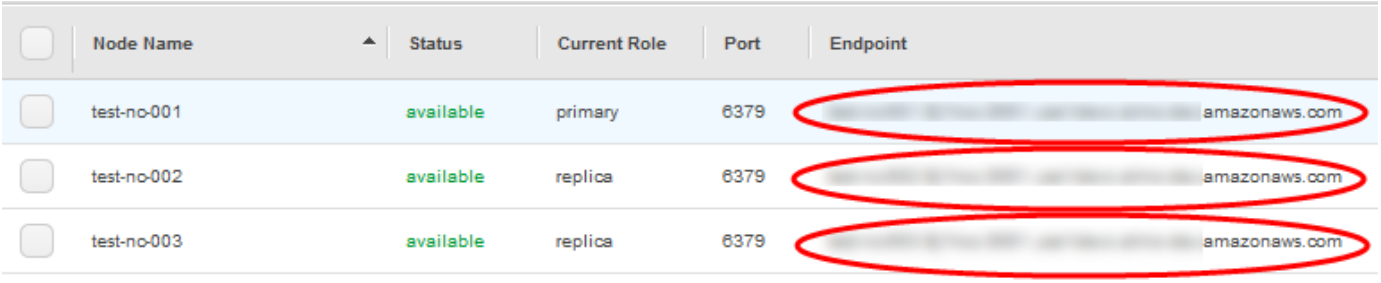
   Layar cluster akan muncul dengan daftar yang akan mencakup cache nirserver Redis yang ada, klaster Redis (mode klaster dinonaktifkan) dan Redis (mode klaster diaktifkan). Pilih klaster yang Anda buat di bagian Membuat klaster Redis (Mode Klaster Dinonaktifkan) (Konsol).
3. Untuk menemukan titik akhir Primer dan/atau Pembaca klaster, pilih nama klaster (bukan tombol radio).




   Titik akhir Primer dan Pembaca untuk klaster Redis (mode klaster dinonaktifkan)

   Jika hanya ada satu simpul dalam klaster, berarti tidak ada titik akhir primer dan Anda dapat melanjutkan ke langkah berikutnya.
4. Jika klaster Redis (mode klaster dinonaktifkan) memiliki simpul replika, Anda dapat menemukan titik akhir simpul replika klaster dengan memilih nama klaster lalu memilih tab Simpul.

   Layar simpul muncul dengan setiap simpul yang ada di klaster, primer dan replika, yang tercantum dengan titik akhirnya.




Titik akhir simpul untuk klaster Redis (mode klaster dinonaktifkan)

5. Untuk menyalin titik akhir ke clipboard Anda:

   a. Temukan satu per satu titik akhir yang ingin Anda salin.

   b. Pilih ikon salin langsung di depan titik akhir.

   Titik akhir sekarang disalin ke clipboard Anda. Untuk informasi tentang menggunakan titik akhir agar terhubung ke simpul, lihat Menghubungkan ke simpul.

Titik akhir primer Redis (mode klaster dinonaktifkan) terlihat seperti berikut ini. Ada perbedaan yang tergantung pada apakah enkripsi Bergerak aktif atau tidak.

Enkripsi bergerak tidak diaktifkan

```
clusterName.xxxxxx.nodeId.regionAndAz.cache.amazonaws.com:port

redis-01.7abc2d.0001.usw2.cache.amazonaws.com:6379
```

Enkripsi in-transit diaktifkan

```
master.clusterName.xxxxxx.regionAndAz.cache.amazonaws.com:port

master.ncit.ameaqx.use1.cache.amazonaws.com:6379
```

Untuk mengetahui lebih banyak cara menemukan titik akhir Anda, lihat topik yang relevan untuk tipe klaster dan mesin yang Anda jalankan.

- Menemukan titik akhir koneksi
- Menemukan Titik Akhir Klaster Redis (Mode Klaster Dinonaktifkan) (Konsol)—Anda memerlukan titik akhir Konfigurasi dari klaster.

- Menemukan Titik Akhir (AWS CLI)
- Menemukan Titik Akhir (API ElastiCache)

Menyambung ke klaster atau grup replikasi Redis (Linux)

Setelah memiliki titik akhir yang dibutuhkan, Anda dapat masuk ke instans EC2 dan menyambungkan ke klaster atau grup replikasi. Pada contoh berikut, Anda menggunakan utilitas redis-cli untuk menyambung ke klaster. Versi terbaru dari redis-cli juga mendukung SSL/TLS untuk menyambungkan klaster dengan dukungan enkripsi/autentikasi.

Contoh berikut menggunakan instans Amazon EC2 yang menjalankan Amazon Linux dan Amazon Linux 2. Untuk detail pemasangan dan kompilasi redis-cli dengan distribusi Linux lain, lihat dokumentasi untuk sistem operasi spesifik Anda.

> **Note**
>
> Proses ini mencakup pengujian koneksi menggunakan utilitas redis-cli untuk penggunaan yang tidak direncanakan saja. Untuk daftar klien yang mendukung Redis, lihat Dokumentasi Redis. Untuk contoh penggunaan SDK AWS dengan ElastiCache, lihat Mulai Menggunakan ElastiCache dan SDK AWS.

Menyambung ke klaster tanpa enkripsi dengan mode klaster nonaktif

1. Jalankan perintah berikut untuk menyambung ke klaster dan ganti *titik akhir primer* dan *nomor port* dengan titik akhir klaster dan nomor port Anda. (Port default untuk Redis adalah 6379.)

   ```
   src/redis-cli -h primary-endpoint -p port number
   ```

   Hasil pada prompt perintah Redis akan terlihat seperti berikut ini:

   ```
   primary-endpoint:port number
   ```

2. Sekarang Anda dapat menjalankan perintah Redis.

   ```
   set x Hello
   OK
   ```

```
get x
"Hello"
```

Menyambung ke klaster tanpa enkripsi dengan mode klaster aktif

1. Jalankan perintah berikut untuk menyambung ke klaster dan ganti *titik akhir konfigurasi* dan *nomor port* dengan titik akhir klaster dan nomor port Anda. (Port default untuk Redis adalah 6379.)

   ```
   src/redis-cli -h configuration-endpoint -c -p port number
   ```

   > **Note**
   >
   > Pada perintah sebelumnya, opsi -c memungkinkan mode klaster mengikuti pengalihan -ASK dan -MOVED.

   Hasil pada prompt perintah Redis akan terlihat seperti berikut ini:

   ```
   configuration-endpoint:port number
   ```

2. Sekarang Anda dapat menjalankan perintah Redis. Perhatikan bahwa pengalihan terjadi karena Anda mengaktifkannya menggunakan opsi -c. Jika pengalihan tidak diaktifkan, perintah akan menghasilkan kesalahan MOVED. Untuk informasi lain tentang kesalahan MOVED, lihat Spesifikasi klaster Redis.

   ```
   set x Hi
   -> Redirected to slot [16287] located at 172.31.28.122:6379
   OK
   set y Hello
   OK
   get y
   "Hello"
   set z Bye
   -> Redirected to slot [8157] located at 172.31.9.201:6379
   OK
   get z
   "Bye"
   get x
   ```

```
-> Redirected to slot [16287] located at 172.31.28.122:6379
"Hi"
```

Menyambung ke klaster dengan Enkripsi/Autentikasi aktif

Secara default, redis-cli menggunakan koneksi TCP tanpa enkripsi saat menyambung ke Redis. Opsi `BUILD_TLS=yes` mengaktifkan SSL/TLS pada saat kompilasi redis-cli seperti yang ditunjukkan pada bagian Unduh dan atur redis-cli sebelumnya. Mengaktifkan AUTH bersifat opsional. Namun, Anda harus mengaktifkan enkripsi in-transit untuk mengaktifkan AUTH. Untuk detail selengkapnya enkripsi dan autentikasi ElastiCache, lihat Enkripsi bergerak ElastiCache (TLS).

> **Note**
>
> Anda dapat menggunakan opsi `--tls` dengan redis-cli untuk terhubung ke klaster terenkripsi yang mengaktifkan mode klaster maupun yang tidak. Jika token AUTH pada klaster telah diatur, maka Anda dapat menggunakan opsi `-a` untuk menyediakan kata sandi AUTH.

Pada contoh berikut, pastikan untuk mengganti *`titik akhir klaster`* dan *`nomor port`* dengan titik akhir klaster dan nomor port Anda. (Port default untuk Redis adalah 6379.)

Menyambung ke klaster terenkripsi dengan mode klaster nonaktif

Contoh berikut menyambungkan ke klaster yang mengaktifkan enkripsi dan autentikasi:

```
src/redis-cli -h cluster-endpoint --tls -a your-password -p port number
```

Contoh berikut menyambung ke klaster yang hanya mengaktifkan enkripsi:

```
src/redis-cli -h cluster-endpoint --tls -p port number
```

Menyambungkan ke klaster terenkripsi dengan mode klaster aktif

Contoh berikut menyambungkan ke klaster yang mengaktifkan enkripsi dan autentikasi:

```
src/redis-cli -c -h cluster-endpoint --tls -a your-password -p port number
```

Contoh berikut menyambung ke klaster yang hanya mengaktifkan enkripsi:

```
src/redis-cli -c -h cluster-endpoint --tls -p port number
```

Setelah Anda tersambung ke klaster, Anda dapat menjalankan perintah Redis seperti yang ditunjukkan pada contoh sebelumnya untuk klaster tanpa enkripsi.

Altenatif redis-cli

Jika klaster bukan mode klaster aktif dan Anda perlu membuat sambungan ke klaster untuk uji singkat tanpa melalui kompilasi redis-cli, Anda dapat menggunakan telnet atau openssl. Pada contoh berikut, pastikan untuk mengganti *titik akhir klaster* dan *nomor port* dengan titik akhir klaster dan nomor port Anda. (Port default untuk Redis adalah 6379.)

Contoh berikut menyambungkan ke klaster (mode klaster nonaktif ) dengan enkripsi dan/atau autentikasi aktif:

```
openssl s_client -connect cluster-endpoint:port number
```

Jika kata sandi klaster telah ditetapkan, sambungkan ke klaster terlebih dahulu. Setelah tersambung, lakukan autentikasi pada klaster menggunakan perintah berikut, dan kemudian tekan tombol `Enter`. Pada contoh berikut, ganti *kata sandi Anda* dengan kata sandi untuk klaster Anda.

```
Auth your-password
```

Contoh berikut menyambungkan ke klaster (mode klaster nonaktif ) yang tidak memiliki enkripsi atau autentikasi aktif:

```
telnet cluster-endpoint port number
```

Menyambung ke klaster atau grup replikasi Redis (Windows)

Agar tersambung ke Klaster Redis dari instans EC2 Windows menggunakan Redis CLI, Anda harus mengunduh paket redis-cli dan menggunakan redis-cli.exe untuk menyambung ke Klaster Redis dari instans EC2 Windows.

Pada contoh berikut, Anda menggunakan utilitas redis-cli untuk menyambung ke klaster yang menjalankan Redis dan tidak mengaktifkan enkripsi. Untuk informasi selengkapnya mengenai Redis dan perintah Redis yang tersedia, lihat Perintah Redis di situs web Redis.

Untuk menyambung ke klaster Redis tanpa enkripsi aktif menggunakan redis-cli

1. Sambungkan ke instans Amazon EC2 Anda menggunakan utilitas koneksi pilihan Anda. Untuk petunjuk cara menghubungkan ke instans Amazon EC2, lihat Panduan Memulai Amazon EC2.
2. Salin dan tempel tautan https://github.com/microsoftarchive/redis/releases/download/win-3.0.504/Redis-x64-3.0.504.zip di peramban Internet untuk mengunduh file zip untuk klien Redis dari rilis yang tersedia di GitHub https://github.com/microsoftarchive/redis/releases/tag/win-3.0.504

   Ekstrak file zip tersebut ke folder/jalur yang Anda inginkan.

   Buka Prompt Perintah dan ubah ke direktori Redis dan jalankan perintah `c:\Redis>redis-cli -h Redis_Cluster_Endpoint -p 6379`.

   Contoh:

   ```
   c:\Redis>redis-cli -h cmd.xxxxxxx.ng.0001.usw2.cache.amazonaws.com -p 6379
   ```
3. Jalankan perintah Redis.

   Sekarang Anda tersambung ke klaster dan dapat menjalankan perintah Redis seperti berikut.

   ```
   set a "hello"          // Set key "a" with a string value and no expiration
   OK
   get a                  // Get value for key "a"
   "hello"
   get b                  // Get value for key "b" results in miss
   (nil)
   set b "Good-bye" EX 5  // Set key "b" with a string value and a 5 second expiration
   "Good-bye"
   get b                  // Get value for key "b"
   "Good-bye"
                          // wait >= 5 seconds
   get b
   (nil)                  // key has expired, nothing returned
   quit                   // Exit from redis-cli
   ```

## Langkah 5: Menghapus klaster

Selama klaster dalam status tersedia, Anda akan dikenakan biaya, terlepas dari apakah Anda secara aktif menggunakannya atau tidak. Untuk berhenti dikenakan biaya, hapus klaster tersebut.

> **Warning**
>
> Saat Anda menghapus klaster ElastiCache for Redis, snapshot manual Anda akan disimpan. Anda juga dapat membuat snapshot akhir sebelum klaster dihapus. Snapshot cache otomatis tidak dipertahankan. Untuk informasi selengkapnya, lihat Melakukan snapshot dan pemulihan.

Menggunakan AWS Management Console

Prosedur berikut menghapus satu klaster dari deployment Anda. Untuk menghapus beberapa klaster, ulangi prosedur untuk setiap klaster yang ingin dihapus. Anda tidak perlu menunggu satu klaster selesai dihapus sebelum memulai prosedur untuk menghapus klaster lain.

Untuk menghapus klaster

1. Masuk ke AWS Management Console dan buka konsol Amazon ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Pada dasbor konsol ElastiCache, pilih Redis.

   Daftar semua cache yang menjalankan Redis akan muncul.
3. Untuk memilih klaster yang akan dihapus, pilih nama klaster tersebut dari daftar klaster. Dalam kasus ini, nama klaster yang Anda buat di Langkah 2: Buat klaster.

   > **Important**
   >
   > Anda hanya dapat menghapus satu klaster saja pada satu waktu dari konsol ElastiCache. Memilih beberapa klaster akan menonaktifkan operasi hapus.
4. Untuk Tindakan, pilih Hapus.
5. Di layar Konfirmasi Hapus Klaster, ketikkan nama klaster dan pilih Cadangan Akhir. Kemudian pilih Hapus untuk menghapus klaster, atau pilih Batal untuk mempertahankan klaster.

   Jika Anda memilih Hapus, status klaster berubah menjadi menghapus.

Segera setelah klaster Anda tidak lagi tercantum di dalam daftar klaster, Anda berhenti dikenakan biaya untuk itu.

Menggunakan AWS CLI

Kode berikut menghapus klaster cache `my-cluster`. Dalam kasus ini, ganti `my-cluster` dengan nama klaster yang Anda buat di Langkah 2: Buat klaster.

```
aws elasticache delete-cache-cluster --cache-cluster-id my-cluster
```

Tindakan CLI `delete-cache-cluster` hanya menghapus satu klaster cache. Untuk menghapus beberapa klaster cache, panggil `delete-cache-cluster` untuk setiap klaster cache yang ingin dihapus. Anda tidak perlu menunggu satu klaster cache selesai dihapus untuk dapat menghapus yang lain.

Untuk Linux, macOS, atau Unix:

```
aws elasticache delete-cache-cluster \
    --cache-cluster-id my-cluster \
    --region us-east-2
```

Untuk Windows:

```
aws elasticache delete-cache-cluster ^
    --cache-cluster-id my-cluster ^
    --region us-east-2
```

Untuk informasi selengkapnya, lihat topik AWS CLI untuk ElastiCache `delete-cache-cluster`.

## Tutorial dan video ElastiCache

Tutorial berikut membahas tugas yang menarik bagi pengguna Amazon ElastiCache.

- Video ElastiCache
- Tutorial: Mengonfigurasi Fungsi Lambda untuk Mengakses Amazon ElastiCache di Amazon VPC

# Video ElastiCache

Pada bagian berikut ini, Anda dapat menemukan video untuk membantu Anda mempelajari konsep dasar dan lanjutan Amazon ElastiCache. Untuk informasi tentang Pelatihan AWS, lihat Pelatihan & Sertifikasi AWS.

Topik

- Video Pengantar
- Video Lanjutan

## Video Pengantar

Video berikut memberikan pengantar Amazon ElastiCache.

Topik

- AWS re:Invent 2020: Yang baru di Amazon ElastiCache
- AWS re:Invent 2019: Yang baru di Amazon ElastiCache
- AWS re:Invent 2017: Yang baru di Amazon ElastiCache
- DAT204—Membuat Aplikasi yang Dapat Diskalakan di Layanan NoSQL AWS (re:Invent 2015)
- DAT207—Mempercepat Performa Aplikasi dengan Amazon ElastiCache (AWS re:Invent 2013)

### AWS re:Invent 2020: Yang baru di Amazon ElastiCache

AWS re:Invent 2020: Yang baru di Amazon ElastiCache

### AWS re:Invent 2019: Yang baru di Amazon ElastiCache

AWS re:Invent 2019: Yang baru di Amazon ElastiCache

### AWS re:Invent 2017: Yang baru di Amazon ElastiCache

AWS re:Invent 2017: Yang baru di Amazon ElastiCache

### DAT204—Membuat Aplikasi yang Dapat Diskalakan di Layanan NoSQL AWS (re:Invent 2015)

Dalam sesi ini, kita membahas manfaat basis data NoSQL dan mempelajari layanan NoSQL utama yang ditawarkan oleh AWS—Amazon DynamoDB dan Amazon ElastiCache. Kemudian, kita

akan mendengar pendapat dari dua pelanggan terkemuka, Expedia dan Mapbox, tentang kasus penggunaan dan tantangan arsitektur mereka, dan cara mereka menanganinya menggunakan layanan AWS NoSQL, termasuk pola desain dan praktik terbaik. Setelah melalui sesi ini, Anda akan memiliki pemahaman yang lebih baik tentang NoSQL dan kemampuan NoSQL yang andal untuk bersiap mengatasi tantangan basis data Anda dengan percaya diri.

DAT204—Membuat Aplikasi yang Dapat Diskalakan di Layanan AWS NoSQL (re:Invent 2015)

DAT207—Mempercepat Performa Aplikasi dengan Amazon ElastiCache (AWS re:Invent 2013)

Dalam video ini, pelajari cara menggunakan Amazon ElastiCache untuk men-deploy sistem cache dalam memori dengan mudah untuk mempercepat performa aplikasi Anda. Kami menunjukkan cara menggunakan Amazon ElastiCache untuk meningkatkan latensi aplikasi dan mengurangi beban pada server basis data Anda. Kami juga akan menunjukkan cara membuat sebuah lapisan caching yang mudah dikelola dan diskalakan seiring pertumbuhan aplikasi Anda. Selama sesi ini, kita akan membahas berbagai skenario dan kasus penggunaan yang bermanfaat dengan mengaktifkan caching, serta mendiskusikan fitur yang disediakan oleh Amazon ElastiCache.

DAT207 - Mempercepat Performa Aplikasi dengan Amazon ElastiCache (re:Invent 2013)

Video Lanjutan

Video berikut membahas berbagai topik Amazon ElastiCache lanjutan.

Topik

- Desain untuk keberhasilan dengan praktik terbaik Amazon ElastiCache (re:Invent 2020)
- Memaksimalkan aplikasi waktu nyata dengan Amazon ElastiCache (Re:invent 2019)
- Praktik terbaik: memigrasikan klaster Redis dari Amazon EC2 ke ElastiCache (Re:invent 2019)
- Menskalakan Platform Olahraga Fantasi dengan Amazon ElastiCache & Amazon Aurora STP11 (re:invent 2018)
- Redis yang Andal & Dapat Diskalakan di Cloud dengan Amazon ElastiCache (re:invent 2018)
- Mempelajari Lebih Dalam ElastiCache: Pola desain untuk Penyimpanan Data Dalam Memori (re:invent 2018)
- DAT305—Mempelajari Lebih Dalam Amazon ElastiCache (re:Invent 2017)
- DAT306—Mempelajari Lebih Dalam Amazon ElastiCache (re:Invent 2016)
- DAT317—Bagaimana IFTTT menggunakan ElastiCache for Redis untuk Memprediksi Peristiwa (re:Invent 2016)

- DAT407—Mempelajari Lebih Dalam Amazon ElastiCache (re:Invent 2015)
- SDD402—Mempelajari Lebih Dalam Amazon ElastiCache (re:Invent 2014)
- DAT307—Mempelajari Lebih Dalam Arsitektur dan Pola Desain Amazon ElastiCache (re:Invent 2013)

Desain untuk keberhasilan dengan praktik terbaik Amazon ElastiCache (re:Invent 2020)

Seiring dengan pesatnya pertumbuhan aplikasi waktu nyata dan penting untuk bisnis yang dibangun di Redis, ketersediaan, skalabilitas, dan keamanan telah menjadi pertimbangan utama. Pelajari praktik terbaik untuk menyiapkan Amazon ElastiCache agar Anda berhasil dengan penskalaan online, ketersediaan tinggi di seluruh deployment Multi-AZ, dan konfigurasi keamanan.

Desain untuk keberhasilan dengan praktik terbaik Amazon ElastiCache (re:Invent 2020)

Memaksimalkan aplikasi waktu nyata dengan Amazon ElastiCache (Re:invent 2019)

Dengan pertumbuhan pesat dalam adopsi cloud dan skenario baru yang diberdayakannya, aplikasi memerlukan latensi mikrodetik dan throughput yang tinggi untuk mendukung jutaan permintaan per detik. Developer secara tradisional mengandalkan perangkat keras dan solusi khusus, seperti basis data berbasis disk yang dikombinasikan dengan teknik pengurangan data, untuk mengelola data untuk aplikasi waktu nyata. Pendekatan ini dapat menghabiskan banyak biaya dan tidak dapat diskalakan. Pelajari bagaimana Anda dapat meningkatkan kinerja aplikasi waktu nyata dengan menggunakan Amazon ElastiCache dalam memori yang dikelola sepenuhnya untuk kinerja ekstrem, skalabilitas tinggi, ketersediaan, dan keamanan.

Memaksimalkan aplikasi waktu nyata dengan Amazon ElastiCache (Re:invent 2019:)

Praktik terbaik: memigrasikan klaster Redis dari Amazon EC2 ke ElastiCache (Re:invent 2019)

Mengelola klaster Redis sendiri tidaklah mudah. Anda harus menyediakan perangkat keras, patch perangkat lunak, melakukan backup data, dan memantau beban kerja secara konstan. Dengan fitur Migrasi Online yang baru dirilis untuk Amazon ElastiCache, Anda kini dapat dengan mudah memindahkan data Anda dari Redis yang dihosting sendiri di Amazon EC2 ke Amazon ElastiCache yang terkelola sepenuhnya, dengan mode menonaktifkan klaster. Dalam sesi ini, Anda mempelajari alat Migrasi Online baru, melihat demo, dan, yang lebih penting, mempelajari praktik terbaik secara langsung untuk migrasi yang mulus ke Amazon ElastiCache.

Praktik terbaik: memigrasikan klaster Redis dari Amazon EC2 ke ElastiCache (Re:invent 2019)

Menskalakan Platform Olahraga Fantasi dengan Amazon ElastiCache & Amazon Aurora STP11 (re:invent 2018)

Dream11 adalah perusahaan rintisan teknologi olahraga terkemuka di India. Perusahaan ini memiliki basis pengguna yang berkembang dari 40 juta lebih yang bermain beberapa olahraga, termasuk kriket, sepak bola, dan bola basket fantasi, dan saat ini melayani satu juta pengguna secara serempak, yang menghasilkan tiga juta permintaan per menit dengan waktu respon di bawah 50 milidetik. Dalam diskusi ini, CTO Dream11, Amit Sharma, menjelaskan bagaimana perusahaan menggunakan Amazon Aurora dan Amazon ElastiCache untuk menangani lalu lintas flash, yang dapat menjadi tiga kali lipat dalam jendela waktu respons 30 detik. Sharma juga berbicara tentang penskalaan transaksi tanpa penguncian, dan dia berbagi langkah-langkah untuk menangani lalu lintas flash - hingga melayani lima juta pengguna aktif setiap hari. Judul Lengkap: AWS re:invent 2018: Menskalakan Platform Olahraga Fantasi dengan Amazon ElastiCache & Amazon Aurora (STP11)

Menskalakan Platform Olahraga Fantasi dengan Amazon ElastiCache & Amazon Aurora STP11 (re:invent 2018)

Redis yang Andal & Dapat Diskalakan di Cloud dengan Amazon ElastiCache (re:invent 2018)

Sesi ini membahas fitur dan peningkatan dalam layanan kami yang kompatibel dengan Redis, yaitu Amazon ElastiCache for Redis. Kami membahas fitur utama, seperti Redis 5, skalabilitas serta peningkatan kinerja, keamanan dan kepatuhan, dan banyak lagi. Kami juga membahas fitur yang akan datang dan studi kasus pelanggan.

Redis yang Andal & Dapat Diskalakan di Cloud dengan Amazon ElastiCache (re:invent 2018)

Mempelajari Lebih Dalam ElastiCache: Pola desain untuk Penyimpanan Data Dalam Memori (re:invent 2018)

Dalam sesi ini, kami menyediakan informasi di belakang layar untuk belajar tentang desain dan arsitektur dari Amazon ElastiCache. Lihat pola desain umum dengan penawaran Redis dan Memcached kami dan cara pelanggan menggunakannya untuk pemrosesan data dalam memori untuk mengurangi latensi dan meningkatkan throughput aplikasi. Kami meninjau praktik terbaik, pola desain, dan anti-pola dari ElastiCache.

Mempelajari Lebih Dalam ElastiCache: Pola desain untuk Penyimpanan Data Dalam Memori (re:invent 2018)

DAT305—Mempelajari Lebih Dalam Amazon ElastiCache (re:Invent 2017)

Lihat di balik layar untuk mempelajari tentang desain dan arsitektur Amazon ElastiCache. Lihat pola desain umum dengan penawaran Redis dan Memcached kami dan cara pelanggan menggunakannya untuk pemrosesan data dalam memori untuk mengurangi latensi dan meningkatkan throughput aplikasi. Dalam video ini, kami meninjau praktik terbaik, pola desain, dan anti-pola dari ElastiCache.

Video ini memperkenalkan hal berikut:

- Resharding online ElastiCache for Redis
- Keamanan dan enkripsi ElastiCache
- ElastiCache for Redis versi 3.2.10

DAT305—Mempelajari Lebih Dalam Amazon ElastiCache (re:Invent 2017)

DAT306—Mempelajari Lebih Dalam Amazon ElastiCache (re:Invent 2016)

Lihat di balik layar untuk mempelajari tentang desain dan arsitektur Amazon ElastiCache. Lihat pola desain umum dengan penawaran Redis dan Memcached kami dan cara pelanggan menggunakannya untuk pemrosesan data dalam memori untuk mengurangi latensi dan meningkatkan throughput aplikasi. Selama sesi ini, kami meninjau praktik terbaik, pola desain, dan anti-pola dari ElastiCache.

DAT306—Mempelajari Lebih Dalam Amazon ElastiCache (re:Invent 2016)

DAT317—Bagaimana IFTTT menggunakan ElastiCache for Redis untuk Memprediksi Peristiwa (re:Invent 2016)

IFTTT adalah layanan gratis yang memberdayakan orang untuk berbuat lebih banyak dengan layanan yang mereka sukai, mulai dari automasi tugas sederhana hingga mengubah cara seseorang berinteraksi dan mengendalikan rumah mereka. IFTTT menggunakan ElastiCache for Redis untuk menyimpan riwayat proses transaksi dan menjadwalkan prediksi serta indeks untuk dokumen log di Amazon S3. Lihat sesi ini untuk mempelajari bagaimana kekuatan scripting dari Lua dan jenis data Redis memungkinkan orang untuk mencapai sesuatu yang tidak dapat dilakukan dengan sistem lain.

DAT317—Bagaimana IFTTT menggunakan ElastiCache for Redis untuk Memprediksi Peristiwa (re:Invent 2016)

DAT407—Mempelajari Lebih Dalam Amazon ElastiCache (re:Invent 2015)

Lihat di balik layar untuk mempelajari tentang desain dan arsitektur Amazon ElastiCache. Lihat pola desain umum dengan penawaran Redis dan Memcached kami dan cara pelanggan menggunakannya untuk operasi dalam memori dan mencapai peningkatan latensi dan throughput untuk aplikasi. Selama sesi ini, kami meninjau praktik terbaik, pola desain, dan anti-pola dari Amazon ElastiCache.

DAT407—Mempelajari Lebih Dalam Amazon ElastiCache (re:Invent 2015)

SDD402—Mempelajari Lebih Dalam Amazon ElastiCache (re:Invent 2014)

Dalam video ini, kita memeriksa kasus penggunaan caching umum, mesin Memcached dan Redis, pola yang membantu Anda menentukan mesin yang lebih baik untuk kebutuhan Anda, hashing konsisten, dan lainnya sebagai sarana untuk membangun aplikasi yang cepat dan dapat diskalakan. Frank Wiebe, Ilmuwan Utama di Adobe, merinci cara Adobe menggunakan Amazon ElastiCache untuk meningkatkan pengalaman pelanggan dan menskalakan bisnis mereka.

DAT402—Mempelajari Lebih Dalam Amazon ElastiCache (re:Invent 2014)

DAT307—Mempelajari Lebih Dalam Arsitektur dan Pola Desain Amazon ElastiCache (re:Invent 2013)

Dalam video ini, kita memeriksa caching, strategi caching, penskalaan ke luar, dan pemantauan. Kita juga membandingkan mesin Memcached dan Redis. Selama sesi ini, kami meninjau praktik terbaik, pola desain, dan anti-pola dari Amazon ElastiCache.

DAT307—Mempelajari Lebih Dalam Arsitektur dan Pola Desain Amazon ElastiCache (AWS re:Invent 2013).

## Apa yang saya lakukan selanjutnya?

Setelah mencoba latihan Memulai, Anda dapat menjelajahi bagian berikut untuk mempelajari lebih lanjut ElastiCache dan alat yang tersedia:

- Memulai dengan AWS
- Alat untuk Amazon Web Services
- AWS Command Line Interface
- Referensi API Amazon ElastiCache

Setelah menyelesaikan latihan Memulai, Anda dapat membaca bagian ini untuk mempelajari selengkapnya administrasi ElastiCache:

- Memilih ukuran simpul Anda

  Anda menginginkan cache Anda cukup besar untuk mengakomodasi semua data yang ingin dijadikan cache. Pada saat yang sama, Anda tidak ingin mengeluarkan biaya lebih dari kebutuhan cache Anda. Gunakan topik ini untuk membantu Anda memilih ukuran simpul yang terbaik.
- Praktik terbaik ElastiCache dan strategi caching

  Identifikasi dan atasi masalah yang dapat memengaruhi efisiensi klaster Anda.

# Mengelola simpul

Simpul adalah komponen penyusun terkecil deployment Amazon ElastiCache. Simpul adalah potongan RAM berukuran tetap yang aman dan terpasang ke jaringan. Setiap simpul menjalankan mesin yang dipilih ketika klaster atau grup replikasi dibuat atau terakhir diubah. Setiap simpul mempunyai nama dan port Layanan Nama Domain (DNS) sendiri. Beberapa tipe simpul ElastiCache didukung, masing-masing dengan jumlah memori terkait dan daya komputasi yang bervariasi.

Secara umum, karena dukungan untuk serpihan, deployment Redis (mode klaster diaktifkan) memiliki sejumlah simpul yang lebih kecil. Sebaliknya, deployment Redis (mode klaster dinonaktifkan) memiliki simpul lebih sedikit tetapi ukurannya lebih besar di klaster. Untuk diskusi yang lebih terperinci tentang ukuran simpul yang akan digunakan, lihat Memilih ukuran simpul Anda.

Topik

- Melihat Status ElastiCache Node
- Simpul dan serpihan Redis
- Menghubungkan ke simpul
- Tipe simpul yang didukung
- Mem-boot ulang simpul (mode klaster dinonaktifkan saja)
- Mengganti simpul
- Simpul terpesan ElastiCache
- Memigrasikan simpul generasi sebelumnya

Beberapa operasi penting yang melibatkan simpul adalah sebagai berikut:

- Menambahkan simpul ke klaster
- Menghapus simpul dari klaster
- Penskalaan ElastiCache untuk Redis
- Menemukan titik akhir koneksi

## Melihat Status ElastiCache Node

Menggunakan ElastiCache konsol, Anda dapat dengan cepat mengakses status ElastiCache node Anda. Status ElastiCache node menunjukkan kesehatan node. Anda dapat menggunakan prosedur berikut untuk melihat status ElastiCache node di ElastiCache konsol Amazon, AWS CLI perintah, atau operasi API.

Nilai status yang mungkin untuk ElastiCache node ada di tabel berikut. Tabel ini juga menunjukkan apakah Anda akan ditagih untuk ElastiCache node.

| Jenis | Dikenakan Biaya | Deskripsi |
| --- | --- | --- |
| `available` | Ditagih | ElastiCache Node sehat dan tersedia. |
| `creating` | Tidak ditagih | ElastiCache Node sedang dibuat. Simpul tidak dapat diakses saat sedang dibuat. |
| `deleting` | Tidak ditagih | ElastiCache Node sedang dihapus. |
| `modifying` | Ditagih | ElastiCache Node sedang dimodifikasi karena permintaan pelanggan untuk memodifikasi node. |
| `updating` | Ditagih | Status Memperbarui menunjukkan satu atau beberapa hal berikut ini benar dari ElastiCache simpul Amazon: |

| Jenis | Dikenakan Biaya | Deskripsi |
| --- | --- | --- |
| | | • ElastiCache Node sedang ditambal sebagai bagian dari udpate layanan. Untuk informasi selengkap nya tentang pembaruan layanan, lihat Halaman Bantuan Pemeliharaan ElastiCache Terkelola Amazon dan Pembaruan Layanan.<br>• Grup keamanan VPC memperbarui untuk Cluster. ElastiCache<br>• ElastiCache Cluster sedang ditingkatkan atau diperkecil.<br>• Konfigurasi pengiriman log sedang dimodifikasi untuk ElastiCache Cluster.<br>• Operasi penghapusan untuk ElastiCache node sedang tertunda.<br>• Kata sandi ElastiCache for Redis sedang diperbarui/ diputar menggunakan. AWS Secrets Manager |
| `rebooting cache cluster nodes` | Ditagih | ElastiCache Node sedang di-boot ulang karena permintaa n pelanggan atau ElastiCac he proses Amazon yang memerlukan reboot node. |

| Jenis | Dikenakan Biaya | Deskripsi |
| --- | --- | --- |
| `incompatible_parameters` | Tidak ditagih | Amazon tidak ElastiCache dapat memulai node karena parameter yang ditentukan dalam grup parameter node tidak kompatibel dengan node. Kembalikan perubahan parameter atau buat parameter kompatibel dengan simpul untuk mendapatkan akses ke simpul Anda. Untuk informasi selengkapnya tentang parameter yang tidak kompatibel, periksa daftar Peristiwa untuk ElastiCache node. |
| `incompatible_network` | Tidak ditagih | Status jaringan yang tidak kompatibel menunjukkan satu atau beberapa hal berikut berlaku untuk node Amazon: ElastiCache<br>• Tidak ada alamat IP yang tersedia di subnet tempat ElastiCache node diluncurkan.<br>• Subnet yang disebutkan dalam grup ElastiCache subnet tidak lagi ada di Amazon Virtual Private Cloud (Amazon VPC). |

| Jenis | Dikenakan Biaya | Deskripsi |
| --- | --- | --- |
| `restore_failed` | Tidak ditagih | Status gagal pemulihan menunjukkan salah satu dari berikut ini benar dari simpul Amazon: ElastiCache<br>• Penggantian simpul gagal karena kapasitas instans tidak mencukupi berulang kali. Ini biasanya terjadi ketika menjalankan node generasi sebelumnya yang end-of-life. Namun, itu juga bisa terjadi dengan penggantian node generasi saat ini ketika AWS tidak memiliki kapasitas sesuai permintaan yang cukup untuk memenuhi permintaan Anda di Availability Zone yang ditentukan. Untuk informasi lebih lanjut tentang memperbaiki atau menghapus node ini, lihatMemigrasikan simpul generasi sebelumnya.<br>• Snapshot RDB yang ditentukan gagal dipulihkan.<br>• AWS Akun untuk ElastiCache cluster telah ditangguhkan.<br>• Node gagal dan tidak dapat dipulihkan. |

| Jenis | Dikenakan Biaya | Deskripsi |
| --- | --- | --- |
| `snapshotting` | Ditagih | ElastiCache sedang membuat snapshot dari node ElastiCache for Redis. |

### Melihat Status ElastiCache Node dengan konsol

Untuk melihat status ElastiCache Node dengan konsol:

1. Masuk ke AWS Management Console dan buka ElastiCache konsol Amazon di https://console.aws.amazon.com/elasticache/.
2. Di panel navigasi, pilih Redis Clusters atau Memcached Clusters. Halaman Cache muncul dengan daftar ElastiCache Node. Untuk setiap simpul, nilai status akan ditampilkan.
3. Anda kemudian dapat menavigasi ke tab Pembaruan Layanan untuk cache untuk menampilkan daftar Pembaruan layanan yang berlaku untuk cache.

### Melihat Status ElastiCache Node dengan AWS CLI

Untuk melihat ElastiCache node dan informasi statusnya dengan menggunakan AWS CLI, gunakan `describe-cache-cluster` perintah. Misalnya, AWS CLI perintah berikut menampilkan setiap ElastiCache node.

```
aws elasticache describe-cache-clusters
```

### Melihat Status ElastiCache Node melalui API

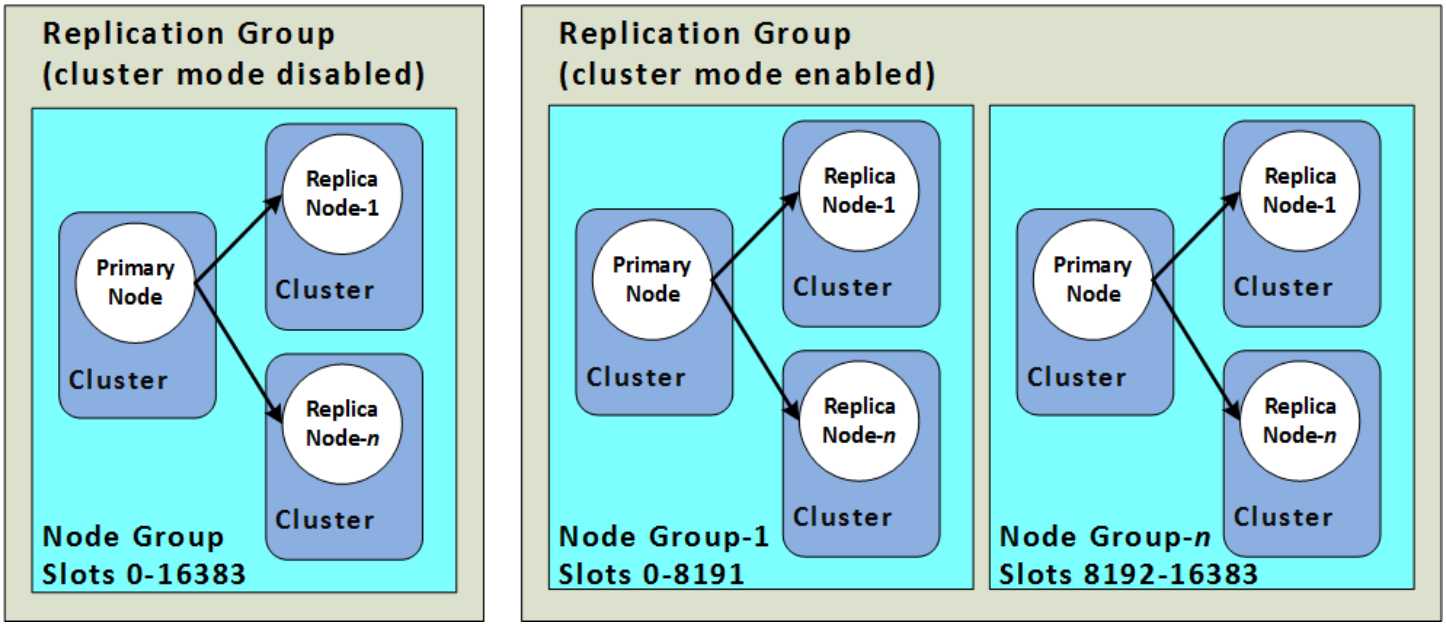
Untuk melihat status ElastiCache node menggunakan Amazon ElastiCache API, panggil `ShowCacheNodeInfo` flag `DescribeCacheClusteroperation` with untuk mengambil informasi tentang node cache individual.

## Simpul dan serpihan Redis

Serpihan (dalam API dan CLI, grup simpul) adalah pengaturan hirarkis simpul, masing-masing dibungkus di dalam klaster. Serpihan mendukung replikasi. Di dalam serpihan, satu simpul berfungsi sebagai simpul primer baca/tulis. Semua simpul lain dalam serpihan berfungsi sebagai replika baca-saja dari simpul primer. Redis versi 3.2 dan yang lebih baru mendukung beberapa serpihan dalam

klaster (dalam API dan CLI, grup replikasi). Dukungan ini memungkinkan pembuatan partisi data Anda di dalam klaster Redis (mode klaster diaktifkan).

Diagram berikut menggambarkan perbedaan antara klaster Redis (mode klaster dinonaktifkan) dan klaster Redis (modus klaster diaktifkan).




Klaster Redis (mode klaster diaktifkan) mendukung replikasi melalui serpihan. Operasi API DescribeReplicationGroups (CLI: describe-replication-groups) mencantumkan grup simpul dengan simpul anggota, peran simpul dalam grup simpul, dan juga informasi lainnya.

Saat Anda membuat klaster Redis, tentukan apakah Anda ingin membuat klaster dengan mengaktifkan pengklasteran. Klaster Redis (mode klaster dinonaktifkan) tidak pernah memiliki lebih dari satu serpihan, yang dapat diskalakan secara horizontal dengan menambahkan (hingga total lima) atau menghapus simpul replika baca. Untuk informasi selengkapnya, lihat Ketersediaan tinggi menggunakan grup replikasi, Menambahkan replika baca, untuk grup replikasi Redis (Mode Klaster Dinonaktifkan), atau Menghapus replika baca, untuk grup replikasi Redis (Mode Klaster Dinonaktifkan) . Klaster Redis (mode klaster dinonaktifkan) juga dapat diskalakan secara vertikal dengan mengubah jenis simpul. Untuk informasi selengkapnya, lihat Penskalaan Redis (Mode Cluster Dinonaktifkan) cluster dengan simpul replika.

Batas simpul atau serpihan dapat ditingkatkan hingga maksimum 500 per klaster jika versi mesin Redis yang digunakan adalah versi 5.0.6 atau lebih tinggi. Sebagai contoh, Anda dapat memilih untuk membuat konfigurasi dari sebuah klaster dengan 500 simpul yang berkisar antara 83 serpihan (satu primer dan 5 replika per serpihan) dan 500 serpihan (primer tunggal dan tidak ada replika). Pastikan alamat IP yang tersedia mencukupi untuk mengakomodasi peningkatan tersebut. Kesalahan umum

termasuk subnet dalam grup subnet memiliki rentang CIDR yang terlalu kecil atau subnet digunakan bersama dan banyak digunakan oleh klaster lain. Untuk informasi selengkapnya, lihat Membuat grup subnet.

Untuk versi di bawah 5.0.6, batasnya adalah 250 per klaster.

Untuk meminta penambahan batas, lihat Batas Layanan AWS dan pilih jenis batas Simpul per klaster per jenis instans.

Setelah klaster Redis (mode klaster diaktifkan) dibuat, ia dapat diubah (diskalakan masuk atau keluar). Untuk informasi selengkapnya, lihat Penskalaan ElastiCache untuk Redis dan Mengganti simpul.

Ketika Anda membuat klaster baru, Anda dapat menyemai klaster itu dengan data dari klaster lama sehingga klaster baru tidak mulai dari kosong. Pendekatan ini bekerja hanya jika grup klaster memiliki jumlah serpihan yang sama dengan klaster lama. Melakukan hal ini dapat membantu jika Anda perlu mengubah jenis simpul atau versi mesin. Untuk informasi selengkapnya, lihat Mengambil cadangan manual dan Melakukan pemulihan dari cadangan ke dalam cache baru.

# Menghubungkan ke simpul

Sebelum mencoba untuk terhubung ke klaster Redis, Anda harus memiliki titik akhir untuk simpul. Untuk menemukan titik akhir, lihat yang berikut ini:

- Menemukan Titik Akhir Klaster Redis (Mode Klaster Dinonaktifkan) (Konsol)
- Menemukan Titik Akhir Klaster Redis (Mode Klaster Dinonaktifkan) (Konsol)
- Menemukan Titik Akhir (AWS CLI)
- Menemukan Titik Akhir (API ElastiCache)

Pada contoh berikut, Anda menggunakan utilitas redis-cli untuk terhubung ke klaster yang menjalankan Redis.

> **Note**
>
> Untuk informasi selengkapnya tentang Redis dan perintah Redis yang tersedia, lihat situs web http://redis.io/commands.

Untuk terhubung ke klaster Redis menggunakan redis-cli

1. Hubungkan ke instans Amazon EC2 Anda menggunakan utilitas koneksi pilihan Anda.

   > **Note**
   >
   > Untuk petunjuk tentang cara menghubungkan ke instans Amazon EC2, lihat Panduan Memulai Amazon EC2.

2. Untuk membangun `redis-cli`, unduh dan instal GNU Compiler Collection (`gcc`). Pada prompt perintah instans EC2 Anda, masukkan perintah berikut dan ketik y pada prompt konfirmasi.

   ```
   sudo yum install gcc
   ```

   Muncul output seperti yang berikut ini.

   ```
   Loaded plugins: priorities, security, update-motd, upgrade-helper
   Setting up Install Process
   Resolving Dependencies
   ```

```
--> Running transaction check

...(output omitted)...

Total download size: 27 M
Installed size: 53 M
Is this ok [y/N]: y
Downloading Packages:
(1/11): binutils-2.22.52.0.1-10.36.amzn1.x86_64.rpm      | 5.2 MB     00:00
(2/11): cpp46-4.6.3-2.67.amzn1.x86_64.rpm                | 4.8 MB     00:00
(3/11): gcc-4.6.3-3.10.amzn1.noarch.rpm                  | 2.8 kB     00:00

...(output omitted)...

Complete!
```

3. Unduh dan kompilasi utilitas redis-cli. Utilitas ini disertakan dalam distribusi perangkat lunak Redis. Pada prompt perintah instans EC2 Anda, ketik perintah berikut:

> **Note**
>
> Untuk sistem Ubuntu, sebelum menjalankan `make`, jalankan `make distclean`.

```
wget http://download.redis.io/redis-stable.tar.gz
tar xvzf redis-stable.tar.gz
cd redis-stable
make distclean      # ubuntu systems only
make
```

4. Pada prompt perintah instans EC2 Anda, ketik perintah berikut.

```
src/redis-cli -c -h mycachecluster.eaogs8.0001.usw2.cache.amazonaws.com -p 6379
```

Muncul prompt perintah Redis seperti yang berikut ini.

```
redis mycachecluster.eaogs8.0001.usw2.cache.amazonaws.com 6379>
```

5. Uji koneksi dengan menjalankan perintah Redis.

Sekarang Anda terhubung ke klaster dan dapat menjalankan perintah Redis. Berikut adalah beberapa contoh perintah dengan respons dari Redis.

```
set a "hello"          // Set key "a" with a string value and no expiration
OK
get a                  // Get value for key "a"
"hello"
get b                  // Get value for key "b" results in miss
(nil)
set b "Good-bye" EX 5  // Set key "b" with a string value and a 5 second expiration
get b
"Good-bye"
                   // wait 5 seconds
get b
(nil)                  // key has expired, nothing returned
quit                   // Exit from redis-cli
```

Untuk terhubung ke simpul atau klaster yang memiliki enkripsi Lapisan Soket Aman (SSL) (enkripsi bergerak aktif), lihat Enkripsi bergerak ElastiCache (TLS).

# Tipe simpul yang didukung

ElastiCache mendukung jenis simpul berikut. Secara umum, tipe generasi saat ini memberikan lebih banyak memori dan daya komputasi dengan biaya lebih rendah dibandingkan dengan tipe generasi sebelumnya yang setara.

Untuk informasi selengkapnya tentang detail performa untuk setiap tipe simpul, lihat Tipe Instans Amazon EC2.

Untuk informasi tentang ukuran simpul yang akan digunakan, lihat Memilih ukuran simpul Anda.

## Generasi Saat Ini

Untuk informasi lebih lanjut tentang Generasi Sebelumnya, harap baca Simpul Generasi Sebelumnya.

> **Note**
>
> Tipe instans dengan peforma jaringan yang dapat dilonjakkan menggunakan mekanisme kredit I/O jaringan untuk melampaui bandwidth dasar mereka dengan upaya terbaik.

Umum

| Jenis instans | Versi Redis minimum yang didukung | I/O ditingkatkan (Redis 5.0.6 +) | TLS Offloading (Redis 6.2.5+) | Peningkatan I/O Multiplexing (Redis 7.0.4+) | Bandwidth dasar (Gbps) | Bandwidth burst (Gbps) |
|---|---|---|---|---|---|---|
| cache.m7g.large | 6,2 | T | T | T | 0,937 | 12,5 |
| cache.m7g.xlarge | 6,2 | Y | Y | Y | 1,876 | 12,5 |
| cache.m7g.2xlarge | 6,2 | Y | Y | Y | 3,75 | 15 |

| Jenis instans | Versi Redis minimum yang didukung | I/O ditingkatkan (Redis 5.0.6 +) | TLS Offloading (Redis 6.2.5+) | Peningkatan I/O Multiplexing (Redis 7.0.4+) | Bandwidth dasar (Gbps) | Bandwidth burst (Gbps) |
|---|---|---|---|---|---|---|
| cache.m7g.4xlarge | 6,2 | Y | Y | Y | 7,5 | 15 |
| cache.m7g.8xlarge | 6,2 | Y | Y | Y | 15 | T/A |
| cache.m7g.12xlarge | 6,2 | Y | Y | Y | 22,5 | T/A |
| cache.m7g.16xlarge | 6,2 | Y | Y | Y | 30 | T/A |
| cache.m6g.large | 5.0.6 | T | T | T | 0,75 | 10,0 |
| cache.m6g.xlarge | 5.0.6 | Y | Y | Y | 1,25 | 10,0 |
| cache.m6g.2xlarge | 5.0.6 | Y | Y | Y | 2,5 | 10,0 |
| cache.m6g.4xlarge | 5.0.6 | Y | Y | Y | 5,0 | 10,0 |
| cache.m6g.8xlarge | 5.0.6 | Y | Y | Y | 12 | T/A |
| cache.m6g.12xlarge | 5.0.6 | Y | Y | Y | 20 | T/A |
| cache.m6g.16xlarge | 5.0.6 | Y | Y | Y | 25 | T/A |

| Jenis instans | Versi Redis minimum yang didukung | I/O ditingkatkan (Redis 5.0.6 +) | TLS Offloading (Redis 6.2.5+) | Peningkatan I/O Multiplexing (Redis 7.0.4+) | Bandwidth dasar (Gbps) | Bandwidth burst (Gbps) |
|---|---|---|---|---|---|---|
| cache.m5.large | 3.2.4 | T | T | T | 0,75 | 10,0 |
| cache.m5.xlarge | 3.2.4 | Y | T | T | 1,25 | 10,0 |
| cache.m5.2xlarge | 3.2.4 | Y | Y | Y | 2,5 | 10,0 |
| cache.m5.4xlarge | 3.2.4 | Y | Y | Y | 5,0 | 10,0 |
| cache.m5.12xlarge | 3.2.4 | Y | Y | Y | 12 | T/A |
| cache.m5.24xlarge | 3.2.4 | Y | Y | Y | 25 | T/A |
| cache.m4.large | 3.2.4 | T | T | T | 0,45 | 1,2 |
| cache.m4.xlarge | 3.2.4 | Y | T | T | 0,75 | 2,8 |
| cache.m4.2xlarge | 3.2.4 | Y | Y | Y | 1,0 | 10,0 |
| cache.m4.4xlarge | 3.2.4 | Y | Y | Y | 2,0 | 10,0 |
| cache.m4.10xlarge | 3.2.4 | Y | Y | Y | 5,0 | 10,0 |
| cache.t4g.micro | 3.2.4 | T | T | T | 0,064 | 5,0 |
| cache.t4g.small | 5.0.6 | T | T | T | 0,128 | 5,0 |

| Jenis instans | Versi Redis minimum yang didukung | I/O ditingkatkan (Redis 5.0.6 +) | TLS Offloading (Redis 6.2.5+) | Peningkatan I/O Multiplexing (Redis 7.0.4+) | Bandwidth dasar (Gbps) | Bandwidth burst (Gbps) |
|---|---|---|---|---|---|---|
| cache.t4g.medium | 5.0.6 | T | T | T | 0,256 | 5,0 |
| cache.t3.micro | 3.2.4 | T | T | T | 0,064 | 5,0 |
| cache.t3.small | 3.2.4 | T | T | T | 0,128 | 5,0 |
| cache.t3.medium | 3.2.4 | T | T | T | 0,256 | 5,0 |
| cache.t2.micro | 3.2.4 | T | T | T | 0,064 | 1,024 |
| cache.t2.small | 3.2.4 | T | T | T | 0,128 | 1,024 |
| cache.t2.medium | 3.2.4 | T | T | T | 0,256 | 1,024 |

Memori yang dioptimalkan

| Jenis instans | Versi Redis minimum yang didukung | I/O ditingkatkan (Redis 5.0.6 +) | TLS Offloading (Redis 6.2.5+) | Peningkatan I/O Multiplexing (Redis 7.0.4+) | Bandwidth dasar (Gbps) | Bandwidth burst (Gbps) |
|---|---|---|---|---|---|---|
| cache.r7g.large | 6,2 | T | T | T | 0,937 | 12,5 |
| cache.r7g.xlarge | 6,2 | Y | Y | Y | 1,876 | 12,5 |

| Jenis instans | Versi Redis minimum yang didukung | I/O ditingkatkan (Redis 5.0.6 +) | TLS Offloading (Redis 6.2.5+) | Peningkatan I/O Multiplexing (Redis 7.0.4+) | Bandwidth dasar (Gbps) | Bandwidth burst (Gbps) |
|---|---|---|---|---|---|---|
| cache.r7g.2xlarge | 6,2 | Y | Y | Y | 3,75 | 15 |
| cache.r7g.4xlarge | 6,2 | Y | Y | Y | 7,5 | 15 |
| cache.r7g.8xlarge | 6,2 | Y | Y | Y | 15 | T/A |
| cache.r7g.12xlarge | 6,2 | Y | Y | Y | 22,5 | T/A |
| cache.r7g.16xlarge | 6,2 | Y | Y | Y | 30 | T/A |
| cache.r6g.large | 5.0.6 | T | T | T | 0,75 | 10,0 |
| cache.r6g.xlarge | 5.0.6 | Y | Y | Y | 1,25 | 10,0 |
| cache.r6g.2xlarge | 5.0.6 | Y | Y | Y | 2,5 | 10,0 |
| cache.r6g.4xlarge | 5.0.6 | Y | Y | Y | 5,0 | 10,0 |
| cache.r6g.8xlarge | 5.0.6 | Y | Y | Y | 12 | T/A |
| cache.r6g.12xlarge | 5.0.6 | Y | Y | Y | 20 | T/A |

| Jenis instans | Versi Redis minimum yang didukung | I/O ditingkatkan (Redis 5.0.6 +) | TLS Offloading (Redis 6.2.5+) | Peningkatan I/O Multiplexing (Redis 7.0.4+) | Bandwidth dasar (Gbps) | Bandwidth burst (Gbps) |
|---|---|---|---|---|---|---|
| cache.r6g.16xlarge | 5.0.6 | Y | Y | Y | 25 | T/A |
| cache.r5.large | 3.2.4 | T | T | T | 0,75 | 10,0 |
| cache.r5.xlarge | 3.2.4 | Y | T | T | 1,25 | 10,0 |
| cache.r5.2xlarge | 3.2.4 | Y | Y | Y | 2,5 | 10,0 |
| cache.r5.4xlarge | 3.2.4 | Y | Y | Y | 5,0 | 10,0 |
| cache.r5.12xlarge | 3.2.4 | Y | Y | Y | 12 | T/A |
| cache.r5.24xlarge | 3.2.4 | Y | Y | Y | 25 | T/A |
| cache.r4.large | 3.2.4 | T | T | T | 0,75 | 10,0 |
| cache.r4.xlarge | 3.2.4 | Y | T | T | 1,25 | 10,0 |
| cache.r4.2xlarge | 3.2.4 | Y | Y | Y | 2,5 | 10,0 |
| cache.r4.4xlarge | 3.2.4 | Y | Y | Y | 5,0 | 10,0 |
| cache.r4.8xlarge | 3.2.4 | Y | Y | Y | 12 | T/A |
| cache.r4.16xlarge | 3.2.4 | Y | Y | Y | 25 | T/A |

Memori yang dioptimalkan dengan tingkatan data

| Jenis instans | Versi Redis minimum yang didukung | I/O ditingkatkan (Redis 5.0.6 +) | TLS Offloading (Redis 6.2.5+) | Peningkatan I/O Multiplexing (Redis 7.0.4+) | Bandwidth dasar (Gbps) | Bandwidth burst (Gbps) |
|---|---|---|---|---|---|---|
| cache.r6gd.xlarge | 6.2.0 | Y | T | T | 1,25 | 10 |
| cache.r6gd.2xlarge | 6.2.0 | Y | Y | Y | 2,5 | 10 |
| cache.r6gd.4xlarge | 6.2.0 | Y | Y | Y | 5,0 | 10 |
| cache.r6gd.8xlarge | 6.2.0 | Y | Y | Y | 12 | T/A |
| cache.r6gd.12xlarge | 6.2.0 | Y | Y | Y | 20 | T/A |
| cache.r6gd.16xlarge | 6.2.0 | Y | Y | Y | 25 | T/A |

Jaringan yang dioptimalkan

| Jenis instans | Versi Redis minimum yang didukung | I/O ditingkatkan (Redis 5.0.6 +) | TLS Offloading (Redis 6.2.5+) | Peningkatan I/O Multiplexing (Redis 7.0.4+) | Bandwidth dasar (Gbps) | Bandwidth burst (Gbps) |
|---|---|---|---|---|---|---|
| cache.c7gn.large | 6,2 | T | T | T | 6,25 | 30 |

| Jenis instans | Versi Redis minimum yang didukung | I/O ditingkatkan (Redis 5.0.6 +) | TLS Offloading (Redis 6.2.5+) | Peningkatan I/O Multiplexing (Redis 7.0.4+) | Bandwidth dasar (Gbps) | Bandwidth burst (Gbps) |
|---|---|---|---|---|---|---|
| cache.c7gn.xlarge | 6,2 | Y | Y | Y | 12,5 | 40 |
| cache.c7gn.2xlarge | 6,2 | Y | Y | Y | 25 | 50 |
| cache.c7gn.4xlarge | 6,2 | Y | Y | Y | 50 | T/A |
| cache.c7gn.8xlarge | 6,2 | Y | Y | Y | 100 | T/A |
| cache.c7gn.12xlarge | 6,2 | Y | Y | Y | 150 | T/A |
| cache.c7gn.16xlarge | 6,2 | Y | Y | Y | 200 | T/A |

## Jenis simpul yang didukung oleh Wilayah AWS

Jenis simpul yang didukung dapat bervariasi antar Wilayah AWS. Untuk detail selengkapnya, lihat harga Amazon ElastiCache.

## instans Performa yang Dapat Melonjak

Anda dapat meluncurkan simpul cache tujuan umum T4g, T3-Standard, dan T2-Standard yang dapat melonnjak di Amazon ElastiCache. Simpul ini menyediakan tingkat performa CPU dasar dengan kemampuan untuk menangani lonjakan penggunaan CPU kapan pun hingga kredit terakumulasi habis. Kredit CPU menyediakan performa inti CPU penuh selama satu menit.

Simpul T4g, T3 dan T2 dari Amazon ElastiCache dikonfigurasi sebagai standar dan sesuai untuk beban kerja dengan penggunaan CPU rata-rata yang secara konsisten di bawah performa dasar instans. Untuk melonjak di atas batas dasar, simpul menggunakan kredit yang telah diakumulasikan dalam saldo kredit CPU. Jika simpul kehabisan kredit yang dikumpulkan, performa secara bertahap diturunkan ke tingkat performa dasar. Penurunan bertahap ini memastikan simpul tidak mengalami penurunan performa yang tajam saat saldo kredit CPU yang dikumpulkannya terpakai. Untuk informasi selengkapnya, lihat Kredit CPU dan Performa Dasar untuk Instans Performa yang Dapat Melonjak di Panduan Pengguna Amazon EC2.

Tabel berikut mencantumkan jenis simpul performa yang dapat melonjak, tingkat di mana kredit CPU dihasilkan per jam. Hal ini juga menunjukkan jumlah maksimum kredit CPU yang dihasilkan yang dapat dikumpulkan oleh simpul dan jumlah vCPUs per simpul. Selain itu, tabel menunjukkan tingkat performa dasar sebagai persentase dari performa inti penuh (menggunakan satu vCPU tunggal).

| Kredit CPU dihasilkan per jam | Kredit maksimum yang dihasilkan yang dapat dikumpulkan* | vCPU | Performa dasar per vCPU | Memori (GiB) | Performa jaringan |
|---|---|---|---|---|---|
| 12 | 288 | 2 | 10% | 0,5 | Hingga 5 Gigabit |
| 24 | 576 | 2 | 20% | 1,37 | Hingga 5 Gigabit |
| 24 | 576 | 2 | 20% | 3,09 | Hingga 5 Gigabit |
| 12 | 288 | 2 | 10% | 0,5 | Hingga 5 Gigabit |
| 24 | 576 | 2 | 20% | 1,37 | Hingga 5 Gigabit |

| Kredit CPU dihasilkan per jam | Kredit maksimum yang dihasilkan yang dapat dikumpulkan* | vCPU | Performa dasar per vCPU | Memori (GiB) | Performa jaringan |
|---|---|---|---|---|---|
| 24 | 576 | 2 | 20% | 3,09 | Hingga 5 Gigabit |
| 6 | 144 | 1 | 10% | 0,5 | Rendah hingga sedang |
| 12 | 288 | 1 | 20% | 1,55 | Rendah hingga sedang |
| 24 | 576 | 2 | 20% | 3,22 | Rendah hingga sedang |

* Jumlah kredit yang dapat terakumulasi setara dengan jumlah kredit yang dapat diperoleh dalam periode 24 jam.

** Performa dasar dalam tabel adalah per vCPU. Beberapa ukuran simpul yang memiliki lebih dari satu vCPU. Untuk ini, hitung pemanfaatan CPU garis dasar untuk simpul dengan cara mengalikan persentase vCPU dengan jumlah vCPUs.

Metrik kredit CPU berikut tersedia untuk instans performa yang dapat melonjak T3 dan T4g:

> **Note**
>
> Metrik ini tidak tersedia untuk instans performa yang dapat melonjak T2.

- `CPUCreditUsage`

- `CPUCreditBalance`

Untuk informasi selengkapnya tentang metrik ini, lihat Metrik Kredit CPU.

Selain itu, perhatikan detail berikut:

- Semua jenis simpul generasi saat ini dibuat di cloud privat virtual (VPC) berdasarkan Amazon VPC secara default.
- File tambahkan-saja (AOF) Redis tidak didukung untuk instans T2. Variabel konfigurasi Redis `appendonly` dan `appendfsync` tidak didukung pada Redis versi 2.8.22 dan yang lebih baru.

### Informasi Terkait

- Fitur dan Detail Produk Amazon ElastiCache
- Parameter spesifik Redis
- Enkripsi dalam Transit (TLS)

## Mem-boot ulang simpul (mode klaster dinonaktifkan saja)

Beberapa perubahan mengharuskan simpul klaster di-boot ulang agar perubahan dapat diterapkan. Misalnya, untuk beberapa parameter, mengubah nilai parameter di dalam grup parameter hanya diterapkan setelah boot ulang.

Untuk klaster Redis (mode klaster dinonaktifkan), parameter tersebut adalah:

- activerehashing
- databases

Anda dapat boot ulang simpul hanya menggunakan konsol ElastiCache. Anda hanya dapat boot ulang satu simpul dalam satu waktu. Untuk boot ulang beberapa simpul Anda harus mengulangi proses untuk setiap simpul.

> **Perubahan parameter Redis (Mode Klaster Diaktifkan)**
>
> Jika Anda membuat perubahan parameter berikut pada klaster Redis (mode klaster diaktifkan), ikuti langkah-langkah berikutnya.

- activerehashing
- databases

1. Membuat cadangan manual klaster Anda. Lihat Mengambil cadangan manual.
2. Menghapus klaster Redis (mode klaster diaktifkan). Lihat Menghapus klaster.
3. Pulihkan klaster menggunakan grup parameter yang sudah diubah dan backup untuk menyemai klaster baru. Lihat Melakukan pemulihan dari cadangan ke dalam cache baru.

Perubahan parameter lain tidak memerlukan ini.

Menggunakan AWS Management Console

Anda dapat boot ulang simpul menggunakan konsol ElastiCache.

Untuk boot ulang simpul (konsol)

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Dari daftar di sudut kanan atas, pilih Wilayah AWS yang berlaku.
3. Di panel navigasi kiri, pilih Redis.

   Muncul daftar berisi klaster yang menjalankan Redis.
4. Pilih klaster di bawah Nama Klaster.
5. Di bawah Nama simpul, pilih tombol radio di samping simpul yang ingin Anda boot ulang.
6. Pilih Tindakan, lalu pilih Boot ulang simpul.

Untuk boot ulang beberapa simpul, ulangi langkah 2 hingga 5 untuk setiap simpul yang ingin Anda boot ulang. Anda tidak perlu menunggu boot satu simpul sampai selesai untuk melakukan boot ulang pada yang lain.

# Mengganti simpul

Amazon ElastiCache for Redis sering meningkatkan armadanya dengan patch dan peningkatan yang diterapkan di instans dengan lancar. Namun, dari waktu ke waktu kita perlu meluncurkan ulang simpul ElastiCache for Redis Anda untuk menerapkan pembaruan OS yang wajib ke host yang mendasarinya. Penggantian ini diperlukan untuk menerapkan peningkatan yang memperkuat keamanan, keandalan, dan performa operasional.

Anda memiliki pilihan untuk mengelola penggantian ini sendiri setiap saat sebelum periode penggantian simpul yang terjadwal. Ketika Anda mengelola penggantian sendiri, instans Anda menerima pembaruan OS ketika Anda meluncurkan kembali simpul tersebut dan penggantian simpul terjadwal Anda dibatalkan. Anda mungkin menerima terus peringatan yang menunjukkan bahwa penggantian simpul sedang terjadi. Jika Anda telah mengurangi kebutuhan pemeliharaan secara manual, Anda dapat mengabaikan peringatan ini.

> **Note**
>
> Penggantian simpul cache yang secara otomatis dihasilkan oleh Amazon ElastiCache mungkin memiliki alamat IP yang berbeda. Anda bertanggung jawab untuk meninjau konfigurasi aplikasi Anda untuk memastikan bahwa simpul cache Anda terkait dengan alamat IP yang sesuai.

Daftar berikut mengidentifikasi tindakan yang dapat Anda lakukan ketika ElastiCache menjadwalkan penggantian salah satu simpul Redis Anda. Untuk mempercepat pencarian informasi yang Anda butuhkan untuk situasi Anda, pilih dari menu berikut.

- Do nothing – Biarkan Amazon ElastiCache mengganti simpul seperti yang dijadwalkan.
- Change your maintenance window – Ubah periode pemeliharaan Anda ke waktu yang lebih baik.
- Konfigurasi Redis (mode klaster diaktifkan)
  - Replace the only node in any Redis cluster – Prosedur untuk mengganti simpul di klaster Redis menggunakan cadangan dan pemulihan.
  - Replace a replica node in any Redis cluster – Prosedur untuk mengganti replika baca di setiap klaster Redis dengan meningkatkan dan mengurangi jumlah replika tanpa waktu henti klaster.
  - Replace any node in a Redis (cluster mode enabled) shard – Prosedur dinamis tanpa waktu henti klaster untuk mengganti simpul dalam klaster Redis (mode klaster diaktifkan) dengan menskalakan ke luar dan ke dalam.

- Konfigurasi Redis (mode klaster dinonaktifkan)
  - Replace the only node in any Redis cluster – Prosedur untuk mengganti simpul apa pun di klaster Redis menggunakan cadangan dan pemulihan.
  - Replace a replica node in any Redis cluster – Prosedur untuk mengganti replika baca di setiap klaster Redis dengan meningkatkan dan mengurangi jumlah replika tanpa waktu henti klaster.
  - Replace a node in a Redis (cluster mode disabled) cluster – Prosedur untuk mengganti simpul dalam klaster Redis (mode klaster dinonaktifkan) menggunakan replikasi.
  - Replace a Redis (cluster mode disabled) read-replica – Prosedur untuk secara manual mengganti replika baca di grup replikasi Redis (mode klaster dinonaktifkan).
  - Replace a Redis (cluster mode disabled) primary node – Prosedur untuk secara manual mengganti simpul primer di grup replikasi Redis (mode klaster dinonaktifkan).

Pilihan penggantian simpul Redis

- Jangan lakukan apa pun – Jika Anda tidak melakukan apa pun, ElastiCache mengganti simpul seperti yang terjadwal.

  Untuk konfigurasi non-klaster dengan pengaktifan failover otomatis, klaster pada Redis 5.0.6 dan di atasnya menyelesaikan penggantian sambil klaster meneruskan tetap online dan melayani permintaan tulis yang masuk. Untuk klaster dengan pengaktifan failover otomatis pada Redis 4.0.10 atau yang lebih lama, Anda mungkin mengalami gangguan singkat selama proses tulis hingga beberapa detik yang terkait dengan pembaruan DNS.

  Jika simpul adalah anggota dari klaster yang mengaktifkan failover otomatis, ElastiCache for Redis menyediakan peningkatan ketersediaan selama proses patch, pembaruan, dan penggantian simpul lain yang terkait pemeliharaan.

  Untuk konfigurasi Klaster ElastiCache for Redis yang diatur untuk menggunakan klien Klaster ElastiCache for Redis, penggantian saat ini dapat diselesaikan sambil klaster melayani permintaan tulis yang masuk.

Untuk konfigurasi non-klaster dengan pengaktifan failover otomatis, klaster pada Redis 5.0.6 dan di atasnya menyelesaikan penggantian sambil klaster meneruskan tetap online dan melayani permintaan tulis yang masuk. Untuk klaster dengan pengaktifan failover otomatis pada Redis 4.0.10 atau yang lebih lama, Anda mungkin mengalami gangguan singkat selama proses tulis hingga beberapa detik yang terkait dengan pembaruan DNS.

Jika simpul adalah mandiri, Amazon ElastiCache pertama meluncurkan simpul pengganti, lalu menyinkronkan dari simpul yang ada. Simpul yang ada tidak tersedia untuk permintaan layanan selama waktu ini. Setelah sinkronisasi selesai, simpul yang ada akan dihentikan dan simpul baru akan menggantikannya. ElastiCache melakukan upaya terbaik untuk mempertahankan data Anda selama operasi ini.

- Ubah periode pemeliharaan Anda - Untuk peristiwa pemeliharaan terjadwal, Anda menerima email atau peristiwa notifikasi dari ElastiCache. Dalam kasus ini, jika Anda mengubah periode pemeliharaan Anda sebelum waktu penggantian terjadwal, simpul Anda sekarang digantikan pada waktu yang baru. Untuk informasi selengkapnya tentang IAM, lihat hal berikut:
  - Memodifikasi sebuah cluster ElastiCache
  - Mengubah grup replikasi

  > **Note**
  >
  > Kemampuan untuk mengubah periode penggantian dengan memindahkan periode pemeliharaan hanya tersedia jika notifikasi ElastiCache mencakup periode pemeliharaan. Jika notifikasi tidak memasukkan periode pemeliharaan, Anda tidak dapat mengubah periode penggantian.

  Misalnya, katakanlah pemeliharaan dilakukan pada Kamis, 9 November, pukul 15:00 dan jendela pemeliharaan berikutnya adalah Jumat, 10 November, pukul 17:00. Berikut adalah tiga skenario dengan hasilnya:

  - Anda mengubah periode pemeliharaan Anda ke Jumat pukul 16:00, setelah tanggal dan waktu saat ini dan sebelum jendela pemeliharaan terjadwal berikutnya. Simpul digantikan pada hari Jumat, 10 November, pukul 16:00.

- Anda mengubah periode pemeliharaan Anda ke Sabtu pukul 16:00, setelah tanggal dan waktu saat ini dan sebelum jendela pemeliharaan terjadwal berikutnya. Simpul digantikan pada hari Sabtu, 11 November, pukul 16:00.
- Anda mengubah jendela pemeliharaan Anda menjadi hari Rabu pukul 16:00, di awal minggu dari tanggal dan waktu saat ini). Simpul digantikan pada Rabu depan setelahnya, 15 November, pukul 16:00.

Untuk petunjuk, lihat Mengelola pemeliharaan.

- Ganti satu-satunya simpul dalam setiap klaster Redis – Jika klaster tidak memiliki replika baca apa pun, Anda dapat menggunakan prosedur berikut untuk mengganti simpul.

  Untuk mengganti satu-satunya simpul menggunakan cadangan dan pemulihan

  1. Buat snapshot dari klaster simpul. Untuk petunjuk, lihat Mengambil cadangan manual.
  2. Buat klaster baru dengan penyemaian data dari snapshot. Untuk petunjuk, lihat Melakukan pemulihan dari cadangan ke dalam cache baru.
  3. Hapus klaster dengan simpul yang dijadwalkan untuk penggantian. Untuk petunjuk, lihat Menghapus klaster.
  4. Di aplikasi Anda, ganti titik akhir dari simpul lama dengan titik akhir dari simpul baru.

- Ganti simpul replika di setiap klaster Redis – Untuk mengganti klaster replika, tingkatkan jumlah replika Anda. Untuk melakukannya, tambahkan replika kemudian kurangi jumlah replika dengan menghapus replika yang ingin Anda ganti. Proses ini bersifat dinamis dan tidak menimbulkan waktu henti klaster.

  > **Note**
  >
  > Jika serpihan atau grup replikasi Anda sudah memiliki replika, balikkan langkah 1 dan 2.

  Untuk mengganti replika di setiap klaster Redis

  1. Tingkatkan jumlah replika dengan menambahkan replika ke serpihan atau grup replikasi. Untuk informasi selengkapnya, lihat Menambah jumlah replika dalam serpihan.

2. Hapus replika yang ingin Anda ganti. Untuk informasi selengkapnya, lihat Mengurangi jumlah replika dalam serpihan.
3. Perbarui titik akhir dalam aplikasi Anda.

- Ganti setiap simpul dalam serpihan Redis (mode klaster diaktifkan) – Untuk mengganti simpul dalam klaster tanpa waktu henti, gunakan resharding online. Pertama, tambahkan serpihan dengan melakukan penskalaan ke luar, lalu hapus serpihan dengan simpul yang akan diganti dengan melakukan penskalakan ke dalam.

  Untuk mengganti setiap simpul di klaster Redis (mode klaster diaktifkan)

  1. Skalakan keluar: Tambahkan serpihan tambahan dengan konfigurasi yang sama seperti serpihan yang ada dengan simpul yang akan diganti. Untuk informasi selengkapnya, lihat Menambahkan serpihan dengan resharding online.
  2. Penskalaan ke dalam: Hapus serpihan dengan simpul yang akan diganti. Untuk informasi selengkapnya, lihat Menghapus serpihan dengan resharding online.
  3. Perbarui titik akhir dalam aplikasi Anda.

- Ganti simpul dalam sebuah klaster Redis (mode klaster dinonaktifkan) – Jika klaster adalah klaster Redis (mode klaster dinonaktifkan) tanpa replika baca, gunakan prosedur berikut untuk menggantikan simpul.

  Untuk mengganti simpul menggunakan replikasi (hanya mode penonaktifan klaster)

  1. Tambahkan replikasi ke klaster dengan simpul yang dijadwalkan untuk penggantian sebagai primer. Jangan mengaktifkan Multi-AZ di klaster ini. Untuk petunjuk, lihat Untuk menambahkan replikasi ke klaster Redis tanpa serpihan.
  2. Tambahkan replika-baca ke klaster. Untuk petunjuk, lihat Untuk menambahkan simpul ke klaster (konsol).
  3. Naikkan replika-baca yang baru dibuat menjadi primer. Untuk petunjuk, lihat Menaikkan replika baca menjadi primer, untuk grup replikasi Redis (mode klaster dinonaktifkan).
  4. Hapus simpul yang dijadwalkan untuk penggantian. Untuk petunjuk, lihat Menghapus simpul dari klaster.

5. Di aplikasi Anda, ganti titik akhir dari simpul lama dengan titik akhir dari simpul baru.

- Ganti replika baca Redis (mode klaster dinonaktifkan) – Jika simpul adalah replika baca, gantikan simpul tersebut.

  Jika klaster Anda memiliki hanya satu simpul replika dan Multi-AZ diaktifkan, Anda harus menonaktifkan Multi-AZ sebelum dapat menghapus replika. Untuk petunjuk, lihat Mengubah grup replikasi.

  Untuk mengganti replika baca Redis (mode klaster dinonaktifkan)

  1. Hapus replika yang dijadwalkan untuk penggantian. Untuk instruksi, lihat yang berikut ini:
     - Mengurangi jumlah replika dalam serpihan
     - Menghapus simpul dari klaster
  2. Tambahkan replika baru untuk menggantikan salah satu yang dijadwalkan untuk penggantian. Jika Anda menggunakan nama yang sama dengan replika yang baru saja dihapus, Anda dapat melewatkan langkah 3. Untuk instruksi, lihat yang berikut ini:
     - Menambah jumlah replika dalam serpihan
     - Menambahkan replika baca, untuk grup replikasi Redis (Mode Klaster Dinonaktifkan)
  3. Di aplikasi Anda, ganti titik akhir dari replika lama dengan titik akhir dari replika baru.
  4. Jika Anda menonaktifkan Multi-AZ di awal, aktifkan kembali sekarang. Untuk petunjuk, lihat Mengaktifkan Multi-AZ .

- Ganti simpul primer Redis (mode klaster dinonaktifkan) – Jika simpul adalah simpul primer, naikkan replika baca menjadi primer terlebih dahulu. Kemudian hapus replika yang sebelumnya menjadi simpul primer.

  Jika klaster Anda memiliki hanya satu replika dan Multi-AZ diaktifkan, Anda harus menonaktifkan Multi-AZ sebelum dapat menghapus replika pada langkah 2. Untuk petunjuk, lihat Mengubah grup replikasi.

Untuk mengganti simpul primer Redis (mode klaster dinonaktifkan)

1. Promosikan replika baca menjadi primer. Untuk petunjuk, lihat Menaikkan replika baca menjadi primer, untuk grup replikasi Redis (mode klaster dinonaktifkan).
2. Hapus simpul yang dijadwalkan untuk penggantian (primer yang lama). Untuk petunjuk, lihat Menghapus simpul dari klaster.
3. Tambahkan replika baru untuk menggantikan replika yang dijadwalkan untuk penggantian. Jika Anda menggunakan nama yang sama dengan simpul yang baru saja Anda hapus, Anda tidak perlu melakukan perubahan titik akhir dalam aplikasi Anda.

   Untuk petunjuk, lihat Menambahkan replika baca, untuk grup replikasi Redis (Mode Klaster Dinonaktifkan).
4. Di aplikasi Anda, ganti titik akhir dari simpul lama dengan titik akhir dari simpul baru.
5. Jika Anda menonaktifkan Multi-AZ di awal, aktifkan kembali sekarang. Untuk petunjuk, lihat Mengaktifkan Multi-AZ .

# Simpul terpesan ElastiCache

Memesan satu atau beberapa simpul dapat menjadi cara untuk mengurangi biaya. Simpul cadangan dikenakan biaya di muka dan bergantung pada jenis simpul dan lamanya reservasi— satu atau tiga tahun.

Untuk melihat apakah simpul terpesan menghemat biaya kasus penggunaan Anda, pertama-tama tentukan dulu ukuran simpul dan jumlah simpul yang Anda butuhkan. Kemudian, estimasikan penggunaan simpul dan bandingkan total biaya jika Anda menggunakan simpul sesuai permintaan versus dengan simpul terpesan. Anda dapat memadupadankan penggunaan simpul terpesan dan simpul sesuai permintaan dalam klaster Anda. Untuk informasi harga, lihat Harga Amazon ElastiCache.

> **Note**
>
> Simpul terpesan bersifat tidak fleksibel; simpul tersebut hanya diterapkan untuk jenis instans yang persis Anda pesan.

## Mengelola biaya dengan simpul terpesan

Menyimpan satu atau beberapa simpul dapat menjadi cara untuk mengurangi biaya. Simpul cadangan dikenakan biaya di muka dan bergantung pada jenis simpul dan lamanya reservasi—satu atau tiga tahun. Biaya ini jauh lebih kecil daripada biaya penggunaan per jam yang dikenakan untuk simpul sesuai permintaan.

Untuk melihat apakah simpul terpesan menghemat biaya kasus penggunaan Anda, pertama-tama tentukan dulu ukuran simpul dan jumlah simpul yang Anda butuhkan. Kemudian, estimasikan penggunaan simpul dan bandingkan total biaya jika Anda menggunakan simpul sesuai permintaan versus dengan simpul terpesan. Anda dapat memadupadankan penggunaan simpul terpesan dan simpul sesuai permintaan dalam klaster Anda. Untuk informasi harga, lihat Harga Amazon ElastiCache.

Wilayah AWS, tipe simpul, dan durasi jangka waktu harus dipilih saat pembelian dan tidak dapat diubah nanti.

Anda dapat menggunakan API AWS Management Console, AWS CLI, atau ElastiCache untuk membuat daftar dan membeli penawaran simpul cadangan yang tersedia.

Untuk informasi selengkapnya tentang simpul terpesan, lihat Simpul Terpesan Amazon ElastiCache.

Topik

- Penawaran simpul terpesan standar
- Penawaran simpul terpesan warisan
- Mendapatkan info tentang penawaran simpul terpesan
- Membeli simpul terpesan
- Mendapatkan info tentang simpul terpesan Anda

## Penawaran simpul terpesan standar

Saat membeli instans simpul terpesan (RI) di Amazon ElastiCache, Anda membeli komitmen untuk mendapatkan tarif diskon pada jenis instans simpul dan Wilayah AWS tertentu selama durasi instans simpul terpesan. Untuk menggunakan instans simpul terpesan Amazon ElastiCache, Anda membuat instans simpul ElastiCache yang baru, seperti yang Anda lakukan untuk instans sesuai permintaan.

Instans simpul baru yang Anda buat harus sama persis dengan spesifikasi instans simpul terpesan. Jika spesifikasi instans simpul baru sama dengan instans simpul terpesan yang sudah ada untuk akun Anda, Anda akan ditagih dengan tarif diskon yang ditawarkan untuk instans simpul terpesan. Jika tidak, instans simpul ditagih dengan tarif sesuai permintaan. RI standar ini tersedia dari keluarga instans R5 dan M5 seterusnya.

> **Note**
>
> Ketiga jenis penawaran yang dibahas berikutnya tersedia dalam jangka waktu satu tahun dan tiga tahun.

### Jenis Penawaran

RI Tanpa Biaya di Muka menyediakan akses ke instans ElastiCache terpesan tanpa memerlukan pembayaran di muka. Instans ElastiCache terpesan Tanpa Biaya di Muka Anda mengenakan tagihan dengan tarif per jam yang didiskon untuk setiap jam dalam jangka waktu yang ditentukan, terlepas dari penggunaan.

RI Biaya di Muka Sebagian mengharuskan sebagian instans ElastiCache terpesan untuk dibayar di muka. Sisa jam dalam jangka waktu pemesanan akan ditagih dengan tarif per jam yang didiskon,

terlepas dari penggunaannya. Opsi ini adalah pengganti opsi warisan Penggunaan Berat, yang dijelaskan di bagian berikutnya.

RI Semua Biaya di Muka membutuhkan pembayaran penuh yang harus dilakukan di awal jangka waktu RI. Anda tidak dikenakan biaya lain untuk sisa jangka waktu terlepas dari jumlah jam yang digunakan.

Penawaran simpul terpesan warisan

Ada tiga tingkat reservasi simpul warisan—Penggunaan Berat, Penggunaan Sedang, dan Penggunaan Ringan. Simpul dapat dipesan pada tingkat penggunaan mana pun selama satu atau tiga tahun. Jenis simpul, tingkat penggunaan, dan jangka waktu reservasi memengaruhi total biaya Anda. Verifikasi penghematan yang dapat diperoleh dari simpul cadangan yang dapat memfasilitasi bisnis Anda dengan membandingkan berbagai model sebelum Anda membeli simpul cadangan.

Simpul yang dibeli pada satu tingkat penggunaan atau jangka waktu tidak dapat dikonversi ke tingkat penggunaan atau jangka waktu yang berbeda.

Tingkat Penggunaan

Simpul terpesan Penggunaan Berat memungkinkan beban kerja yang memiliki dasar kapasitas yang konsisten atau menjalankan beban kerja dengan status yang stabil. Simpul terpesan Penggunaan Berat memerlukan komitmen di muka yang tinggi. Namun, jika Anda berencana menjalankan lebih dari 79 persen jangka waktu simpul terpesan, Anda bisa mendapatkan penghematan terbesar (hingga 70 persen dari harga Sesuai Permintaan). Dengan simpul cadangan Penggunaan Berat, Anda membayar biaya satu kali. Hal ini kemudian diikuti dengan biaya per jam yang lebih rendah selama durasi jangka waktu terlepas dari apakah simpul Anda berjalan.

Simpul terpesan Penggunaan Sedang adalah opsi terbaik jika Anda berencana untuk menggunakan simpul cadangan Anda dalam jumlah waktu yang banyak dan Anda ingin biaya satu kali yang lebih rendah atau untuk berhenti membayar simpul Anda ketika Anda mematikannya. Simpul cadangan Penggunaan Sedang adalah pilihan yang lebih hemat biaya jika Anda berencana untuk menjalankan lebih dari 40 persen jangka waktu simpul terpesan. Opsi ini dapat menghemat hingga 64 persen dari harga Sesuai Permintaan. Dengan simpul cadangan Penggunaan Sedang, Anda membayar biaya satu kali sedikit lebih tinggi dibandingkan dengan simpul cadangan Penggunaaan Ringan. Anda juga menerima tarif penggunaan per jam yang lebih rendah ketika Anda menjalankan simpul.

Simpul cadangan Penggunaan Ringan ideal untuk beban kerja periodik yang hanya berjalan beberapa jam sehari atau beberapa hari per minggu. Dengan menggunakan simpul terpesan Penggunaan Ringan, Anda membayar biaya satu kali yang diikuti dengan biaya penggunaan per jam

yang didiskon ketika simpul Anda berjalan. Anda dapat mulai menghemat ketika simpul Anda berjalan lebih dari 17 persen dari jangka waktu simpul terpesan. Anda dapat menghemat hingga 56 persen dari tarif Sesuai Permintaan di sepanjang jangka waktu simpul cadangan.

Penawaran simpul terpesan warisan

| Penawaran | Biaya di muka | Biaya penggunaan | Keuntungan |
|---|---|---|---|
| Penggunaan Berat | Tertinggi | Biaya per jam terendah. Diterapkan pada seluruh jangka waktu, baik Anda menggunakan simpul cadangan maupun tidak. | Biaya keseluruhan terendah jika Anda berencana untuk menjalankan simpul cadangan Anda lebih dari 79 persen selama jangka waktu tiga tahun. |
| Penggunaan Sedang | Sedang | Biaya penggunaan per jam dikenakan untuk setiap jam simpul berjalan. Tidak ada biaya per jam ketika simpul tidak berjalan. | Cocok untuk beban kerja elastis atau jika Anda mengharapkan penggunaan sedang, lebih dari 40 persen selama jangka waktu tiga tahun. |
| Penggunaan Ringan | Terendah | Biaya penggunaan per jam dikenakan untuk setiap jam simpul berjalan. Tidak ada biaya per jam ketika simpul tidak berjalan. Biaya per jam tertinggi dari semua jenis penawaran, tetapi biaya hanya berlaku | Biaya keseluruh an tertinggi jika Anda berencana untuk menjalank annya sepanjang waktu. Namun, biaya ini adalah biaya keseluruhan terendah jika Anda berencana untuk jarang menggunakan |

| Penawaran | Biaya di muka | Biaya penggunaan | Keuntungan |
| --- | --- | --- | --- |
| | | ketika simpul terpesan berjalan. | simpul terpesan, lebih dari sekitar 15 persen selama jangka waktu tiga tahun. |
| Penggunaa n Sesuai Permintaan (Tanpa simpul cadangar | Tidak ada | Biaya per jam tertinggi . Diterapkan setiap kali simpul berjalan. | Biaya per jam tertinggi . |

Untuk informasi selengkapnya, lihat Harga Amazon ElastiCache.

## Mendapatkan info tentang penawaran simpul terpesan

Sebelum Anda membeli simpul terpesan, Anda bisa mendapatkan informasi tentang penawaran simpul terpesan yang tersedia.

Contoh-contoh berikut menunjukkan cara mendapatkan harga dan informasi tentang penawaran simpul terpesan yang tersedia menggunakan API AWS Management Console, AWS CLI, dan ElastiCache.

Topik

- Mendapatkan info tentang penawaran simpul terpesan (Konsol)
- Mendapatkan info tentang penawaran simpul terpesan (AWS CLI)
- Mendapatkan info tentang penawaran simpul terpesan (API ElastiCache)

### Mendapatkan info tentang penawaran simpul terpesan (Konsol)

Untuk mendapatkan harga dan informasi selengkapnya tentang penawaran klaster terpesan yang tersedia menggunakan AWS Management Console, gunakan prosedur berikut.

Untuk mendapatkan informasi tentang penawaran simpul terpesan yang tersedia

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Di panel navigasi, pilih Simpul Terpesan.
3. Pilih Beli Simpul terpesan.
4. Untuk Mesin, pilih Redis.
5. Untuk menentukan penawaran yang tersedia, tentukan opsi berikut:

   - Jenis Simpul
   - Jangka Waktu
   - Jenis Penawaran

   Setelah Anda menentukan pilihan ini, biaya per simpul dan total biaya pilihan Anda akan ditampilkan di Detail reservasi.
6. Pilih Batalkan untuk menghindari pembelian simpul dan dikenakan biaya.

Mendapatkan info tentang penawaran simpul terpesan (AWS CLI)

Untuk mendapatkan harga dan informasi selengkapnya tentang penawaran simpul terpesan yang tersedia, ketikkan perintah berikut pada prompt perintah:

```
aws elasticache describe-reserved-cache-nodes-offerings
```

Operasi ini menghasilkan output yang serupa dengan yang berikut (format JSON):

```
{
          "ReservedCacheNodesOfferingId": "0xxxxxxxx-xxeb-44ex-xx3c-xxxxxxxx072",
          "CacheNodeType": "cache.xxx.large",
          "Duration": 94608000,
          "FixedPrice": XXXX.X,
          "UsagePrice": X.X,
          "ProductDescription": "redis",
          "OfferingType": "All Upfront",
          "RecurringCharges": [
              {
                  "RecurringChargeAmount": X.X,
                  "RecurringChargeFrequency": "Hourly"
              }
          ]
 },
 {
          "ReservedCacheNodesOfferingId": "0xxxxxxxx-xxeb-44ex-xx3c-xxxxxxxx072",
          "CacheNodeType": "cache.xxx.xlarge",
          "Duration": 94608000,
          "FixedPrice": XXXX.X,
          "UsagePrice": X.X,
          "ProductDescription": "redis",
          "OfferingType": "Partial Upfront",
          "RecurringCharges": [
              {
                  "RecurringChargeAmount": X.XXX,
                  "RecurringChargeFrequency": "Hourly"
              }
          ]
 },
 {
          "ReservedCacheNodesOfferingId": "0xxxxxxxx-xxeb-44ex-xx3c-xxxxxxxx072",
          "CacheNodeType": "cache.xxx.large",
          "Duration": 31536000,
```

```
            "FixedPrice": X.X,
            "UsagePrice": X.X,
            "ProductDescription": "redis",
            "OfferingType": "No Upfront",
            "RecurringCharges": [
                {
                    "RecurringChargeAmount": X.XXX,
                    "RecurringChargeFrequency": "Hourly"
                }
            ]
}
```

Untuk informasi selengkapnya, lihat describe-reserved-cache-nodes-offerings di Referensi AWS CLI.

Mendapatkan info tentang penawaran simpul terpesan (API ElastiCache)

Untuk mendapatkan harga dan informasi tentang penawaran simpul terpesan yang tersedia, panggil tindakan `DescribeReservedCacheNodesOfferings`.

Example

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeReservedCacheNodesOfferings
    &Version=2014-12-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &X-Amz-Algorithm
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

Untuk informasi selengkapnya, lihat DescribeReservedCacheNodesOfferings di Referensi API ElastiCache.

## Membeli simpul terpesan

Contoh-contoh berikut menunjukkan cara membeli penawaran simpul terpesan menggunakan API AWS Management Console, AWS CLI, dan ElastiCache.

> **Important**
>
> Mengikuti contoh di bagian ini akan dikenakan biaya pada akun AWS yang tidak dapat Anda balikkan.

Topik

- Membeli simpul terpesan (Konsol)
- Membeli simpul terpesan AWS CLI
- Membeli simpul cadangan (API ElastiCache)

### Membeli simpul terpesan (Konsol)

Contoh ini menunjukkan pembelian penawaran simpul terpesan yang spesifik, 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f, dengan ID simpul terpesan myreservationID.

Prosedur berikut menggunakan AWS Management Console untuk membeli penawaran simpul terpesan dengan menawarkan id.

Untuk membeli simpul terpesan

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Pada daftar navigasi, pilih tautan Simpul Terpesan.
3. Pilih tombol Beli simpul terpesan.
4. Untuk Mesin, pilih Redis.
5. Untuk menentukan penawaran yang tersedia, tentukan opsi berikut:
   - Jenis Simpul
   - Jangka Waktu
   - Jenis Penawaran
   - ID simpul terpesan opsional

Setelah Anda menentukan pilihan ini, biaya per simpul dan total biaya pilihan Anda akan ditampilkan di Detail reservasi.

6. Pilih Beli.

### Membeli simpul terpesan AWS CLI

Contoh berikut menunjukkan pembelian penawaran klaster terpesan yang spesifik, 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f, dengan ID simpul terpesan myreservationID.

Ketikkan perintah berikut pada prompt perintah:

Untuk Linux, macOS, atau Unix:

```
aws elasticache purchase-reserved-cache-nodes-offering \
    --reserved-cache-nodes-offering-id 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f \
    --reserved-cache-node-id myreservationID
```

Untuk Windows:

```
aws elasticache purchase-reserved-cache-nodes-offering ^
    --reserved-cache-nodes-offering-id 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f ^
    --reserved-cache-node-id myreservationID
```

Perintah tersebut mengembalikan output serupa dengan berikut ini:

```
RESERVATION  ReservationId      Class           Start Time                Duration
 Fixed Price  Usage Price  Count  State            Description      Offering Type
RESERVATION  myreservationid    cache.xx.small  2013-12-19T00:30:23.247Z  1y
 XXX.XX USD   X.XXX USD    1      payment-pending  memcached        Medium Utilization
```

Untuk informasi selengkapnya, lihat purchase-reserved-cache-nodes-offering di Referensi AWS CLI.

### Membeli simpul cadangan (API ElastiCache)

Contoh berikut menunjukkan pembelian penawaran simpul terpesan yang spesifik, 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f, dengan ID klaster terpesan myreservationID.

Panggil operasi `PurchaseReservedCacheNodesOffering` dengan parameter berikut ini:

- `ReservedCacheNodesOfferingId` = `649fd0c8-cf6d-47a0-bfa6-060f8e75e95f`
- `ReservedCacheNodeID` = `myreservationID`
- `CacheNodeCount` = `1`

Example

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=PurchaseReservedCacheNodesOffering
    &ReservedCacheNodesOfferingId=649fd0c8-cf6d-47a0-bfa6-060f8e75e95f
    &ReservedCacheNodeID=myreservationID
    &CacheNodeCount=1
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20141201T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

Untuk informasi selengkapnya, lihat PurchaseReservedCacheNodesOffering di Referensi API ElastiCache.

## Mendapatkan info tentang simpul terpesan Anda

Anda bisa mendapatkan informasi tentang simpul terpesan yang telah Anda beli menggunakan APIAWS Management Console, AWS CLI, dan ElastiCache.

Topik

- Mendapatkan info tentang simpul terpesan Anda (Konsol)
- Mendapatkan info tentang simpul terpesan Anda (AWS CLI)
- Mendapatkan info tentang simpul terpesan Anda (API ElastiCache)

### Mendapatkan info tentang simpul terpesan Anda (Konsol)

Prosedur berikut menjelaskan cara menggunakan AWS Management Console untuk mendapatkan informasi tentang simpul terpesan yang Anda beli.

Untuk mendapatkan informasi tentang simpul terpesan yang dibeli

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Pada daftar navigasi, pilih tautan Simpul terpesan.

   Simpul terpesan untuk akun Anda muncul di daftar Simpul terpesan. Anda dapat memilih salah satu dari simpul terpesan dalam daftar untuk melihat informasi mendetail tentang simpul terpesan di panel detail di bagian bawah konsol.

### Mendapatkan info tentang simpul terpesan Anda (AWS CLI)

Untuk mendapatkan informasi tentang simpul terpesan untuk akun AWS Anda, ketikkan perintah berikut pada prompt perintah:

```
aws elasticache describe-reserved-cache-nodes
```

Operasi ini menghasilkan output yang serupa dengan yang berikut (format JSON):

```
{
    "ReservedCacheNodeId": "myreservationid",
    "ReservedCacheNodesOfferingId": "649fd0c8-cf6d-47a0-bfa6-060f8e75e95f",
    "CacheNodeType": "cache.xx.small",
    "DataTiering": "disabled",
```

```
    "Duration": "31536000",
    "ProductDescription": "memcached",
    "OfferingType": "Medium Utilization",
    "MaxRecords": 0
}
```

Untuk informasi selengkapnya, lihat describe--reserved-cache-nodes di Referensi AWS CLI.

Mendapatkan info tentang simpul terpesan Anda (API ElastiCache)

Untuk mendapatkan informasi tentang simpul terpesan untuk akun AWS Anda, panggil operasi `DescribeReservedCacheNodes`.

Example

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeReservedCacheNodes
    &Version=2014-12-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20141201T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

Untuk informasi selengkapnya, lihat DescribeReservedCacheNodes di Referensi API ElastiCache.

## Memigrasikan simpul generasi sebelumnya

Simpul generasi sebelumnya adalah jenis simpul yang sedang dihentikan secara bertahap. Jika Anda tidak memiliki klaster yang menggunakan jenis simpul generasi sebelumnya, ElastiCache tidak mendukung pembuatan klaster baru dengan jenis simpul tersebut.

Karena jumlah terbatas jenis simpul generasi sebelumnya, kami tidak dapat menjamin pengganti sukses ketika simpul menjadi tidak sehat di klaster Anda. Dalam skenario seperti tersebut, ketersediaan klaster Anda mungkin berdampak negatif.

Sebaiknya migrasikan klaster Anda ke tipe simpul baru untuk ketersediaan dan kinerja yang lebih baik. Untuk tipe simpul yang direkomendasikan untuk dimigrasikan, lihat Jalur Peningkatan. Untuk

daftar lengkap jenis simpul yang didukung dan jenis simpul generasi sebelumnya di ElastiCache, lihat Tipe simpul yang didukung.

## Migrasi simpul di klaster Redis

Prosedur berikut menjelaskan cara memigrasikan tipe simpul klaster Redis Anda menggunakan Konsol ElastiCache. Selama proses ini, klaster Redis Anda akan terus melayani permintaan dengan waktu henti minimal. Bergantung pada konfigurasi klaster Anda, Anda mungkin mengalami waktu henti berikut. Berikut ini adalah perkiraan dan mungkin berbeda berdasarkan konfigurasi spesifik Anda:

- Mode klaster dinonaktifkan (simpul tunggal) mungkin melihat sekitar 60 detik, terutama karena propagasi DNS.
- Mode klaster dinonaktifkan (dengan replika simpul) mungkin melihat sekitar 1 detik untuk klaster yang menjalankan Redis 5.0.6 dan di atas. Semua versi yang lebih rendah dapat mengalami sekitar 10 detik.
- Mode klaster diaktifkan mungkin melihat sekitar 1 detik.

Untuk memodifikasi jenis simpul klaster Redis menggunakan konsol tersebut:

1. Masuk ke Konsol dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Dari panel navigasi, pilih Klaster Redis.
3. Pada daftar klaster, pilih klaster yang ingin Anda migrasikan.
4. Pilih Tindakan, lalu pilih Ubah.
5. Pilih jenis simpul baru dari daftar tipe simpul.
6. Jika Anda ingin segera melakukan proses migrasi, pilih Terapkan segera. Jika Terapkan segera tidak dipilih, proses migrasi dilakukan selama periode pemeliharaan klaster berikutnya.
7. Pilih Ubah. Jika Anda memilih Terapkan segera pada langkah sebelumnya, status klaster berubah ke sedang diubah. Ketika status berubah ke tersedia, pengubahan selesai dan Anda dapat mulai menggunakan klaster baru tersebut.

Untuk memodifikasi jenis simpul klaster Redis menggunakan AWS CLI:

Menggunakan API modify-replication-group seperti yang ditunjukkan berikut:

Untuk Linux, macOS, atau Unix:

```
aws elasticache modify-replication-group /
 --replication-group-id my-replication-group /
 --cache-node-type new-node-type /
 --apply-immediately
```

Untuk Windows:

```
aws elasticache modify-replication-group ^
 --replication-group-id my-replication-group ^
 --cache-node-type new-node-type ^
 --apply-immediately
```

Dalam skenario ini, nilai *new-node-type* adalah jenis simpul yang Anda migrasikan. Dengan melewati parameter `--apply-immediately`, pembaruan akan diterapkan segera ketika grup replikasi bertransisi dari status sedang diubah menjadi tersedia. Jika Terapkan segera tidak dipilih, proses migrasi dilakukan selama periode pemeliharaan klaster berikutnya.

> **Note**
>
> Jika Anda tidak dapat mengubah klaster dengan `InvalidCacheClusterState`, Anda perlu menghapus simpul yang gagal-pulih terlebih dahulu.

Memperbaiki atau menghapus restore-failed-node (s)

Prosedur berikut menjelaskan cara memperbaiki atau menghapus simpul yang gagal melakukan pemulihan dari klaster Redis Anda. Untuk mempelajari lebih lanjut cara simpul ElastiCache memasukkan status gagal pemulihan, lihat Melihat Status ElastiCache Node. Sebaiknya hapus terlebih dahulu simpul yang berada dalam status gagal pemulihan, lalu migrasikan simpul generasi sebelumnya yang tersisa di klaster ElastiCache ke tipe simpul generasi yang lebih baru, dan terakhir tambahkan kembali jumlah simpul yang diperlukan.

Untuk menghapus simpul gagal-pulih (konsol):

1. Masuk ke Konsol dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Dari panel navigasi, pilih Klaster Redis.
3. Dari daftar klaster, pilih nama klaster yang ingin Anda hapus simpulnya.

4. Dari daftar serpihan, pilih nama serpihan yang ingin Anda hapus simpulnya. Lewati langkah ini jika modw klaster dinonaktifkan untuk klaster.
5. Dari daftar simpul, memilih simpul dengan status `restore-failed`.
6. Pilih Tindakan, lalu pilih Hapus simpul.

Setelah menghapus simpul yang gagal-pulih dari klaster ElastiCache, Anda sekarang dapat bermigrasi ke jenis generasi yang lebih baru. Untuk informasi selengkapnya, lihat di atas di Migrasi simpul di klaster Redis.

Untuk menambahkan kembali simpul ke klaster ElastiCache Anda, lihat Menambahkan simpul ke klaster.

# Mengelola klaster

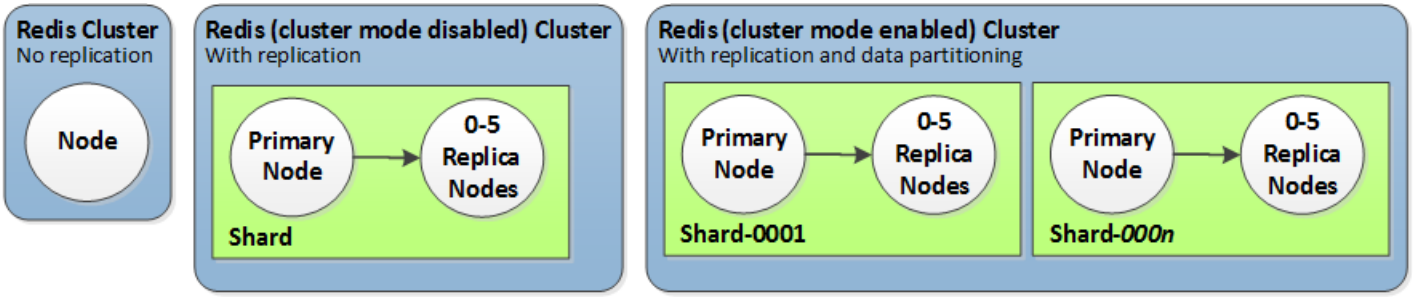
Klaster adalah kumpulan dari satu atau beberapa simpul cache, yang semuanya menjalankan instans dari perangkat lunak mesin cache Redis. Saat membuat klaster, Anda menentukan mesin dan versi yang akan digunakan oleh semua simpul.

Diagram berikut menggambarkan klaster Redis secara umum. Klaster Redis dapat berisi satu simpul atau hingga enam simpul dalam serpihan (API/CLI: grup simpul), satu klaster Redis (mode klaster dinonaktifkan) simpul tunggal tidak memiliki serpihan, dan klaster Redis (mode klaster dinonaktifkan) multi-simpul memiliki satu serpihan. Klaster Redis (mode klaster diaktifkan) dapat memiliki hingga 500 serpihan, dengan data Anda dipartisi di seluruh serpihan. Batas simpul atau serpihan dapat ditingkatkan hingga maksimum 500 per klaster jika mesin Redis yang digunakan memiliki versi 5.0.6 atau yang lebih tinggi. Sebagai contoh, Anda dapat memilih untuk membuat konfigurasi dari sebuah klaster dengan 500 simpul yang berkisar antara 83 serpihan (satu primer dan 5 replika per serpihan) dan 500 serpihan (primer tunggal dan tidak ada replika). Pastikan alamat IP yang tersedia mencukupi untuk mengakomodasi peningkatan tersebut. Kesalahan umum termasuk subnet dalam grup subnet memiliki rentang CIDR yang terlalu kecil atau subnet digunakan bersama dan banyak digunakan oleh klaster lain. Untuk informasi selengkapnya, lihat Membuat grup subnet. Untuk versi di bawah 5.0.6, batasnya adalah 250 per klaster.

Untuk meminta penambahan batas, lihat Batas Layanan AWS dan pilih jenis batas Simpul per klaster per jenis instans.

Jika Anda memiliki beberapa simpul dalam serpihan, salah satu simpulnya adalah simpul primer baca/tulis. Semua simpul lain dalam serpihan adalah replika hanya-baca.

Klaster Redis umumnya terlihat sebagai berikut.




Sebagian besar ElastiCache operasi dilakukan di tingkat cluster. Anda dapat menyiapkan klaster dengan jumlah simpul tertentu dan grup parameter yang mengontrol properti untuk setiap simpul. Semua simpul dalam klaster dirancang agar berupa jenis simpul yang sama serta memiliki parameter dan pengaturan grup keamanan yang sama.

Setiap klaster harus memiliki pengidentifikasi klaster. Pengidentifikasi klaster adalah nama yang diberikan pelanggan untuk klaster. Identifier ini menentukan cluster tertentu ketika berinteraksi dengan ElastiCache API dan perintah. AWS CLI Pengidentifikasi klaster harus unik untuk pelanggan tersebut di suatu AWS Wilayah.

ElastiCache mendukung beberapa versi mesin. Kecuali jika Anda memiliki alasan tertentu, kami menyarankan menggunakan versi terbaru.

ElastiCache cluster dirancang untuk diakses menggunakan instans Amazon EC2. Jika Anda meluncurkan klaster Anda di cloud privat virtual (VPC) berdasarkan layanan Amazon VPC, Anda dapat mengaksesnya dari luar AWS. Untuk informasi selengkapnya, lihat Mengakses sumber daya ElastiCache dari luar AWS.

Untuk daftar versi Redis yang didukung, lihat Versi ElastiCache for Redis yang Didukung.

# Memilih jenis jaringan

ElastiCache mendukung Internet Protocol versi 4 dan 6 (IPv4 dan IPv6), memungkinkan Anda untuk mengonfigurasi klaster untuk menerima:

- hanya koneksi IPv4,
- hanya koneksi IPv6,
- koneksi IPv4 dan IPv6 sekaligus (tumpukan ganda)

IPv6 didukung untuk beban kerja menggunakan mesin Redis versi 6.2 dan seterusnya pada semua instans yang dibangun pada sistem Nitro. Tidak ada biaya tambahan untuk mengakses ElastiCache melalui IPv6.

> **Note**
>
> Migrasi klaster yang dibuat sebelum ketersediaan IPV6/tumpukan ganda tidak didukung. Beralih antar jenis jaringan pada klasteryang baru dibuat juga tidak didukung.

## Mengonfigurasi subnet untuk jenis jaringan

Jika Anda membuat klaster di Amazon VPC, Anda harus menentukan grup subnet. ElastiCache menggunakan grup subnet tersebut untuk memilih subnet dan alamat IP dalam subnet tersebut yang akan diasosiasikan dengan simpul Anda. Klaster ElastiCache memerlukan subnet tumpukan ganda dengan alamat IPv4 dan IPv6 yang ditetapkan untuk beroperasi dalam mode tumpukan ganda dan subnet khusus IPv6 untuk beroperasi sebagai IPv6 saja.

## Menggunakan tumpukan ganda

Saat menggunakan ElastiCache for Redis dalam mode klaster diaktifkan, dari sudut pandang aplikasi, sambungan ke semua simpul klaster melalui titik akhir konfigurasi tidak berbeda dengan sambungan secara langsung ke tiap-tiap simpul cache. Untuk mencapai hal ini, klien yang sadar klaster harus terlibat dalam proses penemuan klaster dan meminta informasi konfigurasi untuk semua simpul. Protokol penemuan Redis hanya mendukung satu IP per simpul.

Untuk mempertahankan kompatibilitas mundur dengan semua klien yang ada, penemuan IP digunakan, yang memungkinkan Anda memilih jenis IP (yaitu, IPv4 atau IPv6) untuk dinyatakan di protokol penemuan. Meskipun ini membatasi penemuan otomatis hanya untuk satu jenis

IP, tumpukan ganda masih bermanfaat untuk beban kerja mode klaster diaktifkan, karena memungkinkan migrasi (atau rollback) dari IPv4 ke tipe IP Discovery IPv6 tanpa waktu henti.

### Klaster ElastiCache tumpukan ganda dengan TLS diaktifkan

Ketika TLS diaktifkan untuk klaster ElastiCache, fungsi penemuan klaster (`cluster slots`, `cluster shards`, dan `cluster nodes`) akan menampilkan nama host, bukan IP. Nama host kemudian digunakan sebagai pengganti IP untuk terhubung ke klaster ElastiCache dan melakukan handshake TLS. Ini berarti bahwa klien tidak akan terpengaruh oleh parameter IP Discovery. Untuk klaster dengan TLS diaktifkan, parameter IP Discovery tidak berpengaruh pada protokol IP yang disukai. Sebagai gantinya, protokol IP yang digunakan akan ditentukan oleh protokol IP mana yang lebih disukai klien saat menyelesaikan nama host DNS.

Untuk contoh tentang cara mengonfigurasi preferensi protokol IP saat menyelesaikan nama host DNS, lihat TLS mengaktifkan cluster tumpukan ElastiCache ganda.

### Menggunakan AWS Management Console

Saat membuat klaster menggunakan AWS Management Console, di bawah Konektivitas, pilih jenis jaringan, baik IPv4, IPv6 atau Tumpukan ganda. Jika Anda membuat klaster Redis (mode klaster diaktifkan) dan memilih tumpukan ganda, Anda harus memilih jenis IP Discovery, baik IPv6 atau IPv4.

Untuk informasi lebih lanjut, lihat Membuat klaster Redis (mode klaster diaktifkan) (Konsol) atau Membuat Redis (mode klaster dinonaktifkan) (Konsol).

Saat membuat grup replikasi menggunakan AWS Management Console, pilih jenis jaringan, baik IPv4, IPv6 atau Tumpukan ganda. Jika Anda memilih tumpukan ganda, Anda harus memilih jenis IP Discovery, baik IPv6 atau IPv4.

Untuk informasi lebih lanjut, lihat Membuat grup replikasi Redis (Mode Klaster Dinonaktifkan) dari awal atau Membuat grup replikasi di Redis (Mode Klaster Diaktifkan) dari awal.

### Menggunakan CLI

Saat membuat klaster cache menggunakan CLI, Anda menggunakan perintah create-cache-cluster dan menentukan parameter `NetworkType` dan `IPDiscovery`:

Untuk Linux, macOS, atau Unix:

```
aws elasticache create-cache-cluster \
```

```
    --cache-cluster-id "cluster-test" \
    --engine redis \
    --cache-node-type cache.m5.large \
    --num-cache-nodes 1 \
    --network-type dual_stack \
    --ip-discovery ipv4
```

Untuk Windows:

```
aws elasticache create-cache-cluster ^
    --cache-cluster-id "cluster-test" ^
    --engine redis ^
    --cache-node-type cache.m5.large ^
    --num-cache-nodes 1 ^
    --network-type dual_stack ^
    --ip-discovery ipv4
```

Saat membuat grup replikasi dengan mode cluster dinonaktifkan menggunakan CLI, Anda menggunakan perintah create-replication-group dan menentukan parameter `NetworkType` dan `IPDiscovery`:

Untuk Linux, macOS, atau Unix:

```
aws elasticache create-replication-group \
   --replication-group-id sample-repl-group \
   --replication-group-description "demo cluster with replicas" \
   --num-cache-clusters 3 \
   --primary-cluster-id redis01 \
   --network-type dual_stack \
   --ip-discovery ipv4
```

Untuk Windows:

```
aws elasticache create-replication-group ^
   --replication-group-id sample-repl-group ^
   --replication-group-description "demo cluster with replicas" ^
   --num-cache-clusters 3 ^
   --primary-cluster-id redis01 ^
```

```
   --network-type dual_stack ^
   --ip-discovery ipv4
```

Saat membuat grup replikasi dengan mode klaster diaktifkan dan menggunakan IPv4 untuk penemuan IP menggunakan CLI, Anda menggunakan perintah create-replication-group dan menentukan parameter `NetworkType` dan `IPDiscovery`:

Untuk Linux, macOS, atau Unix:

```
aws elasticache create-replication-group \
  --replication-group-id demo-cluster \
  --replication-group-description "demo cluster" \
  --cache-node-type cache.m5.large \
  --num-node-groups 2 \
  --engine redis \
  --cache-subnet-group-name xyz \
  --network-type dual_stack \
  --ip-discovery ipv4 \
  --region us-east-1
```

Untuk Windows:

```
aws elasticache create-replication-group ^
  --replication-group-id demo-cluster ^
  --replication-group-description "demo cluster" ^
  --cache-node-type cache.m5.large ^
  --num-node-groups 2 ^
  --engine redis ^
  --cache-subnet-group-name xyz ^
  --network-type dual_stack ^
  --ip-discovery ipv4 ^
  --region us-east-1
```

Saat membuat grup replikasi dengan mode klaster diaktifkan dan menggunakan IPv6 untuk penemuan IP menggunakan CLI, Anda menggunakan perintah create-replication-group dan menentukan parameter `NetworkType` dan `IPDiscovery`:

Untuk Linux, macOS, atau Unix:

```
aws elasticache create-replication-group \
  --replication-group-id demo-cluster \
```

```
  --replication-group-description "demo cluster" \
  --cache-node-type cache.m5.large \
  --num-node-groups 2 \
  --engine redis \
  --cache-subnet-group-name xyz \
  --network-type dual_stack \
  --ip-discovery ipv6 \
  --region us-east-1
```

Untuk Windows:

```
aws elasticache create-replication-group ^
  --replication-group-id demo-cluster ^
  --replication-group-description "demo cluster" ^
  --cache-node-type cache.m5.large ^
  --num-node-groups 2 ^
  --engine redis ^
  --cache-subnet-group-name xyz ^
  --network-type dual_stack ^
  --ip-discovery ipv6 ^
  --region us-east-1
```

## Tingkatan data

Klaster yang terdiri dari grup replikasi dan menggunakan jenis simpul dari rangkaian r6gd akan memiliki data yang diberi tingkatan antara penyimpanan memori dan SSD (solid state drive) lokal. Tingkatan data menyediakan opsi harga-performa baru untuk beban kerja Redis dengan memanfaatkan solid state drive (SSD) yang berbiaya lebih rendah di setiap simpul klaster selain menyimpan data dalam memori. Hal ini sangat ideal untuk beban kerja yang mengakses hingga 20 persen dari keseluruhan set data mereka secara rutin, dan untuk aplikasi yang dapat menoleransi latensi tambahan saat mengakses data di SSD.

Pada cluster dengan tiering data, ElastiCache memantau waktu akses terakhir dari setiap item yang disimpannya. Ketika memori yang tersedia (DRAM) sepenuhnya dikonsumsi, ElastiCache menggunakan algoritma yang paling tidak baru digunakan (LRU) untuk secara otomatis memindahkan item yang jarang diakses dari memori ke SSD. Ketika data pada SSD kemudian diakses, ElastiCache secara otomatis dan asinkron memindahkannya kembali ke memori sebelum memproses permintaan. Jika Anda memiliki beban kerja yang mengakses hanya subset dari datanya secara rutin, tingkatan data adalah cara optimal untuk menskalakan kapasitas Anda dengan hemat biaya.

Perhatikan bahwa saat menggunakan tingkatan data, kuncinya itu sendiri selalu tetap dalam memori, sedangkan LRU mengatur penempatan nilai pada memori vs. disk. Secara umum, sebaiknya ukuran kunci Anda lebih kecil dari ukuran nilai Anda saat menggunakan tingkatan data.

Tingkatan data dirancang untuk memiliki dampak performa minimal pada beban kerja aplikasi. Misalnya, anggaplah ada nilai String 500-byte, Anda dapat mengalami 300 mikrodetik tambahan latensi rata-rata untuk permintaan terhadap data yang tersimpan di SSD dibandingkan dengan permintaan terhadap data dalam memori.

Dengan ukuran simpul tingkatan data terbesar (cache.r6gd.16xlarge), Anda dapat menyimpan hingga 1 petabyte dalam satu klaster 500 simpul (500 TB saat menggunakan 1 replika baca). Tiering data kompatibel dengan semua perintah Redis dan struktur data yang didukung di. ElastiCache Anda tidak memerlukan perubahan sisi klien untuk menggunakan fitur ini.

Topik



## Praktik terbaik

Kami merekomendasikan praktik terbaik berikut:

- Tingkatan data sangat ideal untuk beban kerja yang mengakses hingga 20 persen dari keseluruhan set data mereka secara rutin, dan untuk aplikasi yang dapat menoleransi latensi tambahan saat mengakses data di SSD.
- Saat menggunakan kapasitas SSD yang tersedia pada simpul tingkatan data, kami menyarankan agar ukuran nilai lebih besar dari ukuran kunci. Ketika item dipindahkan antara DRAM dan SSD, kunci ini akan selalu tetap dalam memori dan hanya nilai yang dipindahkan ke tingkat SSD.

## Batasan

Tingkatan data memiliki batasan berikut:

- Anda hanya dapat menggunakan tingkatan data pada klaster yang merupakan bagian dari grup replikasi.
- Jenis simpul yang Anda gunakan harus berasal dari rangkaian r6gd, yang tersedia di wilayah berikut: `us-east-2`, `us-east-1`, `us-west-2`, `us-west-1`, `eu-west-1`, `eu-central-1`, `eu-north-1`, `eu-west-3`, `ap-northeast-1`, `ap-southeast-1`, `ap-southeast-2`, `ap-south-1`, `ca-central-1`, dan `sa-east-1`.
- Anda harus menggunakan mesin Redis 6.2 atau yang lebih baru.
- Anda tidak dapat memulihkan cadangan klaster r6gd ke klaster lain kecuali klaster lain tersebut menggunakan r6gd juga.
- Anda tidak dapat mengekspor cadangan ke Amazon S3 untuk klaster tingkatan data.
- Migrasi online tidak didukung untuk klaster yang berjalan pada jenis simpul r6gd.
- Penskalaan tidak didukung dari klaster tingkatan data (misalnya, sebuah klaster yang menggunakan jenis simpul r6gd) ke klaster yang tidak menggunakan tingkatan data (misalnya, klaster yang menggunakan jenis simpul r6g). Untuk informasi selengkapnya, lihat Penskalaan ElastiCache untuk Redis .
- Penskalaan otomatis didukung pada klaster menggunakan tingkatan data untuk Redis versi 7.0.7 dan yang lebih baru. Lihat informasi yang lebih lengkap di Auto Scaling ElastiCache untuk kluster Redis
- Tingkatan data hanya mendukung kebijakan maxmemory `volatile-lru`, `allkeys-lru`, `volatile-lfu`, `allkeys-lfu`, dan `noeviction`.
- Penyimpanan forkless didukung untuk Redis versi 7.0.7 dan yang lebih baru. Untuk informasi selengkapnya, lihat Cara penerapan sinkronisasi dan pencadangan.
- Item yang lebih besar dari 128 MiB tidak dipindahkan ke SSD.

## Harga

Simpul R6gd memiliki kapasitas total 4,8x lebih banyak (memori + SSD) dan dapat membantu Anda mencapai lebih dari 60 persen penghematan ketika berjalan pada pemanfaatan maksimum dibandingkan dengan simpul R6g (memori saja). Untuk informasi lebih lanjut, lihat ElastiCache harga.

## Pemantauan

ElastiCache untuk Redis menawarkan metrik yang dirancang khusus untuk memantau kluster kinerja yang menggunakan tingkatan data. Untuk memantau rasio item dalam DRAM dibandingkan dengan SSD, Anda dapat menggunakan metrik `CurrItems` dalam Metrik untuk Redis. Anda dapat

menghitung persentase sebagai: (CurrItems dengan Dimensi: Tingkat = Memori* 100)/(CurrItems tanpa filter dimensi). Jika persentase item dalam memori berkurang di bawah 5 persen, sebaiknya pertimbangkan untuk melakukan penskalaan ke luar untuk klaster Mode Klaster Diaktifkan atau menaikkan skala untuk klaster Mode Klaster Dinonaktifkan. Untuk informasi selengkapnya, lihat Metrik untuk klaster Redis yang menggunakan tingkatan data di Metrik untuk Redis.

## Menggunakan tingkatan data

Menggunakan data tiering menggunakan AWS Management Console

Ketika membuat sebuah klaster sebagai bagian dari grup replikasi, Anda akan menggunakan tingkatan data dengan memilih jenis simpul dari rangkaian r6gd, seperti cache.r6gd.xlarge. Pemilihan jenis simpul tersebut secara otomatis mengaktifkan tingkatan data.

Untuk informasi selengkapnya tentang cara membuat klaster, lihat Membuat klaster.

Mengaktifkan tiering data menggunakan AWS CLI

Saat membuat grup replikasi menggunakan AWS CLI, Anda menggunakan tiering data dengan memilih tipe node dari keluarga r6gd, seperti cache.r6gd.xlarge dan menyetel parameternya. `--data-tiering-enabled`

Anda tidak dapat membatalkan penggunaan tingkatan data ketika memilih jenis simpul dari rangkaian r6gd. Jika Anda mengatur parameter `--no-data-tiering-enabled`, operasi akan gagal.

Untuk Linux, macOS, atau Unix:

```
aws elasticache create-replication-group \
   --replication-group-id redis-dt-cluster \
   --replication-group-description "Redis cluster with data tiering" \
   --num-node-groups 1 \
   --replicas-per-node-group 1 \
   --cache-node-type cache.r6gd.xlarge \
   --engine redis \
   --cache-subnet-group-name default \
   --automatic-failover-enabled \
   --data-tiering-enabled
```

Untuk Windows:

```
aws elasticache create-replication-group ^
   --replication-group-id redis-dt-cluster ^
```

```
    --replication-group-description "Redis cluster with data tiering" ^
    --num-node-groups 1 ^
    --replicas-per-node-group 1 ^
    --cache-node-type cache.r6gd.xlarge ^
    --engine redis ^
    --cache-subnet-group-name default ^
    --automatic-failover-enabled ^
    --data-tiering-enabled
```

Setelah menjalankan operasi ini, Anda akan melihat respons seperti yang berikut ini:

```
{
    "ReplicationGroup": {
        "ReplicationGroupId": "redis-dt-cluster",
        "Description": "Redis cluster with data tiering",
        "Status": "creating",
        "PendingModifiedValues": {},
        "MemberClusters": [
            "redis-dt-cluster"
        ],
        "AutomaticFailover": "enabled",
        "DataTiering": "enabled",
        "SnapshotRetentionLimit": 0,
        "SnapshotWindow": "06:00-07:00",
        "ClusterEnabled": false,
        "CacheNodeType": "cache.r6gd.xlarge",
        "TransitEncryptionEnabled": false,
        "AtRestEncryptionEnabled": false
    }
}
```

## Memulihkan data dari cadangan ke dalam klaster dengan tingkatan data diaktifkan

Anda dapat memulihkan cadangan ke klaster baru dengan tiering data yang diaktifkan menggunakan (Console), (AWS CLI) atau (ElastiCache API). Ketika Anda membuat sebuah klaster menggunakan jenis simpul dalam rangkaian r6gd, tingkatan data diaktifkan.

Memulihkan data dari cadangan ke dalam klaster dengan tingkatan data diaktifkan (konsol)

Untuk memulihkan cadangan ke klaster baru dengan tingkatan data diaktifkan (konsol)

1. Masuk ke AWS Management Console dan buka ElastiCache konsol di https://console.aws.amazon.com/elasticache/.

2. Dari panel navigasi, pilih Cadangan.
3. Di daftar cadangan, pilih kotak di sebelah kiri nama cadangan yang ingin Anda pulihkan.
4. Pilih Pulihkan.
5. Lengkapi kotak dialog Pulihkan Klaster. Pastikan untuk melengkapi semua bidang Wajib dan semua bidang yang ingin Anda ubah dari default.

    1. ID Klaster – Wajib. Nama klaster baru.
    2. Mode klaster diaktifkan (penskalaan ke luar) – Pilih opsi ini untuk klaster Redis (mode klaster diaktifkan).
    3. Jenis Simpul – Tentukan cache.r6gd.xlarge atau jenis simpul lainnya dari rangkaian r6gd.
    4. Jumlah Serpihan – Memilih jumlah serpihan yang Anda inginkan di klaster baru (API/CLI: grup simpul).
    5. Replika per Serpihan – Pilih jumlah simpul replika baca yang Anda inginkan di setiap serpihan.
    6. Slot dan keyspace – Pilih cara yang Anda inginkan untuk mendistribusikan kunci di antara serpihan. Jika Anda memilih untuk menentukan distribusi kunci, lengkapi tabel yang menentukan rentang kunci untuk setiap serpihan.
    7. Zona ketersediaan – Tentukan cara yang Anda inginkan untuk memilih Zona Ketersediaan dari klaster.
    8. Port – Ubah opsi ini hanya jika Anda menginginkan klaster baru menggunakan port yang berbeda.
    9. Pilih VPC – Pilih VPC tempat klaster ini akan dibuat.
    10. Grup Parameter – Pilih grup parameter yang mencadangkan memori yang cukup untuk overhead Redis untuk jenis simpul yang Anda pilih.
6. Jika pengaturan sudah sesuai keinginan Anda, pilih Buat.

Untuk informasi selengkapnya tentang cara membuat klaster, lihat Membuat klaster.

Memulihkan data dari cadangan ke dalam klaster dengan tingkatan data diaktifkan (AWS CLI)

Saat membuat grup replikasi menggunakan AWS CLI, tiering data secara default digunakan dengan memilih tipe node dari keluarga r6gd, seperti cache.r6gd.xlarge dan mengatur parameter. `--data-tiering-enabled`

Anda tidak dapat membatalkan penggunaan tingkatan data ketika memilih jenis simpul dari rangkaian r6gd. Jika Anda mengatur parameter `--no-data-tiering-enabled`, operasi akan gagal.

Untuk Linux, macOS, atau Unix:

```
aws elasticache create-replication-group \
   --replication-group-id redis-dt-cluster \
   --replication-group-description "Redis cluster with data tiering" \
   --num-node-groups 1 \
   --replicas-per-node-group 1 \
   --cache-node-type cache.r6gd.xlarge \
   --engine redis \
   --cache-subnet-group-name default \
   --automatic-failover-enabled \
   --data-tiering-enabled \
   --snapshot-name my-snapshot
```

Untuk Linux, macOS, atau Unix:

```
aws elasticache create-replication-group ^
   --replication-group-id redis-dt-cluster ^
   --replication-group-description "Redis cluster with data tiering" ^
   --num-node-groups 1 ^
   --replicas-per-node-group 1 ^
   --cache-node-type cache.r6gd.xlarge ^
   --engine redis ^
   --cache-subnet-group-name default ^
   --automatic-failover-enabled ^
   --data-tiering-enabled ^
   --snapshot-name my-snapshot
```

Setelah menjalankan operasi ini, Anda akan melihat respons seperti yang berikut ini:

```
{
    "ReplicationGroup": {
        "ReplicationGroupId": "redis-dt-cluster",
        "Description": "Redis cluster with data tiering",
        "Status": "creating",
        "PendingModifiedValues": {},
        "MemberClusters": [
            "redis-dt-cluster"
        ],
        "AutomaticFailover": "enabled",
        "DataTiering": "enabled",
        "SnapshotRetentionLimit": 0,
        "SnapshotWindow": "06:00-07:00",
```

```
        "ClusterEnabled": false,
        "CacheNodeType": "cache.r6gd.xlarge",
        "TransitEncryptionEnabled": false,
        "AtRestEncryptionEnabled": false
    }
}
```

# Menyiapkan klaster

Berikut ini, terdapat petunjuk tentang cara membuat klaster menggunakan konsol ElastiCache, AWS CLI, atau API ElastiCache.

Anda juga dapat membuat klaster ElastiCache menggunakan AWS CloudFormation. Untuk informasi selengkapnya, lihat AWS::ElastiCache::CacheCluster di Panduan Pengguna AWS Cloud Formation, ang mencakup panduan tentang cara menerapkan pendekatan tersebut.

Setiap kali Anda membuat klaster atau grup replikasi, sebaiknya lakukan pekerjaan persiapan tertentu agar Anda tidak perlu melakukan peningkatan atau melakukan perubahan.

Topik

- Menentukan kebutuhan Anda
- Memilih ukuran simpul Anda

## Menentukan kebutuhan Anda

### Persiapan

Mengetahui jawaban atas pertanyaan berikut membantu memudahkan proses pembuatan klaster Anda:

- Jenis instans simpul apa yang dibutuhkan?

  Untuk panduan terkait cara memilih jenis simpul instans, lihat Memilih ukuran simpul Anda.
- Apakah klaster Anda diluncurkan di cloud privat virtual (VPC) berdasarkan Amazon VPC?

> **Important**
>
> Jika Anda akan meluncurkan klaster di VPC, pastikan untuk membuat grup subnet di VPC yang sama sebelum Anda mulai membuat klaster. Untuk informasi selengkapnya, lihat Subnet dan grup subnet.

ElastiCache dirancang untuk diakses dari dalam AWS menggunakan Amazon EC2. Namun, jika Anda meluncurkan di VPC berdasarkan Amazon VPC dan klaster Anda berada di VPC, Anda dapat menyediakan akses dari luar AWS. Untuk informasi selengkapnya, lihat Mengakses sumber daya ElastiCache dari luar AWS.

- Apakah Anda perlu menyesuaikan nilai parameter tertentu?

  Jika perlu, buat grup parameter khusus. Untuk informasi selengkapnya, lihat Membuat grup parameter.

  Jika Anda menjalankan Redis, pertimbangkan untuk mengatur `reserved-memory` atau `reserved-memory-percent`. Untuk informasi selengkapnya, lihat Mengelola Memori Terpesan.
- Apakah Anda perlu membuat grup keamanan VPC Anda sendiri?

  Untuk informasi selengkapnya, lihat Keamanan di VPC Anda.
- Bagaimana Anda akan menerapakan toleransi kesalahan?

  Untuk informasi selengkapnya, lihat Mitigasi Kegagalan.

Topik



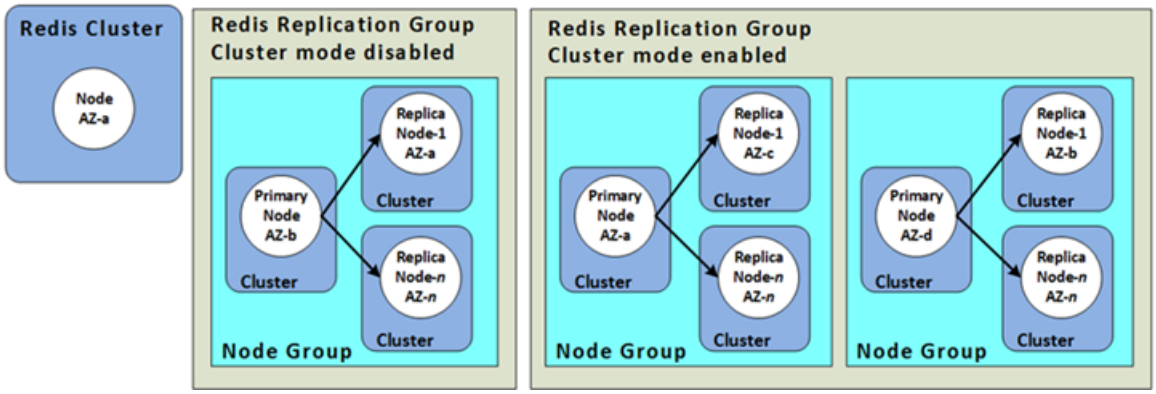
## Persyaratan memori dan prosesor

Blok bangunan dasar Amazon ElastiCache adalah simpul. Simpul dikonfigurasi secara tunggal atau dalam pengelompokan untuk membentuk klaster. Saat menentukan jenis simpul yang akan digunakan untuk klaster Anda, pertimbangkan konfigurasi simpul klaster dan jumlah data yang harus disimpan.

## Konfigurasi klaster Redis

ElastiCache untuk kluster Redis terdiri dari 0 hingga 500 pecahan (juga disebut grup simpul). Data dalam klaster Redis dipartisi di seluruh serpihan di klaster. Aplikasi Anda terhubung klaster Redis

menggunakan alamat jaringan yang disebut Titik Akhir. Simpul dalam serpihan Redis memenuhi salah satu dari dua peran: satu berperan sebagai primer baca/tulis dan semua simpul lain berperan sebagai sekunder hanya-baca (juga disebut replika baca). Selain titik akhir simpul, klaster Redis itu sendiri memiliki titik akhir yang disebut titik akhir konfigurasi. Aplikasi Anda dapat menggunakan endpoint ini untuk membaca dari atau menulis ke cluster, meninggalkan penentuan node mana yang akan dibaca atau ditulis hingga ElastiCache Redis.




Untuk informasi selengkapnya, lihat Mengelola klaster.

Persyaratan penskalaan

Semua klaster dapat dinaikkan skalanya dengan membuat klaster baru yang memiliki jenis simpul baru yang lebih besar. Ketika Anda menaikkan skala klaster Redis, Anda dapat menyemai klaster tersebut dari cadangan dan menghindari klaster baru dimulai dalam keadaan kosong.

Untuk informasi selengkapnya, lihat Penskalaan ElastiCache untuk Redis dalam panduan ini.

Persyaratan akses

Secara desain, ElastiCache kluster Amazon diakses dari instans Amazon EC2. Akses jaringan ke ElastiCache cluster terbatas pada akun yang membuat cluster. Oleh karena itu, sebelum Anda dapat mengakses klaster dari instans Amazon EC2, Anda harus memberikan otorisasi pada instans Amazon EC2 untuk mengakses klaster tersebut. Langkah untuk melakukan hal ini bervariasi, tergantung apakah Anda meluncurkan ke EC2-VPC atau EC2-Classic.

Jika Anda meluncurkan klaster Anda ke EC2-VPC, Anda perlu memberikan izin masuk jaringan ke klaster tersebut. Jika meluncurkan klaster ke EC2-Classic, Anda harus memberikan grup keamanan Amazon Elastic Compute Cloud yang terkait dengan instans akses ke grup keamanan Anda ElastiCache . Untuk instruksi yang lebih lengkap, lihat Langkah 3: Mengizinkan akses ke klaster dalam panduan ini.

Persyaratan Wilayah, Availability Zone, dan Zona Lokal

Amazon ElastiCache mendukung semua AWS wilayah. Dengan menempatkan ElastiCache kluster Anda di AWS Wilayah yang dekat dengan aplikasi Anda, Anda dapat mengurangi latensi. Jika klaster Anda memiliki beberapa simpul, menemukan simpul Anda di Availability Zone atau di Local Zones yang berbeda dapat mengurangi dampak kegagalan pada klaster Anda.

Untuk informasi selengkapnya tentang IAM, lihat hal berikut:

- Memilih wilayah dan zona ketersediaan
- Menggunakan zona lokal dengan ElastiCache
- Mitigasi Kegagalan

## Memilih ukuran simpul Anda

Ukuran simpul yang Anda pilih untuk klaster Anda memengaruhi biaya, performa, dan toleransi kesalahan.

Memilih ukuran simpul Anda

Untuk informasi tentang manfaat prosesor Graviton, lihat Prosesor AWS Graviton.

Menjawab pertanyaan-pertanyaan berikut dapat membantu Anda menentukan jenis simpul minimum yang Anda butuhkan untuk implementasi Redis Anda:

- Apakah Anda mengharapkan beban kerja terikat throughput dengan beberapa koneksi klien?

  Jika ya dan Anda menjalankan Redis versi 5.0.6 atau lebih tinggi, Anda bisa mendapatkan throughput dan latensi yang lebih baik dengan fitur I/O kami yang disempurnakan, yang memungkinkan CPU yang tersedia digunakan untuk mengalihkan beban koneksi klien, untuk mendukung mesin Redis. Jika Anda menjalankan Redis versi 7.0.4 atau lebih tinggi, selain I/O yang disempurnakan, Anda akan mendapatkan akselerasi tambahan dengan multiplexing I/O yang disempurnakan. Hal ini memungkinkan setiap thread IO khusus melakukan pipelining terhadap perintah dari beberapa klien ke mesin Redis, dengan memanfaatkan kemampuan Redis untuk memproses perintah secara efisien dalam batch. DI ElastiCache for Redis v7.1 dan lebih tinggi, kami memperluas fungsionalitas thread I/O yang disempurnakan agar juga menangani logika lapisan presentasi. Yang kami maksud dengan lapisan presentasi adalah bahwa thread I/O yang disempurnakan sekarang tidak hanya membaca input klien, tetapi juga mengurai input ke

dalam format perintah biner Redis, yang kemudian diteruskan ke thread utama untuk dieksekusi, sehingga memberikan peningkatan performa. Lihat postingan blog dan halaman versi yang didukung untuk detail tambahan.

- Apakah Anda memiliki beban kerja yang mengakses sebagian kecil dari datanya secara berkala?

  Jika ya dan Anda menjalankan beban kerja di Redis mesin versi 6.2 atau yang lebih baru, Anda dapat memanfaatkan tingkatan data dengan memilih jenis simpul r6gd. Dengan tingkatan data, data yang sudah lama tidak digunakan akan disimpan di SSD. Ketika data ini diambil, ada sedikit latensi, tetapi diimbangi dengan penghematan biaya. Untuk informasi selengkapnya, lihat Tingkatan data.

  Untuk informasi selengkapnya, lihat Tipe simpul yang didukung.

- Berapa banyak memori total yang dibutuhkan untuk data Anda?

  Untuk mendapatkan estimasi umum, cari tahu ukuran item yang ingin Anda simpan dalam cache. Kalikan ukuran ini dengan jumlah item yang ingin Anda simpan dalam cache pada waktu yang sama. Untuk mendapatkan estimasi yang wajar terkait ukuran item, pertama-tama serialisasikan item cache Anda, kemudian hitung karakternya. Kemudian bagi jumlah karakter dengan jumlah serpihan dalam klaster Anda.

  Untuk informasi selengkapnya, lihat Tipe simpul yang didukung.

- Apa versi Redis yang Anda jalankan?

  Redis versi sebelum 2.8.22 mengharuskan Anda untuk mencadangkan lebih banyak memori untuk operasi failover, snapshot, sinkronisasi, dan promosi replika ke primer. Persyaratan ini muncul karena Anda harus memiliki cukup memori yang tersedia untuk semua penulisan yang terjadi selama proses.

  Redis versi 2.8.22 dan lebih baru menggunakan proses penyimpanan forkless yang membutuhkan ketersediaan memori yang lebih kecil dibandingkan proses sebelumnya.

  Untuk informasi selengkapnya tentang IAM, lihat hal berikut:

  - Cara penerapan sinkronisasi dan pencadangan
  - Memastikan bahwa Anda memiliki cukup memori untuk membuat snapshot Redis

- Seberapa berat proses tulis aplikasi Anda?

  Aplikasi dengan proses tulis berat dapat membutuhkan jauh lebih banyak memori yang tersedia, yaitu memori yang tidak digunakan oleh data, saat mengambil snapshot atau failover. Setiap

kali proses BGSAVE dilakukan, Anda harus memiliki cukup memori yang tidak digunakan oleh data untuk mengakomodasi semua proses tulis yang berlangsung selama proses BGSAVE. Contohnya adalah ketika mengambil snapshot, ketika menyinkronkan klaster primer dengan replika dalam klaster, dan ketika mengaktifkan fitur append-only file (AOF). Contoh lainnya adalah saat mempromosikan replika ke primer (jika Anda memiliki Multi-AZ yang aktif). Kemungkinan terburuknya adalah ketika semua data Anda ditulis ulang selama proses berlangsung. Dalam hal ini, Anda memerlukan ukuran instans simpul dengan memori dua kali lebih banyak dari yang diperlukan untuk data saja.

Untuk informasi selengkapnya, lihat Memastikan bahwa Anda memiliki cukup memori untuk membuat snapshot Redis.

- Apakah implementasi Anda akan menjadi klaster Redis mandiri (mode klaster dinonaktifkan) atau klaster Redis (mode klaster diaktifkan) dengan beberapa serpihan?

  Klaster Redis (mode klaster dinonaktifkan)

  Jika Anda mengimplementasikan klaster Redis (mode klaster dinonaktifkan), jenis simpul Anda harus mampu mengakomodasi semua data Anda ditambah overhead yang diperlukan seperti yang dijelaskan dalam poin sebelumnya.

  Sebagai contoh, misalnya Anda memperkirakan bahwa total ukuran semua item Anda adalah 12 GB. Dalam kasus ini, Anda dapat menggunakan simpul `cache.m3.xlarge` dengan memori 13,3 GB atau simpul `cache.r3.large` dengan memori 13,5 GB. Namun, Anda mungkin memerlukan lebih banyak memori untuk operasi BGSAVE. Jika aplikasi Anda memiliki proses tulis berat, gandakan kebutuhan memori menjadi setidaknya 24 GB. Dengan demikian, gunakan `cache.m3.2xlarge` dengan memori 27,9 GB atau `cache.r3.xlarge` dengan memori 30,5 GB.

  Redis (mode klaster diaktifkan) dengan beberapa serpihan

  Jika Anda mengimplementasikan klaster Redis (mode klaster diaktifkan) dengan beberapa serpihan, maka jenis simpulnya harus mampu mengakomodasi `bytes-for-data-and-overhead / number-of-shards` byte data.

  Sebagai contoh, misalnya Anda memperkirakan bahwa total ukuran semua item Anda adalah 12 GB dan Anda memiliki dua serpihan. Dalam kasus ini, Anda dapat menggunakan simpul `cache.m3.large` dengan memori 6,05 GB (12 GB / 2). Namun, Anda mungkin memerlukan lebih banyak memori untuk operasi BGSAVE. Jika aplikasi Anda memiliki proses tulis berat, gandakan kebutuhan memori menjadi setidaknya 12 GB per serpihan. Dengan demikian, gunakan `cache.m3.xlarge` dengan memori 13,3 GB atau `cache.r3.large` dengan memori 13,5 GB.

- Apakah Anda menggunakan Zona Lokal?

  Zona Lokal memungkinkan Anda menempatkan sumber daya seperti klaster ElastiCache dalam beberapa lokasi yang dekat dengan pengguna Anda. Namun, ketika Anda memilih ukuran simpul Anda, perhatikan bahwa ukuran simpul yang tersedia saat ini terbatas seperti yang tertera berikut ini, terlepas dari persyaratan kapasitas:

  - Generasi saat ini:

    Tipe simpul M5: `cache.m5.large`, `cache.m5.xlarge`, `cache.m5.2xlarge`, `cache.m5.4xlarge`, `cache.m5.12xlarge`, `cache.m5.24xlarge`

    Tipe simpul R5: `cache.r5.large`, `cache.r5.xlarge`, `cache.r5.2xlarge`, `cache.r5.4xlarge`, `cache.r5.12xlarge`, `cache.r5.24xlarge`

    Tipe simpul T3: `cache.t3.micro`, `cache.t3.small`, `cache.t3.medium`

Saat klaster berjalan, Anda dapat memantau metrik penggunaan memori, pemanfaatan prosesor, cache hit, dan cache miss yang diterbitkan ke CloudWatch. Anda mungkin memperhatikan bahwa klaster Anda tidak memiliki laju hit yang Anda inginkan atau bahwa kunci terlalu sering dikosongkan. Dalam kasus ini, Anda dapat memilih ukuran simpul yang berbeda dengan spesifikasi CPU dan memori yang lebih besar.

Saat memantau penggunaan CPU, ingat bahwa Redis memiliki thread tunggal. Dengan demikian, kalikan penggunaan CPU yang dilaporkan dengan jumlah inti CPU untuk mengetahui penggunaan yang sebenarnya. Sebagai contoh, CPU empat inti yang melaporkan tingkat penggunaan 20 persen sebenarnya adalah Redis satu inti yang berjalan pada 80 persen pemanfaatan.

# Membuat klaster

Contoh berikut menunjukkan cara membuat cluster Redis menggunakan AWS Management Console, AWS CLI dan ElastiCache API.

Membuat Redis (mode klaster dinonaktifkan) (Konsol)

ElastiCache mendukung replikasi saat Anda menggunakan mesin Redis. Untuk memantau latensi antara saat data ditulis ke cluster primer baca/tulis Redis dan ketika disebarkan ke cluster sekunder hanya-baca, ElastiCache tambahkan ke cluster kunci khusus,. `ElastiCacheMasterReplicationTimestamp` Kunci ini adalah waktu Universal Universal (UTC) saat ini. Karena klaster Redis dapat ditambahkan ke grup replikasi pada lain waktu, kunci ini dimasukkan ke dalam semua klaster Redis, meskipun jika awalnya klaster tersebut bukan anggota grup replikasi. Untuk informasi selengkapnya tentang grup replikasi, lihat Ketersediaan tinggi menggunakan grup replikasi.

Untuk membuat klaster Redis (mode klaster dinonaktifkan), ikuti langkah di Membuat klaster Redis (Mode Klaster Dinonaktifkan) (Konsol).

Setelah status klaster Anda menjadi tersedia, Anda dapat memberikan akses ke Amazon EC2, menyambungkannya, dan mulai menggunakannya. Untuk informasi selengkapnya, lihat Langkah 3: Mengizinkan akses ke klaster dan Langkah 4: Menyambung ke simpul klaster.

> **Important**
>
> Setelah klaster Anda tersedia, Anda akan ditagih untuk setiap jam atau durasi saat klaster aktif, meskipun Anda tidak sedang aktif menggunakannya. Untuk menghentikan tagihan biaya untuk klaster ini, Anda harus menghapusnya. Lihat Menghapus klaster.

Membuat klaster Redis (mode klaster diaktifkan) (Konsol)

Jika Anda menjalankan Redis 3.2.4 atau yang lebih baru, Anda dapat membuat sebuah klaster Redis (mode klaster diaktifkan). Klaster Redis (mode klaster diaktifkan) mendukung partisi data di 1 sampai 500 serpihan (API/CLI: grup simpul) tetapi dengan beberapa keterbatasan. Untuk perbandingan Redis (mode klaster dinonaktifkan) dan Redis (mode klaster diaktifkan), lihat Versi ElastiCache for Redis yang Didukung.

Untuk membuat cluster Redis (mode cluster diaktifkan) menggunakan konsol ElastiCache

1. Masuk ke AWS Management Console dan buka ElastiCache konsol Amazon di https://console.aws.amazon.com/elasticache/.
2. Dari daftar di sudut kanan atas, pilih AWS Wilayah tempat Anda ingin meluncurkan cluster ini.
3. Pilih Mulai di panel navigasi.
4. Pilih Buat VPC dan ikuti langkah yang telah dijelaskan pada Membuat Cloud Privat Virtual (VPC).
5. Pada halaman ElastiCache dasbor, pilih Create cluster dan kemudian pilih Create Redis cluster.
6. Pada Pengaturan klaster, lakukan hal berikut:

   a. Pilih Konfigurasi dan buat klaster baru.

   b. Untuk Mode klaster, pilih Diaktifkan.

   c. Untuk Info klaster masukkan nilai untuk Nama.

   d. (Opsional) Masukkan nilai untuk Deskripsi.
7. Di bawah Lokasi:

   AWS Cloud

   1. Untuk AWS Cloud, sebaiknya Anda menerima pengaturan default untuk Multi-AZ dan Auto-failover. Untuk informasi selengkapnya, lihat Meminimalkan waktu henti ElastiCache untuk Redis dengan Multi-AZ.
   2. Di bawah Pengaturan klaster

      a. Untuk Versi mesin, pilih versi yang tersedia.

      b. Untuk Port, gunakan port default, 6379. Jika Anda ingin menggunakan port yang berbeda, tuliskan nomor port tersebut.

      c. Untuk Grup parameter, pilih grup parameter atau buat yang baru. Grup parameter mengontrol parameter runtime klaster Anda. Untuk informasi selengkapnya tentang grup parameter, lihat Parameter spesifik Redis dan Membuat grup parameter.

      > **Note**
      >
      > Saat Anda memilih grup parameter untuk menetapkan nilai konfigurasi mesin, grup parameter tersebut diterapkan ke semua klaster di penyimpanan data global. Pada halaman Grup Parameter, atribut Global ya/

tidak menunjukkan apakah grup parameter adalah bagian dari penyimpanan data global.

d. Untuk Jenis Simpul, pilih panah bawah
(▼ ).
Pada kotak dialog Ubah jenis simpul, pilih nilai untuk Keluarga instans untuk jenis simpul yang Anda inginkan. Kemudian pilih jenis simpul yang ingin Anda gunakan untuk klaster ini, lalu pilih Simpan.

Untuk informasi selengkapnya, lihat Memilih ukuran simpul Anda.

Jika Anda memilih jenis simpul r6gd, tingkatan data diaktifkan secara otomatis. Untuk informasi selengkapnya, lihat Tingkatan data.

e. Untuk Jumlah serpihan, pilih jumlah serpihan (partisi/grup simpul) yang Anda inginkan untuk klaster Redis (mode klaster aktif) ini.

Untuk beberapa versi Redis (mode klaster aktif), Anda dapat mengubah jumlah serpihan di klaster Anda secara dinamis:

- Redis 3.2.10 dan yang lebih baru – Jika klaster Anda menjalankan Redis 3.2.10 atau versi di atasnya, Anda dapat mengubah jumlah serpihan di klaster Anda secara dinamis. Untuk informasi selengkapnya, lihat Penskalaan klaster di Redis (Mode Klaster Diaktifkan).
- Redis versi lainnya – Jika klaster Anda menjalankan versi Redis sebelum versi 3.2.10, ada pendekatan lain. Untuk mengubah jumlah serpihan di klaster Anda, buat klaster baru dengan jumlah serpihan baru. Untuk informasi selengkapnya, lihat Melakukan pemulihan dari cadangan ke dalam cache baru.

f. Untuk Replika per serpihan, pilih jumlah simpul replika baca yang Anda inginkan dalam setiap serpihan.

Pembatasan berikut berlaku untuk Redis (mode klaster aktif).

- Jika Multi-AZ diaktifkan, pastikan bahwa Anda memiliki setidaknya satu replika per serpihan.
- Replika berjumlah sama untuk setiap serpihan saat membuat klaster menggunakan konsol.
- Jumlah replika baca per serpihan bersifat tetap dan tidak dapat diubah. Jika Anda membutuhkan lebih banyak atau lebih sedikit replika per serpihan (API/CLI: grup

simpul), Anda harus membuat klaster baru dengan jumlah replika yang baru. Untuk informasi selengkapnya, lihat Melakukan seeding klaster yang dirancang sendiri dengan cadangan yang dibuat secara eksternal.

3. Di bagian Konektivitas

   a. Untuk Jenis jaringan, pilih versi IP yang akan didukung oleh klaster ini.

   b. Untuk grup Subnet, pilih subnet yang ingin Anda terapkan ke cluster ini. ElastiCache menggunakan grup subnet itu untuk memilih subnet dan alamat IP dalam subnet itu untuk dikaitkan dengan node Anda. ElastiCache cluster memerlukan subnet dual-stack dengan alamat IPv4 dan IPv6 yang ditetapkan untuk beroperasi dalam mode dual-stack dan subnet khusus IPv6 untuk beroperasi sebagai IPv6 saja.

      Saat membuat grup subnet baru, masukkan ID VPC yang menaungi grup subnet tersebut.

      Pilih jenis IP Discovery. Hanya alamat IP protokol yang Anda pilih yang dikembalikan.

      Untuk informasi selengkapnya, lihat:

      - Memilih jenis jaringan.
      - Membuat subnet di VPC Anda.

      Jika Anda adalah Menggunakan zona lokal dengan ElastiCache , Anda harus membuat atau memilih subnet yang berada di zona lokal.

      Untuk informasi selengkapnya, lihat Subnet dan grup subnet.

4. Untuk Penempatan Zona Ketersediaan, Anda memiliki dua opsi:

   - Tidak ada preferensi — ElastiCache memilih Availability Zone.
   - Tentukan zona ketersediaan – Anda menentukan Zona Ketersediaan untuk setiap klaster.

     Jika Anda memilih untuk menentukan Zona Ketersediaan, untuk setiap klaster di setiap serpihan, pilih Zona Ketersediaan dari daftar.

Untuk informasi selengkapnya, lihat Memilih wilayah dan zona ketersediaan.

5. Pilih Selanjutnya
6. Di bawah Pengaturan lanjutan Redis

    - Untuk Keamanan:

        i. Untuk mengenkripsi data Anda, Anda memiliki opsi berikut:

        - Enkripsi saat diam – Mengaktifkan enkripsi pada data yang disimpan di disk. Untuk informasi selengkapnya, lihat Enkripsi Data Diam.

            > **Note**
            >
            > Anda memiliki opsi untuk menyediakan kunci enkripsi yang berbeda dengan memilih kunci  AWS KMS yang Dikelola Pelanggan dan memilih kunci. Untuk informasi selengkapnya, lihat Menggunakan kunci yang dikelola pelanggan dari AWS KMS.

        - Enkripsi Data Bergerak – Mengaktifkan enkripsi pada data selama pengiriman. Untuk informasi selengkapnya, lihat Enkripsi Data Bergerak. Untuk mesin Redis versi 6.0 dan di atasnya, jika Anda mengaktifkan Enkripsi data bergerak, Anda akan diminta untuk menentukan salah satu dari opsi Kontrol Akses berikut:
            - Tidak ada Kontrol Akses – Ini adalah pengaturan default. Ini menunjukkan tidak ada pembatasan akses pengguna ke klaster.
            - Daftar Kontrol Akses Grup Pengguna – Pilih grup pengguna dengan kelompok pengguna tertentu yang dapat mengakses klaster. Untuk informasi selengkapnya, lihat Mengelola Grup Pengguna dengan Konsol dan CLI.
            - Pengguna Default Redis AUTH – Mekanisme autentikasi untuk server Redis. Untuk informasi selengkapnya, silakan lihat Redis AUTH.
        - Redis AUTH - Mekanisme autentikasi untuk server Redis. Untuk informasi selengkapnya, silakan lihat Redis AUTH.

> **Note**
>
> Untuk versi Redis antara 3.2.6 dan seterusnya, kecuali versi 3.2.10, Redis AUTH adalah satu-satunya pilihan.

ii. Untuk Grup keamanan, pilih grup keamanan yang Anda inginkan untuk klaster ini. Grup keamanan bertindak sebagai firewall untuk mengontrol akses jaringan ke klaster Anda. Anda dapat menggunakan grup keamanan default untuk VPC Anda atau membuat yang baru.

Untuk informasi grup keamanan selengkapnya, lihat Grup Keamanan untuk VPC Anda di Panduan Pengguna Amazon VPC.

7. Untuk pencadangan otomatis terjadwal rutin, pilih Aktifkan pencadangan otomatis, lalu masukkan jumlah hari yang diinginkan untuk menyimpan cadangan otomatis sebelum dihapus secara otomatis. Jika Anda tidak ingin melakukan pencadangan otomatis terjadwal rutin, hapus kotak centang Aktifkan pencadangan otomatis. Apa pun pilihannya, Anda selalu bisa membuat pencadangan secara manual.

   Untuk informasi selengkapnya pencadangan dan pemulihan Redis, lihat Melakukan snapshot dan pemulihan.

8. (Opsional) Menentukan jendela pemeliharaan. Jendela pemeliharaan adalah waktu yang biasanya satu jam setiap minggu saat ElastiCache menjadwalkan pemeliharaan sistem untuk klaster Anda. Anda dapat mengizinkan ElastiCache untuk memilih hari dan waktu untuk jendela pemeliharaan Anda (Tidak ada preferensi), atau Anda dapat memilih hari, waktu, dan durasi sendiri (Tentukan jendela pemeliharaan). Jika Anda memilih Tentukan jendela pemeliharaan dari daftar, pilih Hari mulai, Waktu mulai, dan Durasi (dalam jam) untuk jendela pemeliharaan. Waktu yang digunakan adalah UCT.

   Untuk informasi selengkapnya, lihat Mengelola pemeliharaan.

9. (Opsional) Untuk Log:

   - Di bawah Format log, pilih salah satu dari Teks atau JSON.
   - Di bawah Jenis Tujuan, pilih CloudWatch Log atau Kinesis Firehose.
   - Di bawah Tujuan log, pilih Buat baru dan masukkan nama grup CloudWatch log Log atau nama aliran Firehose Anda, atau pilih Pilih yang ada, lalu pilih nama grup CloudWatch log Log atau nama aliran Firehose Anda,

10. Untuk Tag, untuk membantu mengelola cluster dan ElastiCache sumber daya lainnya, Anda dapat menetapkan metadata Anda sendiri ke setiap sumber daya dalam bentuk tag. Untuk informasi selengkapnya, lihat Menandai sumber daya ElastiCache Anda.
11. Pilih Selanjutnya.
12. Tinjau semua entri dan pilihan Anda, lalu lakukan koreksi yang diperlukan. Saat Anda siap, pilih Buat.

On premises

1. Untuk On-premis, sebaiknya Anda membiarkan Auto-failover tetap aktif. Untuk informasi selengkapnya, lihat Meminimalkan waktu henti ElastiCache untuk Redis dengan Multi-AZ
2. Ikuti langkah-langkahnya di Menggunakan Outposts.

Untuk membuat ekuivalen menggunakan ElastiCache API atau AWS CLI bukan ElastiCache konsol, lihat yang berikut ini:

- API: CreateReplicationGroup
- CLI: create-replication-group

Setelah status klaster Anda menjadi tersedia, Anda dapat memberikan akses ke EC2, menyambungkannya, dan mulai menggunakannya. Untuk informasi selengkapnya, lihat Langkah 3: Mengizinkan akses ke klaster dan Langkah 4: Menyambung ke simpul klaster.

> **Important**
>
> Setelah klaster Anda tersedia, Anda akan ditagih untuk setiap jam atau durasi saat klaster aktif, meskipun Anda tidak sedang aktif menggunakannya. Untuk menghentikan tagihan biaya untuk klaster ini, Anda harus menghapusnya. Lihat Menghapus klaster.

### Membuat klaster (AWS CLI)

Untuk membuat cluster menggunakan AWS CLI, gunakan `create-cache-cluster` perintah.

> **Important**
>
> Setelah klaster Anda tersedia, Anda akan ditagih untuk setiap jam atau durasi saat klaster aktif, meskipun Anda tidak sedang aktif menggunakannya. Untuk menghentikan tagihan biaya untuk klaster ini, Anda harus menghapusnya. Lihat Menghapus klaster.

### Membuat klaster (CLI) Redis (mode klaster dinonaktifkan)

#### Example – Klaster Redis (mode klaster dinonaktifkan) tanpa replika baca

Kode CLI berikut membuat klaster cache Redis (mode klaster dinonaktifkan) tanpa replika.

> **Note**
>
> Ketika membuat klaster menggunakan jenis simpul dari keluarga r6gd, Anda harus meneruskan parameter `data-tiering-enabled`.

Untuk Linux, macOS, atau Unix:

```
aws elasticache create-cache-cluster \
--cache-cluster-id my-cluster \
--cache-node-type cache.r4.large \
--engine redis \
--num-cache-nodes 1 \
--cache-parameter-group default.redis6.x \
--snapshot-arns arn:aws:s3:::my_bucket/snapshot.rdb
```

Untuk Windows:

```
aws elasticache create-cache-cluster ^
--cache-cluster-id my-cluster ^
--cache-node-type cache.r4.large ^
--engine redis ^
--num-cache-nodes 1 ^
--cache-parameter-group default.redis6.x ^
```

```
--snapshot-arns arn:aws:s3:::my_bucket/snapshot.rdb
```

Membuat klaster Redis (modus klaster diaktifkan) (AWS CLI)

Klaster Redis (mode klaster diaktifkan) (API/CLI: grup replikasi) tidak dapat dibuat menggunakan operasi `create-cache-cluster`. Untuk membuat klaster Redis (mode klaster diaktifkan) (API/CLI: grup replikasi), lihat Membuat grup replikasi di Redis (Mode Klaster Diaktifkan) dari awal (AWS CLI).

Untuk informasi selengkapnya, lihat topik ElastiCache referensi AWS CLI untuk `create-replication-group`.

Membuat cluster (ElastiCache API)

Untuk membuat cluster menggunakan ElastiCache API, gunakan `CreateCacheCluster` tindakan.

> **Important**
>
> Setelah klaster Anda tersedia, Anda akan ditagih untuk setiap jam atau durasi saat klaster aktif, meskipun Anda tidak sedang menggunakannya. Untuk menghentikan tagihan biaya untuk klaster ini, Anda harus menghapusnya. Lihat Menghapus klaster.

Topik

- Membuat cluster cache Redis (mode cluster dinonaktifkan) (ElastiCache API)
- Membuat cluster cache di Redis (mode cluster diaktifkan) (ElastiCache API)

Membuat cluster cache Redis (mode cluster dinonaktifkan) (ElastiCache API)

Kode berikut membuat Redis (mode cluster dinonaktifkan) cache cluster (ElastiCache API).

Jeda baris ditambahkan agar lebih mudah dibaca.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=CreateCacheCluster
    &CacheClusterId=my-cluster
    &CacheNodeType=cache.r4.large
    &CacheParameterGroup=default.redis3.2
    &Engine=redis
    &EngineVersion=3.2.4
    &NumCacheNodes=1
```

```
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &SnapshotArns.member.1=arn%3Aaws%3As3%3A%3A%3AmyS3Bucket%2Fdump.rdb
    &Timestamp=20150508T220302Z
    &Version=2015-02-02
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Credential=<credential>
    &X-Amz-Date=20150508T220302Z
    &X-Amz-Expires=20150508T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Signature=<signature>
```

Membuat cluster cache di Redis (mode cluster diaktifkan) (ElastiCache API)

Klaster Redis (mode klaster diaktifkan) (API/CLI: grup replikasi) tidak dapat dibuat menggunakan operasi `CreateCacheCluster`. Untuk membuat klaster Redis (mode klaster diaktifkan) (API/CLI: grup replikasi), lihat Membuat grup replikasi di Redis (Mode Klaster Diaktifkan) dari awal (API ElastiCache).

Untuk informasi selengkapnya, lihat topik referensi ElastiCache API `CreateReplicationGroup`.

# Melihat detail klaster

Anda dapat melihat informasi detail tentang satu atau lebih klaster menggunakan konsol ElastiCache, AWS CLI, atau API ElastiCache.

Melihat detail dari klaster Redis (Mode Klaster Dinonaktifkan) (Konsol)

Anda dapat melihat detail klaster Redis (mode klaster dinonaktifkan) menggunakan konsol ElastiCache, AWS CLI untuk ElastiCache, atau API ElastiCache.

Prosedur berikut memerinci cara untuk melihat detail klaster Redis (mode klaster dinonaktifkan) menggunakan konsol ElastiCache.

Melihat detail dari klaster Redis (mode klaster dinonaktifkan)

1. Masuk ke AWS Management Console dan buka konsol Amazon ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Pada dasbor konsol ElastiCache, pilih Redis untuk menampilkan daftar semua klaster yang menjalankan versi Redis apa pun.
3. Untuk melihat detail klaster, pilih kotak centang di sebelah kiri nama klaster. Pastikan bahwa Anda memilih klaster yang menjalankan mesin Redis, bukan Clustered Redis. Melakukan hal ini akan menampilkan detail tentang klaster, termasuk titik akhir primer dari klaster.
4. Untuk melihat informasi simpul:
    a. Pilih nama klaster.
    b. Pilih tab Serpihan dan Simpul. Melakukan hal ini akan menampilkan detail tentang setiap simpul, termasuk titik akhir simpul yang perlu digunakan untuk membaca dari klaster.
5. Untuk melihat metrik, pilih tab Metrik, yang menampilkan metrik yang relevan untuk semua simpul di klaster. Untuk informasi selengkapnya, lihat Memantau penggunaan dengan Metrik CloudWatch
6. Untuk melihat log, pilih tab Log, yang menunjukkan apakah klaster menggunakan log Lambat atau log Mesin dan memberikan detail yang relevan. Untuk informasi selengkapnya, lihat Pengiriman log.
7. Pilih tab Jaringan dan keamanan untuk melihat detail tentang konektivitas jaringan klaster dan konfigurasi grup subnet. Untuk informasi selengkapnya, lihat Subnet dan grup subnet.
8. Pilih tab Pemeliharaan untuk melihat detail di pengaturan pemeliharaan klaster. Untuk informasi selengkapnya, lihat Mengelola pemeliharaan.

9. Pilih tab Pembaruan layanan untuk melihat detail tentang pembaruan layanan yang tersedia beserta tanggal penerapan yang direkomendasikan. Untuk informasi selengkapnya, lihat Pembaruan layanan di ElastiCache.
10. Pilih tab Tanda untuk melihat detail pada tada apa pun yang diterapkan ke sumber daya klaster. Untuk informasi selengkapnya, lihat Menandai sumber daya ElastiCache Anda.

### Melihat detail untuk klaster Redis (Mode Klaster Diaktifkan) (Konsol)

Anda dapat melihat detail klaster Redis (mode klaster diaktifkan) menggunakan konsol ElastiCache, AWS CLI untuk ElastiCache, atau API ElastiCache.

Prosedur berikut memerinci cara untuk melihat detail klaster Redis (mode klaster diaktifkan) menggunakan konsol ElastiCache.

#### Melihat detail dari klaster Redis (mode klaster diaktifkan)

1. Masuk ke AWS Management Console dan buka konsol Amazon ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Dari daftar di sudut kanan atas, pilih Wilayah AWS yang Anda minati.
3. Pada dasbor konsol ElastiCache, pilih Redis untuk menampilkan daftar semua klaster yang menjalankan versi Redis apa pun.
4. Untuk melihat detail klaster Redis (mode klaster diaktifkan), memilih kotak di sebelah kiri nama klaster. Pastikan bahwa Anda memilih klaster yang menjalankan mesin Redis, bukan Clustered Redis.

   Layar akan melebar di bagian klaster dan menampilkan detail tentang klaster, termasuk titik akhir konfigurasi klaster.
5. Untuk melihat daftar serpihan klaster dan jumlah simpul di setiap serpihan, pilih tab Serpihan dan simpul.
6. Untuk melihat informasi spesifik pada sebuah simpul:

   - Pilih ID serpihan.

     Melakukan hal ini akan menampilkan informasi tentang setiap simpul, termasuk titik akhir simpul yang perlu digunakan untuk membaca dari klaster.

7. Untuk melihat metrik, pilih tab Metrik, yang menampilkan metrik yang relevan untuk semua simpul di klaster. Untuk informasi selengkapnya, lihat Memantau penggunaan dengan Metrik CloudWatch
8. Untuk melihat log, pilih tab Log, yang menunjukkan apakah klaster menggunakan log Lambat atau log Mesin dan memberikan detail yang relevan. Untuk informasi selengkapnya, lihat Pengiriman log.
9. Pilih tab Jaringan dan keamanan untuk melihat detail tentang konektivitas jaringan klaster dan konfigurasi grup subnet, grup keamanan VPC dan metode enkripsi hal yang diaktifkan di klaster, jika ada. Untuk informasi selengkapnya, silakan lihat Subnet dan grup subnet dan Keamanan data di Amazon ElastiCache.
10. Pilih tab Pemeliharaan untuk melihat detail di pengaturan pemeliharaan klaster. Untuk informasi selengkapnya, lihat Mengelola pemeliharaan.
11. Pilih tab Pembaruan layanan untuk melihat detail tentang pembaruan layanan yang tersedia beserta tanggal penerapan yang direkomendasikan. Untuk informasi selengkapnya, lihat Pembaruan layanan di ElastiCache.
12. Pilih tab Tanda untuk melihat detail pada tada apa pun yang diterapkan ke sumber daya klaster. Untuk informasi selengkapnya, lihat Menandai sumber daya ElastiCache Anda.

Melihat detail klaster (AWS CLI)

Kode berikut menampilkan detail untuk *my-cluster*:

```
aws elasticache describe-cache-clusters --cache-cluster-id my-cluster
```

Ganti *my-cluster* dengan nama klaster Anda dalam kasus saat klaster dibuat dengan 1 simpul cache dan 0 serpihan menggunakan perintah `create-cache-cluster`.

```
{
    "CacheClusters": [
        {
            "CacheClusterStatus": "available",
            "SecurityGroups": [
                {
                    "Status": "active",
                    "SecurityGroupId": "sg-dbe93fa2"
                }
            ],
```

```
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "Engine": "redis",
            "PreferredMaintenanceWindow": "wed:12:00-wed:13:00",
            "CacheSubnetGroupName": "default",
            "SnapshotWindow": "08:30-09:30",
            "TransitEncryptionEnabled": false,
            "AtRestEncryptionEnabled": false,
            "CacheClusterId": "my-cluster1",
            "CacheClusterCreateTime": "2018-02-26T21:06:43.420Z",
            "PreferredAvailabilityZone": "us-west-2c",
            "AuthTokenEnabled": false,
            "PendingModifiedValues": {},
            "CacheNodeType": "cache.r4.large",
           "DataTiering": "disabled",
            "CacheParameterGroup": {
                "CacheNodeIdsToReboot": [],
                "ParameterApplyStatus": "in-sync",
                "CacheParameterGroupName": "default.redis3.2"
            },
            "SnapshotRetentionLimit": 0,
            "AutoMinorVersionUpgrade": true,
            "EngineVersion": "3.2.10",
            "CacheSecurityGroups": [],
            "NumCacheNodes": 1
        }
```

```
{
    "CacheClusters": [
        {
            "SecurityGroups": [
                {
                    "Status": "active",
                    "SecurityGroupId": "sg-dbe93fa2"
                }
            ],
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "AuthTokenEnabled": false,
            "CacheSubnetGroupName": "default",
            "SnapshotWindow": "12:30-13:30",
            "AutoMinorVersionUpgrade": true,
            "CacheClusterCreateTime": "2018-02-26T21:13:24.250Z",
```

```
            "CacheClusterStatus": "available",
            "AtRestEncryptionEnabled": false,
            "PreferredAvailabilityZone": "us-west-2a",
            "TransitEncryptionEnabled": false,
            "ReplicationGroupId": "my-cluster2",
            "Engine": "redis",
            "PreferredMaintenanceWindow": "sun:08:30-sun:09:30",
            "CacheClusterId": "my-cluster2-001",
            "PendingModifiedValues": {},
            "CacheNodeType": "cache.r4.large",
            "DataTiering": "disabled",
            "CacheParameterGroup": {
                "CacheNodeIdsToReboot": [],
                "ParameterApplyStatus": "in-sync",
                "CacheParameterGroupName": "default.redis6.x"
            },
            "SnapshotRetentionLimit": 0,
            "EngineVersion": "6.0",
            "CacheSecurityGroups": [],
            "NumCacheNodes": 1
        },
        {
            "SecurityGroups": [
                {
                    "Status": "active",
                    "SecurityGroupId": "sg-dbe93fa2"
                }
            ],
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "AuthTokenEnabled": false,
            "CacheSubnetGroupName": "default",
            "SnapshotWindow": "12:30-13:30",
            "AutoMinorVersionUpgrade": true,
            "CacheClusterCreateTime": "2018-02-26T21:13:24.250Z",
            "CacheClusterStatus": "available",
            "AtRestEncryptionEnabled": false,
            "PreferredAvailabilityZone": "us-west-2b",
            "TransitEncryptionEnabled": false,
            "ReplicationGroupId": "my-cluster2",
            "Engine": "redis",
            "PreferredMaintenanceWindow": "sun:08:30-sun:09:30",
            "CacheClusterId": "my-cluster2-002",
            "PendingModifiedValues": {},
```

```
            "CacheNodeType": "cache.r4.large",
            "DataTiering": "disabled",
            "CacheParameterGroup": {
                "CacheNodeIdsToReboot": [],
                "ParameterApplyStatus": "in-sync",
                "CacheParameterGroupName": "default.redis6.x"
            },
            "SnapshotRetentionLimit": 0,
            "EngineVersion": "6.0",
            "CacheSecurityGroups": [],
            "NumCacheNodes": 1
        },
        {
            "SecurityGroups": [
                {
                    "Status": "active",
                    "SecurityGroupId": "sg-dbe93fa2"
                }
            ],
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "AuthTokenEnabled": false,
            "CacheSubnetGroupName": "default",
            "SnapshotWindow": "12:30-13:30",
            "AutoMinorVersionUpgrade": true,
            "CacheClusterCreateTime": "2018-02-26T21:13:24.250Z",
            "CacheClusterStatus": "available",
            "AtRestEncryptionEnabled": false,
            "PreferredAvailabilityZone": "us-west-2c",
            "TransitEncryptionEnabled": false,
            "ReplicationGroupId": "my-cluster2",
            "Engine": "redis",
            "PreferredMaintenanceWindow": "sun:08:30-sun:09:30",
            "CacheClusterId": "my-cluster2-003",
            "PendingModifiedValues": {},
            "CacheNodeType": "cache.r4.large",
            "DataTiering": "disabled",
            "CacheParameterGroup": {
                "CacheNodeIdsToReboot": [],
                "ParameterApplyStatus": "in-sync",
                "CacheParameterGroupName": "default.redis3.2"
            },
            "SnapshotRetentionLimit": 0,
            "EngineVersion": "3.2.10",
```

```
            "CacheSecurityGroups": [],
            "NumCacheNodes": 1
        }
```

```
{
    "CacheClusters": [
        {
            "SecurityGroups": [
                {
                    "Status": "active",
                    "SecurityGroupId": "sg-dbe93fa2"
                }
            ],
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "AuthTokenEnabled": true,
            "CacheSubnetGroupName": "default",
            "SnapshotWindow": "12:30-13:30",
            "AutoMinorVersionUpgrade": true,
            "CacheClusterCreateTime": "2018-02-26T21:17:01.439Z",
            "CacheClusterStatus": "available",
            "AtRestEncryptionEnabled": true,
            "PreferredAvailabilityZone": "us-west-2a",
            "TransitEncryptionEnabled": true,
            "ReplicationGroupId": "my-cluster3",
            "Engine": "redis",
            "PreferredMaintenanceWindow": "thu:11:00-thu:12:00",
            "CacheClusterId": "my-cluster3-0001-001",
            "PendingModifiedValues": {},
            "CacheNodeType": "cache.r4.large",
            "DataTiering": "disabled",
            "CacheParameterGroup": {
                "CacheNodeIdsToReboot": [],
                "ParameterApplyStatus": "in-sync",
                "CacheParameterGroupName": "default.redis6.x.cluster.on"
            },
            "SnapshotRetentionLimit": 0,
            "EngineVersion": "6.0",
            "CacheSecurityGroups": [],
            "NumCacheNodes": 1
        },
        {
            "SecurityGroups": [
```

```
                {
                    "Status": "active",
                    "SecurityGroupId": "sg-dbe93fa2"
                }
            ],
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "AuthTokenEnabled": true,
            "CacheSubnetGroupName": "default",
            "SnapshotWindow": "12:30-13:30",
            "AutoMinorVersionUpgrade": true,
            "CacheClusterCreateTime": "2018-02-26T21:17:01.439Z",
            "CacheClusterStatus": "available",
            "AtRestEncryptionEnabled": true,
            "PreferredAvailabilityZone": "us-west-2b",
            "TransitEncryptionEnabled": true,
            "ReplicationGroupId": "my-cluster3",
            "Engine": "redis",
            "PreferredMaintenanceWindow": "thu:11:00-thu:12:00",
            "CacheClusterId": "my-cluster3-0001-002",
            "PendingModifiedValues": {},
            "CacheNodeType": "cache.r4.large",
             "DataTiering": "disabled",
            "CacheParameterGroup": {
                "CacheNodeIdsToReboot": [],
                "ParameterApplyStatus": "in-sync",
                "CacheParameterGroupName": "default.redis3.2.cluster.on"
            },
            "SnapshotRetentionLimit": 0,
            "EngineVersion": "3.2.6",
            "CacheSecurityGroups": [],
            "NumCacheNodes": 1
        },
        {
            "SecurityGroups": [
                {
                    "Status": "active",
                    "SecurityGroupId": "sg-dbe93fa2"
                }
            ],
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "AuthTokenEnabled": true,
            "CacheSubnetGroupName": "default",
```

```
            "SnapshotWindow": "12:30-13:30",
            "AutoMinorVersionUpgrade": true,
            "CacheClusterCreateTime": "2018-02-26T21:17:01.439Z",
            "CacheClusterStatus": "available",
            "AtRestEncryptionEnabled": true,
            "PreferredAvailabilityZone": "us-west-2c",
            "TransitEncryptionEnabled": true,
            "ReplicationGroupId": "my-cluster3",
            "Engine": "redis",
            "PreferredMaintenanceWindow": "thu:11:00-thu:12:00",
            "CacheClusterId": "my-cluster3-0001-003",
            "PendingModifiedValues": {},
            "CacheNodeType": "cache.r4.large",
             "DataTiering": "disabled",
            "CacheParameterGroup": {
                "CacheNodeIdsToReboot": [],
                "ParameterApplyStatus": "in-sync",
                "CacheParameterGroupName": "default.redis6.x.cluster.on"
            },
            "SnapshotRetentionLimit": 0,
            "EngineVersion": "6.0",
            "CacheSecurityGroups": [],
            "NumCacheNodes": 1
        },
        {
            "SecurityGroups": [
                {
                    "Status": "active",
                    "SecurityGroupId": "sg-dbe93fa2"
                }
            ],
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "AuthTokenEnabled": true,
            "CacheSubnetGroupName": "default",
            "SnapshotWindow": "12:30-13:30",
            "AutoMinorVersionUpgrade": true,
            "CacheClusterCreateTime": "2018-02-26T21:17:01.439Z",
            "CacheClusterStatus": "available",
            "AtRestEncryptionEnabled": true,
            "PreferredAvailabilityZone": "us-west-2b",
            "TransitEncryptionEnabled": true,
            "ReplicationGroupId": "my-cluster3",
            "Engine": "redis",
```

```
            "PreferredMaintenanceWindow": "thu:11:00-thu:12:00",
            "CacheClusterId": "my-cluster3-0002-001",
            "PendingModifiedValues": {},
            "CacheNodeType": "cache.r4.large",
             "DataTiering": "disabled",
            "CacheParameterGroup": {
                "CacheNodeIdsToReboot": [],
                "ParameterApplyStatus": "in-sync",
                "CacheParameterGroupName": "default.redis6.x.cluster.on"
            },
            "SnapshotRetentionLimit": 0,
            "EngineVersion": "6.0",
            "CacheSecurityGroups": [],
            "NumCacheNodes": 1
        },
        {
            "SecurityGroups": [
                {
                    "Status": "active",
                    "SecurityGroupId": "sg-dbe93fa2"
                }
            ],
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "AuthTokenEnabled": true,
            "CacheSubnetGroupName": "default",
            "SnapshotWindow": "12:30-13:30",
            "AutoMinorVersionUpgrade": true,
            "CacheClusterCreateTime": "2018-02-26T21:17:01.439Z",
            "CacheClusterStatus": "available",
            "AtRestEncryptionEnabled": true,
            "PreferredAvailabilityZone": "us-west-2c",
            "TransitEncryptionEnabled": true,
            "ReplicationGroupId": "my-cluster3",
            "Engine": "redis",
            "PreferredMaintenanceWindow": "thu:11:00-thu:12:00",
            "CacheClusterId": "my-cluster3-0002-002",
            "PendingModifiedValues": {},
            "CacheNodeType": "cache.r4.large",
             "DataTiering": "disabled",
            "CacheParameterGroup": {
                "CacheNodeIdsToReboot": [],
                "ParameterApplyStatus": "in-sync",
                "CacheParameterGroupName": "default.redis3.2.cluster.on"
```

```
            },
            "SnapshotRetentionLimit": 0,
            "EngineVersion": "3.2.6",
            "CacheSecurityGroups": [],
            "NumCacheNodes": 1
        },
        {
            "SecurityGroups": [
                {
                    "Status": "active",
                    "SecurityGroupId": "sg-dbe93fa2"
                }
            ],
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "AuthTokenEnabled": true,
            "CacheSubnetGroupName": "default",
            "SnapshotWindow": "12:30-13:30",
            "AutoMinorVersionUpgrade": true,
            "CacheClusterCreateTime": "2018-02-26T21:17:01.439Z",
            "CacheClusterStatus": "available",
            "AtRestEncryptionEnabled": true,
            "PreferredAvailabilityZone": "us-west-2a",
            "TransitEncryptionEnabled": true,
            "ReplicationGroupId": "my-cluster3",
            "Engine": "redis",
            "PreferredMaintenanceWindow": "thu:11:00-thu:12:00",
            "CacheClusterId": "my-cluster3-0002-003",
            "PendingModifiedValues": {},
            "CacheNodeType": "cache.r4.large",
             "DataTiering": "disabled",
            "CacheParameterGroup": {
                "CacheNodeIdsToReboot": [],
                "ParameterApplyStatus": "in-sync",
                "CacheParameterGroupName": "default.redis6.x.cluster.on"
            },
            "SnapshotRetentionLimit": 0,
            "EngineVersion": "6.0",
            "CacheSecurityGroups": [],
            "NumCacheNodes": 1
        }
    ]
}
```

Dalam kasus saat klaster dibuat menggunakan AWS Management Console (simpul klaster diaktifkan atau dinonaktifkan dengan 1 atau beberapa serpihan), gunakan perintah berikut untuk menjelaskan detail klaster (ganti *my-cluster* dengan nama grup replikasi (nama klaster Anda):

```
aws elasticache describe-replication-groups --replication-group-id my-cluster
```

Untuk informasi selengkapnya, lihat topik AWS CLI untuk ElastiCache describe-cache-clusters.

Melihat detail klaster (API ElastiCache)

Anda dapat melihat detail untuk klaster menggunakan tindakan `DescribeCacheClusters` API ElastiCache. Jika parameter `CacheClusterId` disertakan, detail untuk klaster tertentu akan dikembalikan. Jika parameter `CacheClusterId` dihilangkan, detail hingga `MaxRecords` klaster (defaultnya 100) akan dikembalikan. Nilai untuk `MaxRecords` tidak boleh kurang dari 20 atau lebih besar dari 100.

Kode berikut menampilkan detail untuk `my-cluster`.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeCacheClusters
   &CacheClusterId=my-cluster
   &Version=2015-02-02
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

Kode berikut menampilkan detail hingga 25 klaster.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeCacheClusters
   &MaxRecords=25
   &Version=2015-02-02
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

Untuk informasi selengkapnya, lihat topik referensi API ElastiCache DescribeCacheClusters.

# Memodifikasi sebuah cluster ElastiCache

Selain menambahkan atau menghapus simpul dari klaster, ada waktu ketika Anda perlu membuat perubahan lain pada klaster yang ada, seperti, menambahkan grup keamanan, mengubah periode pemeliharaan atau grup parameter.

Sebaiknya atur periode pemeliharaan Anda pada waktu penggunaan terendah. Jadi, Anda mungkin perlu mengubahnya dari waktu ke waktu.

Saat Anda mengubah parameter klaster, perubahan ini diterapkan pada klaster secara langsung atau setelah klaster diaktifkan ulang. Hal ini berlaku terlepas dari apakah Anda mengubah grup parameter klaster itu sendiri atau nilai parameter di dalam grup parameter klaster. Untuk menentukan waktu penerapan perubahan parameter tertentu, lihat bagian Perubahan Berlaku pada kolom Detail di tabel untuk Parameter spesifik Redis.

Menggunakan AWS Management Console

Untuk mengubah klaster

1. Masuk ke AWS Management Console dan buka ElastiCache konsol di https://console.aws.amazon.com/elasticache/.
2. Dari daftar di sudut kanan atas, pilih AWS Wilayah tempat cluster yang ingin Anda modifikasi berada.
3. Di panel navigasi, pilih mesin yang berjalan pada klaster yang ingin diubah.

   Daftar klaster mesin yang dipilih akan muncul.
4. Dalam daftar klaster tersebut, pilih nama klaster yang ingin Anda ubah.
5. Pilih Tindakan, lalu pilih Ubah.

   Jendela Ubah Klaster akan muncul.
6. Di jendela Ubah Klaster, lakukan perubahan yang Anda inginkan. Opsinya meliputi:

   - Deskripsi
   - Mode cluster - Untuk mengubah mode klaster dari Dinonaktifkan ke Diaktifkan, Anda harus terlebih dahulu mengatur mode klaster ke Kompatibel.

     Mode yang kompatibel memungkinkan klien Redis Anda terhubung baik menggunakan mode klaster diaktifkan maupun mode klaster dinonaktifkan. Setelah Anda memigrasikan semua

klien Redis untuk menggunakan mode klaster diaktifkan, Anda kemudian dapat menyelesaikan konfigurasi mode klaster dan mengatur mode klaster ke Diaktifkan.

- Kompatibilitas Versi mesin

> **Important**
>
> Anda dapat meng-upgrade ke versi mesin yang lebih baru. Jika Anda meng-upgrade versi mayor mesin, misalnya dari 5.0.6 ke 6.0, Anda perlu memilih rangkaian grup parameter yang kompatibel dengan versi mesin yang baru. Untuk informasi selengkapnya tentang cara melakukannya, lihat Versi mesin dan pemutakhiran . Namun, Anda tidak dapat men-downgrade ke versi mesin yang lebih lama kecuali dengan menghapus klaster yang ada dan membuatnya lagi.

- Grup Keamanan VPC
- Grup Parameter
- Jenis Simpul

> **Note**
>
> Jika klaster menggunakan jenis simpul dari rangkaian r6gd, Anda hanya dapat memilih ukuran simpul yang berbeda dari dalam rangkaian tersebut. Jika Anda memilih jenis simpul dari rangkaian r6gd, tingkatan data akan diaktifkan secara otomatis. Untuk informasi selengkapnya, lihat Tingkatan data.

- Multi-AZ
- Failover otomatis (hanya mode klaster dinonaktifkan)
- Aktifkan Pencadangan Otomatis
- ID Simpul Cadangan
- Periode Retensi Cadangan
- Periode Pencadangan
- Topik untuk Notifikasi SNS

Kotak Terapkan Segera hanya berlaku untuk perubahan versi mesin. Untuk menerapkan perubahan secara langsung, pilih kotak centang Terapkan Segera. Jika kotak ini tidak dipilih,

perubahan jenis simpul dan versi mesin akan diterapkan pada periode pemeliharaan berikutnya. Perubahan lain, seperti perubahan periode pemeliharaan, akan diterapkan segera.

7. Pilih Ubah.

Untuk mengaktifkan/menonaktifkan pengiriman log

1. Dari daftar klaster, pilih klaster yang ingin diubah. Pilih Nama klaster, bukan kotak centang di sampingnya.
2. Di halaman Detail klaster, pilih tab Log,
3. Untuk mengaktifkan/menonaktifkan log lambat, pilih Aktifkan atau Nonaktifkan.

   Jika Anda memilih Aktifkan:

   a. Di bagian Format log, pilih Teks atau JSON.
   b. Di bawah Jenis tujuan Log, pilih CloudWatch Log atau Kinesis Firehose.
   c. Di bawah Tujuan log, pilih Buat baru dan masukkan nama grup CloudWatchLogs log Anda atau nama aliran Kinesis Data Firehose Anda. Atau pilih Pilih yang ada dan kemudian pilih nama grup CloudWatchLogs log Anda atau nama aliran Kinesis Data Firehose Anda.
   d. Pilih Aktifkan.

Untuk mengubah konfigurasi Anda:

1. Pilih Ubah.
2. Di bagian Format log, pilih Teks atau JSON.
3. Di bawah Jenis Tujuan, pilih CloudWatch Log atau Kinesis Firehose.
4. Di bawah Tujuan log, pilih Buat baru dan masukkan nama grup CloudWatchLogs log Anda atau nama aliran Kinesis Data Firehose Anda. Atau pilih Pilih yang ada lalu pilih nama grup CloudWatchLogs log Anda atau nama aliran Kinesis Data Firehose Anda.

Menggunakan AWS CLI

Anda dapat memodifikasi cluster yang ada menggunakan AWS CLI `modify-cache-cluster` operasi. Untuk mengubah nilai konfigurasi klaster, tentukan ID klaster, parameter yang akan diubah, dan nilai baru parameter. Contoh berikut mengubah periode pemeliharaan untuk klaster bernama `my-cluster` dan menerapkan perubahan tersebut secara langsung.

> **Important**
>
> Anda dapat meng-upgrade ke versi mesin yang lebih baru. Jika Anda meng-upgrade versi mayor mesin, misalnya dari 5.0.6 ke 6.0, Anda perlu memilih rangkaian grup parameter yang kompatibel dengan versi mesin yang baru. Untuk informasi selengkapnya tentang cara melakukannya, lihat Versi mesin dan pemutakhiran . Namun, Anda tidak dapat men-downgrade ke versi mesin yang lebih lama kecuali dengan menghapus klaster yang ada dan membuatnya lagi.

Untuk Linux, macOS, atau Unix:

```
aws elasticache modify-cache-cluster \
    --cache-cluster-id my-cluster \
    --preferred-maintenance-window sun:23:00-mon:02:00
```

Untuk Windows:

```
aws elasticache modify-cache-cluster ^
    --cache-cluster-id my-cluster ^
    --preferred-maintenance-window sun:23:00-mon:02:00
```

Parameter `--apply-immediately` hanya berlaku untuk perubahan pada jenis simpul, versi mesin, dan jumlah simpul di klaster. Jika Anda ingin menerapkan salah satu perubahan ini segera, gunakan parameter `--apply-immediately`. Jika Anda lebih memilih untuk menunda perubahan ini ke periode pemeliharaan berikutnya, gunakan parameter `--no-apply-immediately`. Perubahan lain, seperti perubahan periode pemeliharaan, akan diterapkan segera.

Untuk informasi selengkapnya, lihat ElastiCache topik AWS CLI untuk `modify-cache-cluster`.

Menggunakan ElastiCache API

Anda dapat memodifikasi klaster yang ada menggunakan `ModifyCacheCluster` operasi ElastiCache API. Untuk mengubah nilai konfigurasi klaster, tentukan ID klaster, parameter yang akan diubah, dan nilai baru parameter. Contoh berikut mengubah periode pemeliharaan untuk klaster bernama `my-cluster` dan menerapkan perubahan tersebut secara langsung.

> **Important**
>
> Anda dapat meng-upgrade ke versi mesin yang lebih baru. Jika Anda meng-upgrade versi mayor mesin, misalnya dari 5.0.6 ke 6.0, Anda perlu memilih rangkaian grup parameter yang kompatibel dengan versi mesin yang baru. Untuk informasi selengkapnya tentang cara melakukannya, lihat Versi mesin dan pemutakhiran . Namun, Anda tidak dapat men-downgrade ke versi mesin yang lebih lama kecuali dengan menghapus klaster yang ada dan membuatnya lagi.

Jeda baris ditambahkan agar lebih mudah dibaca.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=ModifyCacheCluster
    &CacheClusterId=my-cluster
    &PreferredMaintenanceWindow=sun:23:00-mon:02:00
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150901T220302Z
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20150202T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20150901T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

Parameter `ApplyImmediately` hanya berlaku untuk perubahan pada jenis simpul, versi mesin, dan jumlah simpul di klaster. Jika Anda ingin menerapkan salah satu perubahan ini secara langsung, tetapkan parameter `ApplyImmediately` ke `true`. Jika Anda lebih memilih untuk menunda perubahan ini ke periode pemeliharaan berikutnya, tetapkan parameter `ApplyImmediately` ke `false`. Perubahan lain, seperti perubahan periode pemeliharaan, akan diterapkan segera.

Untuk informasi selengkapnya, lihat topik referensi ElastiCache API `ModifyCacheCluster`.

# Menambahkan simpul ke klaster

Untuk mengonfigurasi ulang kalster Redis (mode klaster diaktifkan) Anda, lihat Penskalaan klaster di Redis (Mode Klaster Diaktifkan)

Anda dapat menggunakan Konsol Manajemen ElastiCache, AWS CLI, atau API ElastiCache untuk menambahkan simpul ke klaster Anda.

Menggunakan AWS Management Console

Jika Anda ingin menambahkan simpul ke klaster Redis simpul tunggal (mode klaster dinonaktifkan) (yang tidak mengaktifkan replikasi), prosesnya memerlukan dua langkah: pertama tambahkan replikasi, lalu tambahkan simpul replika.

Topik

- Untuk menambahkan replikasi ke klaster Redis tanpa serpihan
- Untuk menambahkan simpul ke klaster (konsol)

Prosedur berikut menambahkan replikasi ke Redis simpul tunggal yang tidak mengaktifkan replikasi. Saat Anda menambahkan replikasi, simpul yang ada menjadi simpul primer dalam klaster yang mengaktifkan replikasi. Setelah replikasi ditambahkan, Anda dapat menambahkan maksimum hingga 5 simpul replika ke klaster.

Untuk menambahkan replikasi ke klaster Redis tanpa serpihan

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Dari panel navigasi, pilih Klaster Redis.

   Daftar klaster yang menjalankan mesin Redis ditampilkan.
3. Pilih nama klaster, bukan kotak di sebelah kiri nama klaster, yang ingin ditambahkan simpul.

   Berikut ini adalah klaster Redis yang tidak mengaktifkan replikasi:

   - Klaster ini menjalankan Redis, bukan Clustered Redis.
   - Klaster ini tidak memiliki serpihan.

     Jika klaster memiliki serpihan, artinya replikasi sudah diaktifkan dan Anda dapat melanjutkan di Untuk menambahkan simpul ke klaster (konsol).

4. Pilih Tambahkan replikasi.
5. Di bagian Tambahkan replikasi, masukkan deskripsi untuk klaster yang mengaktifkan replikasi ini.
6. Pilih Tambahkan.

   Segera setelah status klaster kembali menjadi tersedia, Anda dapat melanjutkan ke prosedur berikutnya dan menambahkan replika ke klaster.

Untuk menambahkan simpul ke klaster (konsol)

Prosedur berikut dapat digunakan untuk menambahkan simpul ke klaster.

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Di panel navigasi, pilih mesin yang berjalan pada klaster yang ingin ditambahkan simpul.

   Daftar klaster yang menjalankan mesin yang dipilih akan muncul.
3. Dari daftar klaster tersebut, pilih nama klaster yang ingin ditambahkan simpul.

   Jika klaster Anda adalah klaster Redis (mode klaster diaktifkan), lihat Penskalaan klaster di Redis (Mode Klaster Diaktifkan).

   Jika klaster Anda adalah klaster Redis (mode klaster dinonaktifkan) tanpa serpihan, selesaikan langkah di Untuk menambahkan replikasi ke klaster Redis tanpa serpihan terlebih dahulu.
4. Pilih Tambahkan Simpul.
5. Lengkapi informasi yang diminta dalam kotak dialog Tambahkan Simpul.
6. Pilih tombol Terapkan Langsung - Ya untuk menambahkan simpul ini secara langsung, atau pilih Tidak untuk menambahkan simpul ini pada periode pemeliharaan berikutnya dari klaster.

   Dampak Permintaan Tambah dan Hapus Baru pada Permintaan Tertunda

| Skenario | Operasi Tertunda | Permintaan Baru | Hasil |
|---|---|---|---|
| Skenario 1 | Hapus | Hapus | Permintaan penghapusan baru, tertunda atau langsung, akan menggantikan permintaan penghapusan yang tertunda. |

| Skenario | Operasi Tertunda | Permintaan Baru | Hasil |
| --- | --- | --- | --- |
| | | | Misalnya, jika penghapus simpul 0001, 0003, dan 0007 tertunda dan permintaan baru untuk menghapus simpul 0002 dan 0004 dibuat, hanya simpul 0002 dan 0004 yang akan dihapus. Simpul 0001, 0003, dan 0007 tidak akan dihapus. |
| Skenario 2 | Hapus | Buat | Permintaan buat baru, tertunda atau langsung, akan menggantikan permintaan hapus tertunda.<br><br>Misalnya, jika pengahpusan simpul 0001, 0003, dan 0007 tertunda dan permintaan baru untuk membuat simpul dibuat, simpul baru akan dibuat dan simpul 0001, 0003, dan 0007 tidak akan dihapus. |
| Skenario 3 | Buat | Hapus | Permintaan hapus baru, tertunda atau langsung, akan menggantikan permintaan buat tertunda.<br><br>Misalnya, jika ada permintaan untuk membuat dua simpul tertunda dan permintaan baru dibuat untuk menghapus simpul 0003, tidak ada simpul baru yang akan dibuat dan simpul 0003 akan dihapus. |

| Skenario | Operasi Tertunda | Permintaan Baru | Hasil |
|---|---|---|---|
| Skenario 4 | Buat | Buat | Permintaan buat baru ditambahkan ke permintaan pembuatan yang tertunda.<br><br>Misalnya, jika ada permintaan tertunda untuk membuat dua simpul dan permintaan baru dibuat untuk membuat tiga simpul, permintaan baru ditambahkan ke permintaan tertunda dan lima simpul akan dibuat.<br><br>**Important**<br>Jika permintaan buat baru diatur ke Terapkan Langsung - Ya, semua permintaan buat akan dilakukan segera. Jika permintaan buat baru diatur ke Terapkan Langsung - Tidak, semua permintaan buat akan ditunda. |

Untuk menentukan operasi apa yang tertunda, pilih tab Deskripsi dan periksa untuk melihat berapa banyak pembuatan tertunda atau penghapusan ditampilkan. Anda tidak dapat memiliki kedua pembuatan tertunda dan penghapusan tertunda.

7. Pilih tombol Tambahkan.

   Setelah beberapa saat, simpul baru akan muncul dalam daftar simpul dengan status membuat. Jika itu tidak muncul, segarkan halaman peramban Anda. Saat status simpul berubah menjadi tersedia, simpul baru sudah dapat digunakan.

### Menggunakan AWS CLI

Jika Anda ingin menambahkan simpul ke klaster Redis (mode klaster dinonaktifkan) yang telah ada dan tidak mengaktifkan replikasi, Anda harus terlebih dahulu membuat grup replikasi yang menentukan klaster yang ada ini sebagai primer. Untuk informasi selengkapnya, lihat Membuat

Grup Replikasi Menggunakan Klaster Cache Redis yang Tersedia (AWS CLI). Setelah grup replikasi menjadi tersedia, Anda dapat melanjutkan proses berikut.

Untuk menambahkan simpul ke klaster menggunakan AWS CLI, gunakan operasi AWS CLI `increase-replica-count` dengan parameter berikut:

- `--replication-group-id` ID grup replikasi yang ingin ditambahkan simpul.
- `--new-replica-count` menentukan jumlah simpul yang diinginkan di dalam grup replikasi ini setelah perubahan diterapkan. Untuk menambahkan simpul ke klaster ini, `--new-replica-count` harus lebih besar dari jumlah simpul saat ini di dalam klaster.
- `--apply-immediately` atau `--no-apply-immediately` yang menentukan apakah akan menambahkan simpul ini secara langsung atau pada periode pemeliharaan berikutnya.

Untuk Linux, macOS, atau Unix:

```
aws elasticache increase-replica-count \
    --replication-group-id my-replication-group \
    --new-replica-count 4 \
    --apply-immediately
```

Untuk Windows:

```
aws elasticache increase-replica-count ^
    --replication-group-id my-replication-group ^
    --new-replica-count 4 ^
    --apply-immediately
```

Operasi ini menghasilkan output yang serupa dengan yang berikut (format JSON):

```
{
    "ReplicationGroup": {
        "ReplicationGroupId": "node-test",
        "Description": "node-test",
        "Status": "modifying",
        "PendingModifiedValues": {},
        "MemberClusters": [
            "node-test-001",
            "node-test-002",
            "node-test-003",
            "node-test-004",
```

```
            "node-test-005"
        ],
        "NodeGroups": [
            {
                "NodeGroupId": "0001",
                "Status": "modifying",
                "PrimaryEndpoint": {
                    "Address": "node-test.zzzzzz.ng.0001.usw2.cache.amazonaws.com",
                    "Port": 6379
                },
                "ReaderEndpoint": {
                    "Address": "node-test.zzzzzz.ng.0001.usw2.cache.amazonaws.com",
                    "Port": 6379
                },
                "NodeGroupMembers": [
                    {
                        "CacheClusterId": "node-test-001",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Address": "node-
test-001.zzzzzz.0001.usw2.cache.amazonaws.com",
                            "Port": 6379
                        },
                        "PreferredAvailabilityZone": "us-west-2a",
                        "CurrentRole": "primary"
                    },
                    {
                        "CacheClusterId": "node-test-002",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Address": "node-
test-002.zzzzzz.0001.usw2.cache.amazonaws.com",
                            "Port": 6379
                        },
                        "PreferredAvailabilityZone": "us-west-2c",
                        "CurrentRole": "replica"
                    },
                    {
                        "CacheClusterId": "node-test-003",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Address": "node-
test-003.zzzzzz.0001.usw2.cache.amazonaws.com",
                            "Port": 6379
```

```
                        },
                        "PreferredAvailabilityZone": "us-west-2b",
                        "CurrentRole": "replica"
                    }
                ]
            }
        ],
        "SnapshottingClusterId": "node-test-002",
        "AutomaticFailover": "enabled",
        "MultiAZ": "enabled",
        "SnapshotRetentionLimit": 1,
        "SnapshotWindow": "07:30-08:30",
        "ClusterEnabled": false,
        "CacheNodeType": "cache.r5.large",
         "DataTiering": "disabled",
        "TransitEncryptionEnabled": false,
        "AtRestEncryptionEnabled": false,
        "ARN": "arn:aws:elasticache:us-west-2:123456789012:replicationgroup:node-test"
    }
}
```

Untuk informasi selengkapnya, lihat AWS CLI topik `increase-replica-count`.

Menggunakan API ElastiCache

Jika Anda ingin menambahkan simpul ke klaster Redis (mode klaster dinonaktifkan) yang telah ada dan tidak mengaktifkan replikasi, Anda harus terlebih dahulu membuat grup replikasi yang menentukan klaster yang ada ini sebagai Primer. Untuk informasi selengkapnya, lihat Menambahkan replika ke klaster Redis (Mode Klaster Dinonaktifkan) mandiri (API ElastiCache). Setelah grup replikasi menjadi tersedia, Anda dapat melanjutkan proses berikut.

Untuk menambahkan simpul ke klaster (API ElastiCache)

- Panggil operasi `IncreaseReplicaCount` dengan parameter berikut ini:
  - `ReplicationGroupId` ID dari klaster yang ingin ditambahkan simpul.
  - `NewReplicaCount` Parameter `NewReplicaCount` menentukan jumlah simpul yang diinginkan dalam klaster ini setelah perubahan diterapkan. Untuk menambahkan simpul ke klaster ini, `NewReplicaCount` harus lebih besar dari jumlah simpul saat ini di dalam klaster. Jika nilai ini kurang dari jumlah simpul saat ini, gunakan API `DecreaseReplicaCount` dengan jumlah simpul yang akan dihapus dari klaster.

- `ApplyImmediately` Menentukan apakah akan menambahkan simpul ini secara langsung atau pada periode pemeliharaan berikutnya.
- `Region` Menentukan Wilayah AWS klaster yang ingin ditambahkan simpul.

Contoh berikut menunjukkan panggilan untuk menambahkan simpul ke klaster.

Example

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=IncreaseReplicaCount
    &ApplyImmediately=true
    &NumCacheNodes=4
    &ReplicationGroupId=my-replication-group
    &Region=us-east-2
    &Version=2014-12-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20141201T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

Untuk informasi selengkapnya, lihat topik API ElastiCache `IncreaseReplicaCount`.

# Menghapus simpul dari klaster

Anda dapat menghapus simpul dari klaster menggunakan AWS Management Console, AWS CLI, atau API ElastiCache.

Menggunakan AWS Management Console

Untuk menghapus simpul dari klaster (konsol)

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Dari daftar di sudut kanan atas, pilih Wilayah AWS sesuai lokasi klaster yang ingin dihapus simpulnya.
3. Di panel navigasi, pilih mesin yang berjalan pada klaster yang ingin dihapus simpulnya itu.

   Daftar klaster yang menjalankan mesin yang dipilih akan muncul.
4. Dari daftar klaster, pilih nama klaster yang ingin dihapus simpulnya.

   Daftar simpul klaster akan muncul.
5. Pilih kotak di sebelah kiri ID simpul untuk simpul yang ingin dihapus. Menggunakan konsol ElastiCache, Anda hanya dapat menghapus satu simpul pada satu waktu. Jadi, jika Anda memilih beberapa simpul, Anda tidak dapat menggunakan tombol Hapus simpul.

   Halaman Hapus Simpul muncul.
6. Untuk menghapus simpul, selesaikan halaman Hapus Simpul dan pilih Hapus Simpul. Untuk mempertahankan simpul, pilih Batalkan.

> **Important**
>
> Jika penghapusan simpul mengakibatkan klaster tidak lagi memenuhi syarat Multi-AZ, hapus kotak centang Multi-AZ terlebih dahulu, lalu hapus simpul tersebut. Jika Anda menghapus kotak centang Multi-AZ, Anda dapat memilih untuk mengaktifkan Failover otomatis.

Dampak Permintaan Tambah dan Hapus Baru pada Permintaan Tertunda

| Skenario | Operasi Tertunda | Permintaan Baru | Hasil |
|---|---|---|---|
| Skenario 1 | Hapus | Hapus | Permintaan penghapusan baru, tertunda atau langsung, akan menggantikan permintaan penghapus an yang tertunda.<br><br>Misalnya, jika penghapus simpul 0001, 0003, dan 0007 tertunda dan permintaan baru untuk menghapus simpul 0002 dan 0004 dibuat, hanya simpul 0002 dan 0004 yang akan dihapus. Simpul 0001, 0003, dan 0007 tidak akan dihapus. |
| Skenario 2 | Hapus | Buat | Permintaan buat baru, tertunda atau langsung, akan menggantikan permintaan hapus tertunda.<br><br>Misalnya, jika pengahpusan simpul 0001, 0003, dan 0007 tertunda dan permintaan baru untuk membuat simpul dibuat, simpul baru akan dibuat dan simpul 0001, 0003, dan 0007 tidak akan dihapus. |
| Skenario 3 | Buat | Hapus | Permintaan hapus baru, tertunda atau langsung, akan menggantikan permintaan buat tertunda.<br><br>Misalnya, jika ada permintaan untuk membuat dua simpul tertunda dan permintaan baru dibuat untuk menghapus simpul 0003, tidak ada simpul baru yang akan dibuat dan simpul 0003 akan dihapus. |
| Skenario 4 | Buat | Buat | Permintaan buat baru ditambahkan ke permintaan pembuatan yang tertunda.<br><br>Misalnya, jika ada permintaan tertunda untuk membuat dua simpul dan permintaan baru dibuat untuk membuat tiga simpul, permintaan baru ditambahkan ke permintaan tertunda dan lima simpul akan dibuat. |

| Skenario | Operasi Tertunda | Permintaan Baru | Hasil |
| --- | --- | --- | --- |
| | | | **Important**<br>Jika permintaan buat baru diatur ke Terapkan Langsung - Ya, semua permintaan buat akan dilakukan segera. Jika permintaan buat baru diatur ke Terapkan Langsung - Tidak, semua permintaan buat akan ditunda. |

Untuk menentukan operasi apa yang tertunda, pilih tab Deskripsi dan periksa untuk melihat berapa banyak pembuatan tertunda atau penghapusan ditampilkan. Anda tidak dapat memiliki kedua pembuatan tertunda dan penghapusan tertunda.

Menggunakan AWS CLI

1. Identifikasi ID simpul yang ingin dihapus. Untuk informasi selengkapnya, lihat Melihat detail klaster.
2. Gunakan operasi CLI `decrease-replica-count` dengan daftar simpul yang akan dihapus, seperti dalam contoh berikut.

   Untuk menghapus simpul dari klaster menggunakan antarmuka baris perintah, gunakan perintah `decrease-replica-count` dengan parameter berikut:

   - `--replication-group-id` ID grup replikasi yang ingin dihapus simpulnya.
   - `--new-replica-count` Parameter `--new-replica-count` menentukan jumlah simpul yang diinginkan dalam klaster ini setelah perubahan diterapkan.
   - `--replicas-to-remove` Daftar ID simpul yang ingin dihapus dari klaster ini.
   - `--apply-immediately` atau `--no-apply-immediately` Menentukan apakah akan menghapus simpul ini dengan segera atau pada periode pemeliharaan berikutnya.
   - `--region` Menentukan Wilayah AWS dari klaster yang ingin dihapus simpulnya.

> **Note**
>
> Anda bisa meneruskan hanya satu parameter `--replicas-to-remove` atau `--new-replica-count` saat memanggil operasi ini.

Untuk Linux, macOS, atau Unix:

```
aws elasticache decrease-replica-count \
    --replication-group-id my-replication-group \
    --new-replica-count 2 \
    --region us-east-2 \
    --apply-immediately
```

Untuk Windows:

```
aws elasticache decrease-replica-count ^
    --replication-group-id my-replication-group ^
    --new-replica-count 3 ^
    --region us-east-2 ^
    --apply-immediately
```

Operasi ini menghasilkan output yang serupa dengan yang berikut (format JSON):

```
{
    "ReplicationGroup": {
        "ReplicationGroupId": "node-test",
        "Description": "node-test"
       },
        "Status": "modifying",
        "PendingModifiedValues": {},
        "MemberClusters": [
            "node-test-001",
            "node-test-002",
            "node-test-003",
            "node-test-004",
            "node-test-005",
            "node-test-006"
        ],
```

```
        "NodeGroups": [
            {
                "NodeGroupId": "0001",
                "Status": "modifying",
                "PrimaryEndpoint": {
                    "Address": "node-test.zzzzzz.ng.0001.usw2.cache.amazonaws.com",
                    "Port": 6379
                },
                "ReaderEndpoint": {
                    "Address": "node-test-
ro.zzzzzz.ng.0001.usw2.cache.amazonaws.com",
                    "Port": 6379
                },
                "NodeGroupMembers": [
                    {
                        "CacheClusterId": "node-test-001",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Address": "node-
test-001.zzzzzz.0001.usw2.cache.amazonaws.com",
                            "Port": 6379
                        },
                        "PreferredAvailabilityZone": "us-west-2a",
                        "CurrentRole": "primary"
                    },
                    {
                        "CacheClusterId": "node-test-002",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Address": "node-
test-002.zzzzzz.0001.usw2.cache.amazonaws.com",
                            "Port": 6379
                        },
                        "PreferredAvailabilityZone": "us-west-2c",
                        "CurrentRole": "replica"
                    },
                    {
                        "CacheClusterId": "node-test-003",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Address": "node-
test-003.zzzzzz.0001.usw2.cache.amazonaws.com",
                            "Port": 6379
                        },
```

```
                        "PreferredAvailabilityZone": "us-west-2b",
                        "CurrentRole": "replica"
                    },
                    {
                        "CacheClusterId": "node-test-004",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Address": "node-
test-004.zzzzzz.0001.usw2.cache.amazonaws.com",
                            "Port": 6379
                        },
                        "PreferredAvailabilityZone": "us-west-2c",
                        "CurrentRole": "replica"
                    },
                    {
                        "CacheClusterId": "node-test-005",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Address": "node-
test-005.zzzzzz.0001.usw2.cache.amazonaws.com",
                            "Port": 6379
                        },
                        "PreferredAvailabilityZone": "us-west-2b",
                        "CurrentRole": "replica"
                    },
                    {
                        "CacheClusterId": "node-test-006",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Address": "node-
test-006.zzzzzz.0001.usw2.cache.amazonaws.com",
                            "Port": 6379
                        },
                        "PreferredAvailabilityZone": "us-west-2b",
                        "CurrentRole": "replica"
                    }
                ]
            }
        ],
        "SnapshottingClusterId": "node-test-002",
        "AutomaticFailover": "enabled",
        "MultiAZ": "enabled",
        "SnapshotRetentionLimit": 1,
        "SnapshotWindow": "07:30-08:30",
```

```
        "ClusterEnabled": false,
        "CacheNodeType": "cache.r5.large",
         "DataTiering": "disabled",
        "TransitEncryptionEnabled": false,
        "AtRestEncryptionEnabled": false,
        "ARN": "arn:aws:elasticache:us-west-2:123456789012:replicationgroup:node-
test"
    }
}
```

Cara lainnya, Anda dapat memanggil `decrease-replica-count` alih-alih meneruskan parameter `--new-replica-count`, Anda bisa meneruskan parameter `--replicas-to-remove`, seperti yang ditunjukkan berikut ini:

Untuk Linux, macOS, atau Unix:

```
aws elasticache decrease-replica-count \
    --replication-group-id my-replication-group \
    --replicas-to-remove node-test-003 \
    --region us-east-2 \
    --apply-immediately
```

Untuk Windows:

```
aws elasticache decrease-replica-count ^
    --replication-group-id my-replication-group ^
    --replicas-to-remove node-test-003 ^
    --region us-east-2 ^
    --apply-immediately
```

Untuk informasi selengkapnya, lihat topik AWS CLI `decrease-replica-count`.

Menggunakan API ElastiCache

Untuk menghapus simpul menggunakan API ElastiCache, panggil operasi API `DecreaseReplicaCount` dengan ID grup replikasi dan daftar simpul yang akan dihapus, seperti ditunjukkan berikut:

- `ReplicationGroupId` ID grup replikasi yang ingin dihapus simpulnya.
- `ReplicasToRemove` Parameter `ReplicasToRemove` menentukan jumlah simpul yang diinginkan dalam klaster ini setelah perubahan diterapkan.

- `ApplyImmediately` Menentukan apakah akan menghapus simpul ini dengan segera atau pada periode pemeliharaan berikutnya.
- `Region` Menentukan Wilayah AWS dari klaster yang ingin dihapus simpulnya.

Contoh berikut segera menghapus simpul 0004 dan 0005 dari klaster my-cluster.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DecreaseReplicaCount
    &ReplicationGroupId=my-replication-group
    &ApplyImmediately=true
    &ReplicasToRemove=node-test-003
    &Region us-east-2
    &Version=2014-12-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20141201T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

Untuk informasi selengkapnya, lihat topik API ElastiCache `DecreaseReplicaCount`.

# Membatalkan operasi penambahan atau penghapusan simpul yang tertunda

Jika Anda memilih untuk tidak langsung menerapkan perubahan, operasi memiliki status tertunda hingga dilakukan pada periode pemeliharaan berikutnya. Anda dapat membatalkan setiap operasi tertunda.

Untuk membatalkan operasi yang tertunda

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Dari daftar di sudut kanan atas, pilih Wilayah AWS yang ingin Anda membatalkan operasi penambahan atau penghapusan simpul yang tertunda.
3. Di panel navigasi, pilih mesin yang berjalan di klaster yang memiliki operasi tertunda yang ingin dibatalkan. Daftar klaster yang menjalankan mesin yang dipilih akan muncul.
4. Dalam daftar klaster, pilih nama klaster, bukan kotak di sebelah kiri nama klaster, yang memiliki operasi tertunda yang ingin Anda batalkan.
5. Untuk menentukan operasi apa yang tertunda, pilih tab Deskripsi dan periksa untuk melihat berapa banyak pembuatan tertunda atau penghapusan ditampilkan. Anda tidak dapat memiliki kedua pembuatan tertunda dan penghapusan tertunda.
6. Pilih tab Simpul.
7. Untuk membatalkan semua operasi yang tertunda, klik Batalkan Penundaan. Kotak dialog Batalkan Penundaan akan muncul.
8. Konfirmasi bahwa Anda ingin membatalkan semua operasi yang tertunda dengan memilih tombol Batalkan Penundaan, atau untuk mempertahankan operasi, pilih Batalkan.

# Menghapus klaster

Selama klaster dalam status tersedia, Anda akan dikenakan biaya, terlepas dari apakah Anda secara aktif menggunakannya atau tidak. Untuk berhenti dikenakan biaya, hapus klaster tersebut.

> **Warning**
>
> Saat Anda menghapus klaster ElastiCache for Redis, snapshot manual Anda akan disimpan. Anda juga dapat membuat snapshot akhir sebelum klaster dihapus. Snapshot cache otomatis tidak dipertahankan.

## Menggunakan AWS Management Console

Prosedur berikut menghapus satu klaster dari deployment Anda. Untuk menghapus beberapa klaster, ulangi prosedur untuk setiap klaster yang ingin dihapus. Anda tidak perlu menunggu satu klaster selesai dihapus sebelum memulai prosedur untuk menghapus klaster lain.

Untuk menghapus klaster

1. Masuk ke AWS Management Console dan buka konsol Amazon ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Pada dasbor konsol ElastiCache, pilih mesin yang dijalankan oleh klaster yang ingin dihapus.

   Daftar semua klaster yang menjalankan mesin tersebut akan muncul.
3. Untuk memilih klaster yang akan dihapus, pilih nama klaster tersebut dari daftar klaster.

   > **Important**
   >
   > Anda hanya dapat menghapus satu klaster saja pada satu waktu dari konsol ElastiCache. Memilih beberapa klaster akan menonaktifkan operasi hapus.
4. Untuk Tindakan, pilih Hapus.
5. Pada layar konfirmasi Hapus Klaster, pilih Hapus untuk menghapus klaster, atau Batal untuk mempertahankan klaster.

   Jika Anda memilih Hapus, status klaster berubah menjadi menghapus.

Segera setelah klaster Anda tidak lagi tercantum di dalam daftar klaster, Anda berhenti dikenakan biaya untuk itu.

Menggunakan AWS CLI

Kode berikut menghapus klaster cache `my-cluster`.

```
aws elasticache delete-cache-cluster --cache-cluster-id my-cluster
```

Tindakan CLI `delete-cache-cluster` hanya menghapus satu klaster cache. Untuk menghapus beberapa klaster cache, panggil `delete-cache-cluster` untuk setiap klaster cache yang ingin dihapus. Anda tidak perlu menunggu satu klaster cache selesai dihapus untuk dapat menghapus yang lain.

Untuk Linux, macOS, atau Unix:

```
aws elasticache delete-cache-cluster \
    --cache-cluster-id my-cluster \
    --region us-east-2
```

Untuk Windows:

```
aws elasticache delete-cache-cluster ^
    --cache-cluster-id my-cluster ^
    --region us-east-2
```

Untuk informasi selengkapnya, lihat topik AWS CLI untuk ElastiCache `delete-cache-cluster`.

Menggunakan API ElastiCache

Kode berikut menghapus klaster `my-cluster`.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DeleteCacheCluster
    &CacheClusterId=my-cluster
    &Region us-east-2
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T220302Z
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20150202T220302Z
```

```
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20150202T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

Operasi API `DeleteCacheCluster` hanya menghapus satu klaster cache. Untuk menghapus beberapa klaster cache, panggil `DeleteCacheCluster` untuk setiap klaster cache yang ingin dihapus. Anda tidak perlu menunggu satu klaster cache selesai dihapus untuk dapat menghapus yang lain.

Untuk informasi lebih lanjut, lihat topik referensi API ElastiCache `DeleteCacheCluster`.

# Mengakses klaster atau grup replikasi Anda

Instans Amazon ElastiCache Anda dirancang untuk diakses melalui instans Amazon EC2.

Jika Anda meluncurkan instans ElastiCache Anda di Amazon Virtual Private Cloud (Amazon VPC), Anda dapat mengakses instans ElastiCache Anda dari instans Amazon EC2 di Amazon VPC yang sama. Anda juga dapat menggunakan peering VPC untuk mengakses instans ElastiCache Anda dari Amazon EC2 di Amazon VPC yang berbeda.

Jika Anda meluncurkan instans ElastiCache Anda di EC2 Classic, Anda mengizinkan instans EC2 untuk mengakses klaster Anda dengan memberikan akses ke grup keamanan cache Anda kepada grup keamanan Amazon EC2 yang terkait dengan instans tersebut. Secara default, akses ke klaster dibatasi pada akun yang meluncurkan klaster tersebut.

Topik



## Memberikan akses ke klaster atau grup replikasi Anda

Anda meluncurkan klaster Anda ke EC2-VPC

Jika Anda meluncurkan klaster ke Amazon Virtual Private Cloud (Amazon VPC), Anda dapat terhubung ke klaster ElastiCache Anda hanya dari instans Amazon EC2 yang berjalan di Amazon VPC yang sama. Dalam hal ini, Anda akan perlu memberikan izin masuk jaringan kepada klaster.

> **Note**
>
> Pastikan Anda telah mengaktifkan Local Zones jika Anda menggunakannya. Untuk informasi lain, lihat Mengaktifkan Local Zones. Dengan mengaktifkannya, VPC Anda diperluas ke Zona Lokal dan VPC Anda akan memperlakukan subnet seperti subnet apa pun di Availability Zone yang lain. Gateway, tabel rute dan pertimbangan grup keamanan lainnya yang berkaitan akan menyesuaikan secara otomatis.

Untuk memberikan izin masuk jaringan dari grup keamanan Amazon VPC ke klaster

1. Masuk ke AWS Management Console dan buka konsol Amazon EC2 di https://console.aws.amazon.com/ec2/.

2. Pada panel navigasi, di bagian Jaringan & Keamanan, pilih Grup Keamanan.
3. Dari daftar grup keamanan, pilih grup keamanan untuk Amazon VPC Anda. Kecuali Anda membuat grup keamanan untuk penggunaan ElastiCache, grup keamanan ini akan diberi nama default.
4. Pilih tab Masuk, lalu lakukan hal berikut:

    a. Pilih Edit.

    b. Pilih Tambahkan aturan.

    c. Di kolom Jenis, pilih Aturan TCP khusus.

    d. Di kotak Rentang port, ketik nomor port untuk simpul klaster Anda. Nomor ini harus sama dengan yang Anda tentukan saat meluncurkan klaster. Port default untuk Redis adalah **6379**.

    e. Di kotak Sumber, pilih Di mana saja yang memiliki rentang port (0.0.0.0/0). Dengan memilih opsi ini, setiap instans Amazon EC2 yang Anda luncurkan di Amazon VPC Anda dapat terhubung ke simpul ElastiCache Anda.

    > **Important**
    >
    > Membuka klaster ElastiCache untuk 0.0.0.0/0 tidak mengekspos klaster ke Internet karena klaster tidak memiliki alamat IP publik sehingga klaster tidak dapat diakses dari luar VPC. Namun, grup keamanan default dapat diterapkan ke instans Amazon EC2 lainnya di akun pelanggan, dan instans tersebut mungkin memiliki alamat IP publik. Jika instans tersebut menjalankan sesuatu di port default, layanan tersebut dapat terekspos secara tak disengaja. Oleh karena itu, sebaiknya buat Grup Keamanan VPC yang akan digunakan secara eksklusif oleh ElastiCache. Untuk informasi selengkapnya, lihat Grup Keamanan Khusus.

    f. Pilih Simpan.

Saat Anda meluncurkan instans Amazon EC2 ke Amazon VPC Anda, instans tersebut dapat terhubung ke klaster ElastiCache Anda.

## Mengakses sumber daya ElastiCache dari luar AWS

Amazon ElastiCache adalah layanan AWS yang menyediakan penyimpanan nilai-kunci dalam memori berbasis cloud. Layanan ini dirancang untuk diakses secara eksklusif dari dalam AWS. Namun, jika klaster ElastiCache di-host di dalam VPC, Anda dapat menggunakan instans Network Address Translation (NAT) untuk menyediakan akses luar.

### Persyaratan

Persyaratan berikut harus dipenuhi agar Anda dapat mengakses sumber daya ElastiCache Anda dari luar AWS:

- Klaster harus berada dalam VPC dan diakses melalui instans Network Address Translation (NAT). Tidak ada pengecualian untuk persyaratan ini.
- Instans NAT harus diluncurkan di VPC yang sama dengan klaster.
- Instans NAT harus diluncurkan di subnet publik yang terpisah dari klaster.
- Alamat IP Elastis (EIP) harus dikaitkan dengan instans NAT. Fitur penerusan port iptables digunakan untuk meneruskan port di instans NAT ke port simpul cache di dalam VPC.

### Pertimbangan

Perhatikan pertimbangan berikut saat mengakses sumber daya ElastiCache Anda dari luar ElastiCache.

- Klien terhubung ke port EIP dan cache dari instans NAT. Penerusan port di instans NAT meneruskan lalu lintas ke simpul klaster cache yang sesuai.
- Jika simpul klaster ditambahkan atau diganti, aturan iptables perlu diperbarui untuk mencerminkan perubahan ini.

### Batasan

Pendekatan ini harus digunakan untuk tujuan pengujian dan pengembangan saja. Sebaiknya jangan digunakan untuk produksi karena batasan berikut:

- Instans NAT bertindak sebagai proxy antara klien dan beberapa klaster. Penambahan proxy berdampak pada performa klaster cache. Dampaknya meningkat dengan jumlah klaster cache yang Anda akses melalui instans NAT.

- Lalu lintas dari klien ke instans NAT tidak terenkripsi. Oleh karena itu, Anda harus menghindari pengiriman data sensitif melalui instans NAT.
- Instans NAT menambahkan overhead untuk mempertahankan instans lain.
- Instans NAT berfungsi sebagai satu titik kegagalan. Untuk informasi tentang cara mengatur NAT ketersediaan tinggi di VPC, lihat Ketersediaan Tinggi untuk Instans NAT Amazon VPC: Contoh.

Cara mengakses sumber daya ElastiCache dari luar AWS

Prosedur berikut menunjukkan cara terhubung ke sumber daya ElastiCache Anda menggunakan instans NAT.

Langkah-langkah ini mengasumsikan sebagai berikut:

- `iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 6380 -j DNAT --to 10.0.1.231:6379`
- `iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 6381 -j DNAT --to 10.0.1.232:6379`

Selanjutnya Anda membutuhkan NAT ke arah yang berlawanan:

```
iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to-source 10.0.0.55
```

Anda juga perlu mengaktifkan penerusan IP, yang dinonaktifkan secara default:

```
sudo sed -i 's/net.ipv4.ip_forward=0/net.ipv4.ip_forward=1/g' /etc/
sysctl.conf sudo sysctl --system
```

- Anda mengakses klaster Redis dengan:
  - Alamat IP – 10.0.1.230
  - Port Redis default – 6379
  - Grup keamanan — sg-bd56b7da
  - Alamat IP instans AWS – sg-bd56b7da
- Klien tepercaya Anda memiliki alamat IP 198.51.100.27.
- Instans NAT Anda memiliki Alamat IP Elastis 203.0.113.73.
- Instans NAT Anda memiliki grup keamanan sg-ce56b7a9.

Untuk terhubung ke resource ElastiCache Anda menggunakan instans NAT

1. Buat instans NAT di VPC yang sama dengan klaster cache Anda tetapi di subnet publik.

   Secara default, wizard VPC akan meluncurkan tipe simpul cache.m1.small. Anda harus memilih ukuran simpul berdasarkan kebutuhan Anda. Anda harus menggunakan EC2 NAT AMI agar dapat mengakses ElastiCache dari luar AWS.

   Untuk informasi tentang cara membuat instans NAT, lihat Instans NAT di Panduan Pengguna VPC AWS.

2. Buat aturan grup keamanan untuk klaster cache dan instans NAT.

   Grup keamanan instans NAT dan instans cluster harus memiliki aturan berikut:

   - Dua aturan masuk
     - Satu untuk mengizinkan koneksi TCP dari klien tepercaya ke setiap port cache yang diteruskan dari instans NAT (6379 - 6381).
     - Sedetik untuk mengizinkan akses SSH ke klien tepercaya.

     Grup keamanan instans NAT - aturan masuk

     | Tipe | Protokol | Rentang port | Sumber |
     |---|---|---|---|
     | Aturan TCP Kustom | TCP | 6379-6380 | 198.51.100.27/32 |
     | SSH | TCP | %22 | 203.0.113.73/32 |

   - Aturan keluar untuk mengizinkan koneksi TCP ke port cache (6379).

     Grup keamanan instans NAT - aturan keluar

     | Tipe | Protokol | Rentang port | Tujuan |
     |---|---|---|---|
     | Aturan TCP Kustom | TCP | 6379 | sg-ce56b7a9 (Grup Keamanan instans klaster) |

   - Aturan masuk untuk grup keamanan klaster yang mengizinkan koneksi TCP dari instans NAT ke port cache (6379).

Grup keamanan instans klaster - aturan masuk

| Tipe | Protokol | Rentang port | Sumber |
| --- | --- | --- | --- |
| Aturan TCP Kustom | TCP | 6379 | sg-bd56b7da (Grup Keamanan Klaster) |

3. Validasi aturan.

   - Konfirmasikan bahwa klien tepercaya dapat melakukan SSH ke instans NAT.
   - Konfirmasikan bahwa klien tepercaya dapat terhubung ke klaster dari instans NAT.

4. Tambahkan aturan iptables ke instans NAT.

   Aturan iptables harus ditambahkan ke tabel NAT untuk setiap simpul di klaster untuk meneruskan port cache dari instans NAT ke simpul klaster. Contohnya mungkin terlihat seperti berikut:

   ```
   iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 6379 -j DNAT --to
    10.0.1.230:6379
   ```

   Nomor port harus unik untuk setiap simpul di klaster. Misalnya, jika bekerja dengan klaster Redis tiga simpul menggunakan port 6379 - 6381, aturan akan terlihat seperti berikut:

   ```
   iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 6379 -j DNAT --to
    10.0.1.230:6379
   iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 6380 -j DNAT --to
    10.0.1.231:6379
   iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 6381 -j DNAT --to
    10.0.1.232:6379
   ```

5. Konfirmasikan bahwa klien tepercaya dapat terhubung ke klaster.

   Klien tepercaya harus terhubung ke EIP yang terkait dengan instans NAT dan port klaster yang sesuai dengan simpul klaster yang sesuai. Misalnya, string koneksi untuk PHP mungkin terlihat seperti berikut:

   ```
   redis->connect( '203.0.113.73', 6379 );
   redis->connect( '203.0.113.73', 6380 );
   redis->connect( '203.0.113.73', 6381 );
   ```

Klien telnet juga dapat digunakan untuk memverifikasi koneksi. Misalnya:

```
telnet 203.0.113.73 6379
telnet 203.0.113.73 6380
telnet 203.0.113.73 6381
```

6. Simpan konfigurasi iptables.

   Simpan aturan setelah Anda menguji dan memverifikasinya. Jika Anda menggunakan distribusi Linux berbasis Redhat (seperti Amazon Linux), jalankan perintah berikut:

   ```
   service iptables save
   ```

Topik terkait

Topik-topik berikut mungkin menarik bagi Anda.

- Pola Akses untuk Mengakses Cache ElastiCache di Amazon VPC
- Mengakses Cache ElastiCache dari Aplikasi yang Berjalan di Pusat Data Pelanggan
- Instans NAT
- Mengonfigurasi Klien ElastiCache
- Ketersediaan Tinggi untuk Instans NAT Amazon VPC: Contoh

## Menemukan titik akhir koneksi

Aplikasi Anda terhubung ke klaster Anda menggunakan titik akhir. Titik akhir adalah alamat unik dari simpul atau klaster.

Jika tidak menggunakan Penemuan Otomatis, Anda harus mengonfigurasi klien Anda untuk menggunakan titik akhir simpul individual untuk proses baca dan tulis. Anda juga harus melacak titik akhir saat menambahkan dan menghapus simpul.

Titik akhir mana yang digunakan

- Simpul mandiri Redis, gunakan titik akhir simpul untuk operasi baca dan tulis.

- Klaster Redis (mode klaster dinonaktifkan), gunakan Titik Akhir Primer untuk semua operasi tulis. Gunakan Titik Akhir Pembaca untuk membagi koneksi masuk ke titik akhir secara merata di antara semua replika baca. GunakanTitik Akhir Simpul individual untuk operasi baca (Dalam API/CLI, ini disebut sebagai Titik Akhir Baca).

- Klaster Redis (mode klaster diaktifkan), gunakan Titik Akhir Konfigurasi klaster untuk semua operasi yang mendukung perintah mode klaster diaktifkan. Anda harus menggunakan klien yang mendukung Klaster Redis (Redis 3.2). Anda masih dapat membaca dari titik akhir simpul individual (Dalam API/CLI, ini disebut sebagai Titik Akhir Baca).

Bagian berikut memandu Anda menemukan titik akhir yang Anda perlukan untuk mesin yang sedang Anda jalankan.

### Menemukan Titik Akhir Klaster Redis (Mode Klaster Dinonaktifkan) (Konsol)

Jika klaster Redis (mode klaster dinonaktifkan) hanya memiliki satu simpul, maka titik akhir simpul tersebut digunakan untuk proses baca dan tulis. Jika klaster Redis (mode klaster dinonaktifkan) memiliki beberapa simpul, ada tiga jenis titik akhir: titik akhir primer, titik akhir pembaca, dan titik akhir simpul.

Titik akhir primer adalah nama DNS yang selalu diresolusi ke simpul primer di klaster. Titik akhir primer tidak terpengaruh oleh perubahan klaster Anda, seperti promosi replika baca ke peran primer. Untuk aktivitas tulis, sebaiknya aplikasi Anda terhubung ke titik akhir primer.

Titik akhir pembaca akan membagi koneksi masuk ke titik akhir secara merata di antara semua replika baca di klaster ElastiCache for Redis. Faktor lain seperti saat aplikasi membuat koneksi atau cara aplikasi menggunakan atau menggunakan (ulang) koneksi akan menentukan distribusi lalu lintas. Titik akhir pembaca tetap mengikuti perubahan klaster dalam waktu nyata saat replika ditambahkan atau dihapus. Anda dapat menempatkan beberapa replika baca dari klaster ElastiCache for Redis pada Zona Ketersediaan (AZ) AWS yang berbeda untuk memastikan ketersediaan tinggi titik akhir pembaca.

> **Note**
>
> Titik akhir pembaca bukan penyeimbang beban. Ini adalah catatan DNS yang akan diresolusi sebagai alamat IP dari salah satu simpul replika dengan metode round robin.

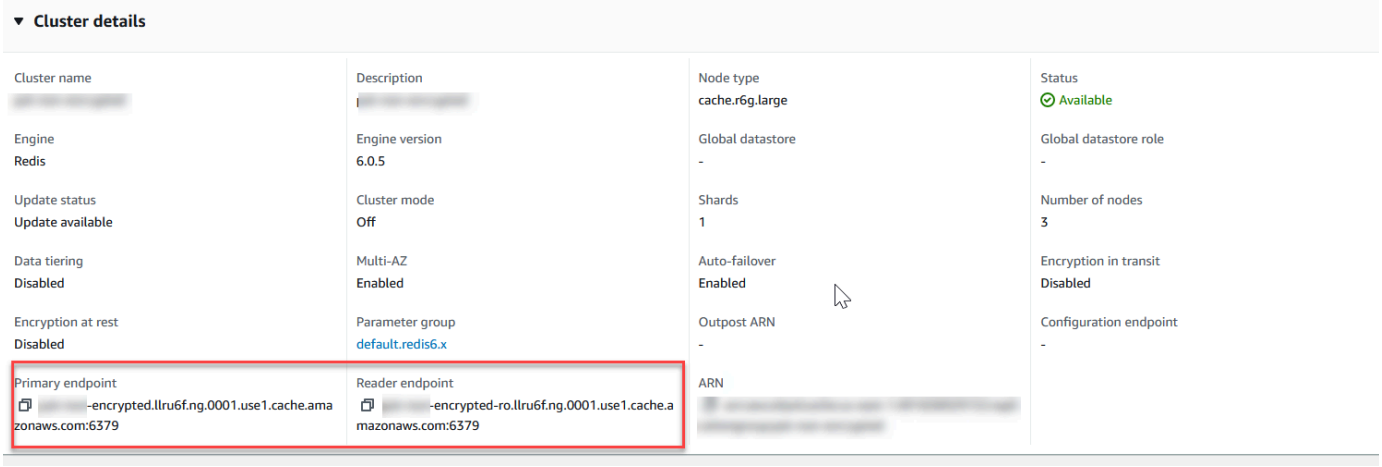
Untuk aktivitas baca, aplikasi juga dapat menghubungkan ke simpul mana pun di klaster. Tidak seperti titik akhir primer, titik akhir simpul diresolusi ke titik akhir tertentu. Jika Anda membuat perubahan di dalam klaster Anda, seperti menambahkan atau menghapus replika, Anda harus memperbarui titik akhir simpul di aplikasi Anda.

Menemukan titik akhir klaster Redis (mode klaster dinonaktifkan)

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Dari panel navigasi, pilih Klaster Redis.

   Layar klaster akan muncul dengan daftar klaster Redis (mode klaster dinonaktifkan) dan Redis (mode klaster diaktifkan).

3. Untuk menemukan titik akhir Primer dan/atau Pembaca klaster, pilih nama klaster (bukan tombol di sebelah kirinya).

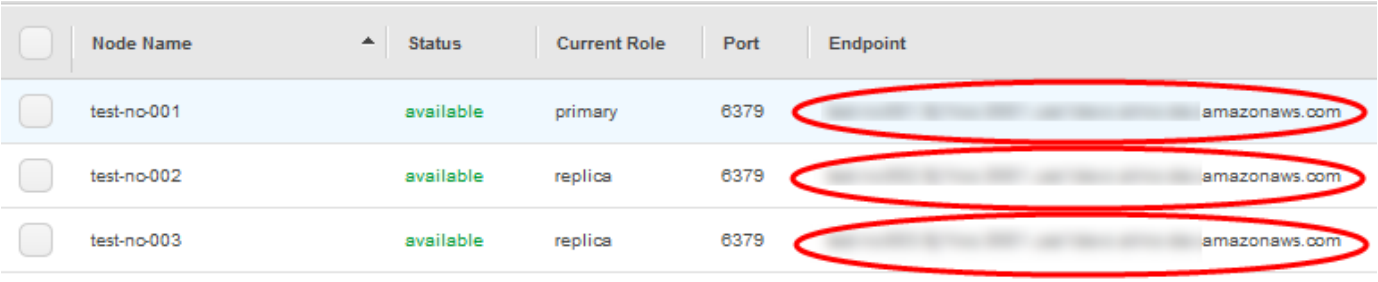



Titik akhir Primer dan Pembaca untuk klaster Redis (mode klaster dinonaktifkan)

Jika hanya ada satu simpul dalam klaster, berarti tidak ada titik akhir primer dan Anda dapat melanjutkan ke langkah berikutnya.

4. Jika klaster Redis (mode klaster dinonaktifkan) memiliki simpul replika, Anda dapat menemukan titik akhir simpul replika klaster dengan memilih nama klaster lalu memilih tab Simpul.

   Layar simpul muncul dengan setiap simpul yang ada di klaster, primer dan replika, yang tercantum dengan titik akhirnya.




   Titik akhir simpul untuk klaster Redis (mode klaster dinonaktifkan)

5. Untuk menyalin titik akhir ke clipboard Anda:

   a. Temukan satu per satu titik akhir yang ingin Anda salin.

   b. Pilih ikon salin langsung di depan titik akhir.

   Titik akhir sekarang disalin ke clipboard Anda. Untuk informasi tentang menggunakan titik akhir agar terhubung ke simpul, lihat Menghubungkan ke simpul.

Titik akhir primer Redis (mode klaster dinonaktifkan) terlihat seperti berikut ini. Ada perbedaan yang tergantung pada apakah enkripsi Bergerak aktif atau tidak.

Enkripsi bergerak tidak diaktifkan

```
clusterName.xxxxxx.nodeId.regionAndAz.cache.amazonaws.com:port

redis-01.7abc2d.0001.usw2.cache.amazonaws.com:6379
```

Enkripsi bergerak diaktifkan

```
master.clusterName.xxxxxx.regionAndAz.cache.amazonaws.com:port

master.ncit.ameaqx.use1.cache.amazonaws.com:6379
```

Menemukan Titik Akhir Klaster Redis (Mode Klaster Dinonaktifkan) (Konsol)

Sebuah klaster Redis (mode klaster diaktifkan) memiliki satu titik akhir konfigurasi. Dengan terhubung ke titik akhir konfigurasi, aplikasi Anda mampu menemukan titik akhir primer dan baca untuk setiap serpihan di klaster.

Untuk menemukan titik akhir klaster Redis (mode klaster dinonaktifkan)

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Dari panel navigasi, pilih Klaster Redis.

   Layar klaster akan muncul dengan daftar klaster Redis (mode klaster dinonaktifkan) dan Redis (mode klaster diaktifkan). Pilih klaster Redis (mode klaster diaktifkan) yang ingin Anda hubungkan.
3. Untuk menemukan titik akhir Konfigurasi klaster, pilih nama klaster (bukan tombol radio).
4. Titik akhir konfigurasi ditampilkan di bagian Detail klaster. Untuk menyalinnya, pilih ikon salin di sebelah kiri titik akhir.

## Menemukan Titik Akhir (AWS CLI)

Anda dapat menggunakan AWS CLI untuk Amazon ElastiCache guna menemukan titik akhir untuk simpul, klaster, dan grup replikasi.

Topik

- Menemukan Titik Akhir untuk Simpul dan Klaster (AWS CLI)
- Menemukan Titik Akhir untuk Grup Replikasi (AWS CLI)

### Menemukan Titik Akhir untuk Simpul dan Klaster (AWS CLI)

Anda dapat menggunakan AWS CLI guna menemukan titik akhir untuk klaster dan simpulnya dengan perintah `describe-cache-clusters`. Untuk klaster Redis, perintah tersebut akan menampilkan titik akhir klaster. Jika Anda menyertakan parameter opsional `--show-cache-node-info`, perintah tersebut juga akan menampilkan titik akhir simpul individual di klaster.

Example

Perintah berikut mengambil informasi klaster untuk klaster Redis (mode klaster dinonaktifkan) simpul tunggal mycluster.

> **Important**
>
> Parameter `--cache-cluster-id` dapat digunakan dengan ID klaster Redis (mode klaster dinonaktifkan) simpul tunggal atau ID simpul tertentu dalam grup replikasi Redis. `--cache-cluster-id` milik grup replikasi Redis adalah nilai 4 digit seperti `0001`. Jika `--cache-cluster-id` adalah ID klaster (simpul) dalam grup replikasi Redis, `replication-group-id` akan disertakan dalam output.

Untuk Linux, macOS, atau Unix:

```
aws elasticache describe-cache-clusters \
    --cache-cluster-id redis-cluster \
    --show-cache-node-info
```

Untuk Windows:

```
aws elasticache describe-cache-clusters ^
```

```
    --cache-cluster-id redis-cluster ^
    --show-cache-node-info
```

Output dari operasi di atas akan terlihat seperti berikut (format JSON).

```
{
    "CacheClusters": [
        {
            "CacheClusterStatus": "available",
            "SecurityGroups": [
                {
                    "SecurityGroupId": "sg-77186e0d",
                    "Status": "active"
                }
            ],
            "CacheNodes": [
                {
                    "CustomerAvailabilityZone": "us-east-1b",
                    "CacheNodeCreateTime": "2018-04-25T18:19:28.241Z",
                    "CacheNodeStatus": "available",
                    "CacheNodeId": "0001",
                    "Endpoint": {
                        "Address": "redis-cluster.amazonaws.com",
                        "Port": 6379
                    },
                    "ParameterGroupStatus": "in-sync"
                }
            ],
            "AtRestEncryptionEnabled": false,
            "CacheClusterId": "redis-cluster",
            "TransitEncryptionEnabled": false,
            "CacheParameterGroup": {
                "ParameterApplyStatus": "in-sync",
                "CacheNodeIdsToReboot": [],
                "CacheParameterGroupName": "default.redis3.2"
            },
            "NumCacheNodes": 1,
            "PreferredAvailabilityZone": "us-east-1b",
            "AutoMinorVersionUpgrade": true,
            "Engine": "redis",
            "AuthTokenEnabled": false,
            "PendingModifiedValues": {},
            "PreferredMaintenanceWindow": "tue:08:30-tue:09:30",
```

```
            "CacheSecurityGroups": [],
            "CacheSubnetGroupName": "default",
            "CacheNodeType": "cache.t2.small",
             "DataTiering": "disabled"
            "EngineVersion": "3.2.10",
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "CacheClusterCreateTime": "2018-04-25T18:19:28.241Z"
        }
    ]
}
```

Untuk informasi selengkapnya, lihat topik describe-cache-clusters.

Menemukan Titik Akhir untuk Grup Replikasi (AWS CLI)

Anda dapat menggunakan AWS CLI guna menemukan titik akhir untuk grup replikasi dan klasternya dengan perintah `describe-replication-groups`. Perintah ini menampilkan titik akhir primer grup replikasi dan daftar semua klaster (simpul) dalam grup replikasi dengan titik akhirnya, bersama dengan titik akhir pembaca.

Operasi berikut mengambil titik akhir primer dan titik akhir pembaca untuk grup replikasi `myreplgroup`. Gunakan titik akhir primer untuk semua operasi tulis.

```
aws elasticache describe-replication-groups \
    --replication-group-id myreplgroup
```

Untuk Windows:

```
aws elasticache describe-replication-groups ^
    --replication-group-id myreplgroup
```

Keluaran dari operasi ini terlihat seperti berikut (format JSON).

```
{
   "ReplicationGroups": [
     {
       "Status": "available",
       "Description": "test",
       "NodeGroups": [
         {
            "Status": "available",
```

```
            "NodeGroupMembers": [
               {
                  "CurrentRole": "primary",
                  "PreferredAvailabilityZone": "us-west-2a",
                  "CacheNodeId": "0001",
                  "ReadEndpoint": {
                     "Port": 6379,
                     "Address": "myreplgroup-001.amazonaws.com"
                  },
                  "CacheClusterId": "myreplgroup-001"
               },
               {
                  "CurrentRole": "replica",
                  "PreferredAvailabilityZone": "us-west-2b",
                  "CacheNodeId": "0001",
                  "ReadEndpoint": {
                     "Port": 6379,
                     "Address": "myreplgroup-002.amazonaws.com"
                  },
                  "CacheClusterId": "myreplgroup-002"
               },
               {
                  "CurrentRole": "replica",
                  "PreferredAvailabilityZone": "us-west-2c",
                  "CacheNodeId": "0001",
                  "ReadEndpoint": {
                     "Port": 6379,
                     "Address": "myreplgroup-003.amazonaws.com"
                  },
                  "CacheClusterId": "myreplgroup-003"
               }
            ],
            "NodeGroupId": "0001",
            "PrimaryEndpoint": {
               "Port": 6379,
               "Address": "myreplgroup.amazonaws.com"
            },
            "ReaderEndpoint": {
               "Port": 6379,
               "Address": "myreplgroup-ro.amazonaws.com"
            }
         }
      ],
      "ReplicationGroupId": "myreplgroup",
```

```
         "AutomaticFailover": "enabled",
         "SnapshottingClusterId": "myreplgroup-002",
         "MemberClusters": [
            "myreplgroup-001",
            "myreplgroup-002",
            "myreplgroup-003"
         ],
         "PendingModifiedValues": {}
      }
   ]
}
```

Untuk informasi selengkapnya, lihat describe-replication-groups dalam Referensi Perintah AWS CLI.

## Menemukan Titik Akhir (API ElastiCache)

Anda dapat menggunakan API Amazon ElastiCache guna menemukan titik akhir untuk simpul, klaster, dan grup replikasi.

Topik

- Menemukan Titik Akhir untuk Simpul dan Klaster (API ElastiCache)
- Menemukan Titik Akhir untuk Grup Replikasi (API ElastiCache)

### Menemukan Titik Akhir untuk Simpul dan Klaster (API ElastiCache)

Anda dapat menggunakan API ElastiCache guna menemukan titik akhir untuk klaster dan simpulnya dengan perintah `DescribeCacheClusters`. Untuk klaster Redis, perintah tersebut akan menampilkan titik akhir klaster. Jika Anda menyertakan parameter opsional `ShowCacheNodeInfo`, tindakan tersebut juga akan menampilkan titik akhir simpul individual di klaster.

Example

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeCacheClusters
    &CacheClusterId=mycluster
    &ShowCacheNodeInfo=true
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &Version=2015-02-02
    &X-Amz-Credential=<credential>
```

### Menemukan Titik Akhir untuk Grup Replikasi (API ElastiCache)

Anda dapat menggunakan API ElastiCache guna menemukan titik akhir untuk grup replikasi dan klasternya dengan tindakan `DescribeReplicationGroups`. Perintah ini menampilkan titik akhir primer grup replikasi dan daftar semua klaster dalam grup replikasi dengan titik akhirnya, bersama dengan titik akhir pembaca.

Operasi berikut mengambil titik akhir primer (PrimaryEndpoint), titik akhir pembaca (ReaderEndpoint), dan titik akhir simpul individual (ReadEndpoint) untuk grup replikasi `myreplgroup`. Gunakan titik akhir primer untuk semua operasi tulis.

```
https://elasticache.us-west-2.amazonaws.com/
```

```
    ?Action=DescribeReplicationGroups
    &ReplicationGroupId=myreplgroup
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &Version=2015-02-02
    &X-Amz-Credential=<credential>
```

Untuk informasi selengkapnya, lihat DescribeReplicationGroups.

## Menggunakan serpihan

Serpihan (API/CLI: grup simpul) adalah kumpulan dari satu hingga enam simpul Redis. Klaster Redis (mode klaster nonaktif) tidak akan pernah memiliki lebih dari satu serpihan. Dengan pecahan, Anda dapat memisahkan database besar menjadi bagian yang lebih kecil, lebih cepat, dan lebih mudah dikelola yang disebut pecahan data. Hal ini dapat meningkatkan efisiensi database dengan mendistribusikan operasi di beberapa bagian terpisah. Menggunakan pecahan dapat menawarkan banyak manfaat termasuk peningkatan kinerja, skalabilitas, dan efisiensi biaya.

Anda dapat membuat klaster dengan jumlah serpihan lebih banyak dan jumlah replika lebih sedikit dengan jumlah hingga 90 simpul per klaster. Konfigurasi klaster ini dapat berkisar dari 90 serpihan dan 0 replika hingga 15 serpihan dan 5 replika, yang merupakan jumlah replika maksimum yang diperbolehkan. Data klaster dipartisi di seluruh serpihan klaster. Jika di dalam serpihan terdapat lebih dari satu simpul, serpihan akan mengimplementasikan replikasi dengan satu simpul menjadi simpul primer baca/tulis dan simpul lain menjadi simpul replika hanya baca.

Batas simpul atau serpihan dapat ditingkatkan hingga maksimum 500 per klaster jika versi mesin Redis yang digunakan adalah versi 5.0.6 atau lebih tinggi. Sebagai contoh, Anda dapat memilih untuk membuat konfigurasi dari sebuah klaster dengan 500 simpul yang berkisar antara 83 serpihan (satu primer dan 5 replika per serpihan) dan 500 serpihan (primer tunggal dan tidak ada replika). Pastikan alamat IP yang tersedia mencukupi untuk mengakomodasi peningkatan tersebut. Kesalahan umum termasuk subnet dalam grup subnet memiliki rentang CIDR yang terlalu kecil atau subnet digunakan bersama dan banyak digunakan oleh klaster lain. Untuk informasi selengkapnya, lihat Membuat grup subnet.

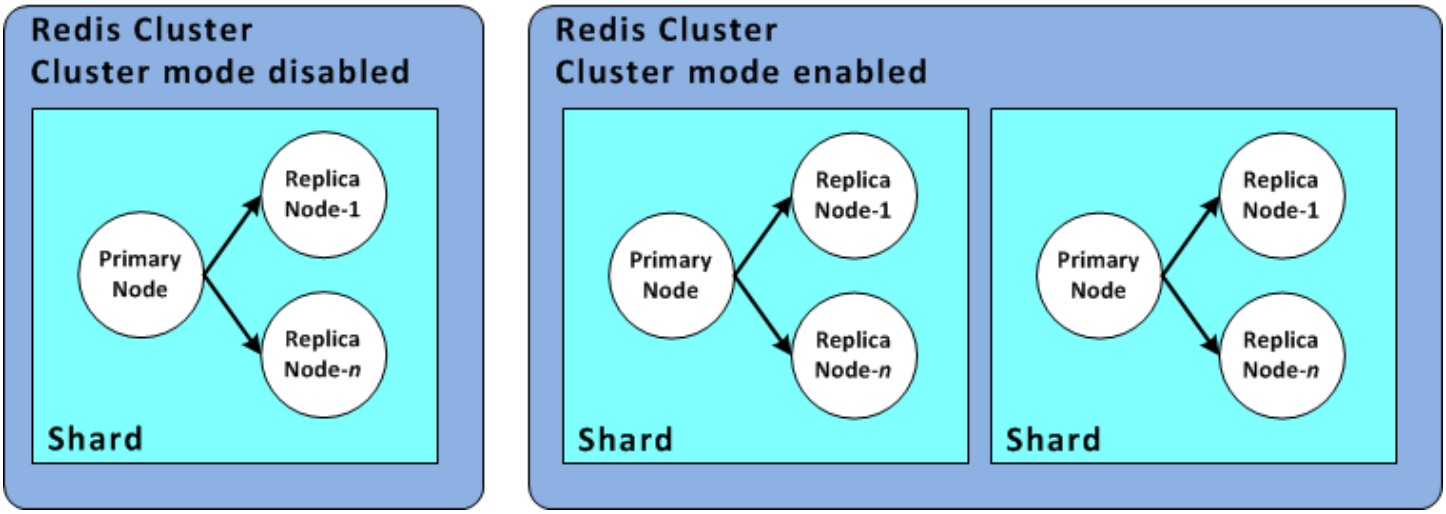
Untuk versi di bawah 5.0.6, batasnya adalah 250 per klaster.

Untuk meminta penambahan batas, lihat Batas layanan AWS dan pilih jenis batas Simpul per klaster per jenis instans.

Saat Anda membuat kluster Redis (mode cluster diaktifkan) menggunakan ElastiCache konsol, Anda menentukan jumlah pecahan di cluster dan jumlah node dalam pecahan. Untuk informasi selengkapnya, lihat Membuat klaster Redis (mode klaster diaktifkan) (Konsol). Jika Anda menggunakan ElastiCache API atau AWS CLI untuk membuat cluster (disebut grup replikasi di API/CLI), Anda dapat mengonfigurasi jumlah node dalam pecahan (API/CLI: grup node) secara independen. Untuk informasi selengkapnya, lihat hal berikut:

- API: CreateReplicationGroup
- CLI: create-replication-group

Setiap simpul dalam serpihan memiliki spesifikasi komputasi, penyimpanan, dan memori yang sama. ElastiCache API memungkinkan Anda mengontrol atribut shard-wide, seperti jumlah node, pengaturan keamanan, dan jendela pemeliharaan sistem.




Konfigurasi serpihan Redis

Untuk informasi lain, lihat Resharding dan penyeimbangan ulang serpihan secara offline untuk Redis (mode klaster diaktifkan) dan Resharding dan penyeimbangan ulang serpihan secara online untuk Redis (mode klaster diaktifkan).

## Menemukan ID serpihan

Anda dapat menemukan ID pecahan menggunakan AWS Management Console, the AWS CLI atau ElastiCache API.

Menggunakan AWS Management Console

Topik

- Untuk Redis (Mode Klaster Nonaktif)
- Untuk Redis (Mode Klaster Aktif)

### Untuk Redis (Mode Klaster Nonaktif)

ID serpihan grup replikasi Redis (mode klaster nonaktif) selalu `0001`.

### Untuk Redis (Mode Klaster Aktif)

Prosedur berikut menggunakan AWS Management Console untuk menemukan ID pecahan grup replikasi Redis (mode cluster diaktifkan).

Untuk menemukan ID serpihan dalam grup replikasi Redis (mode klaster aktif)

1. Masuk ke AWS Management Console dan buka ElastiCache konsol di https://console.aws.amazon.com/elasticache/.
2. Pada panel navigasi, pilih Redis, kemudian pilih nama grup replikasi Redis (mode klaster aktif) yang ingin dicari ID serpihannya.
3. Pada kolom Nama Serpihan, ID serpihan adalah empat digit terakhir dari nama serpihan.

### Menggunakan AWS CLI

Untuk menemukan id pecahan (grup simpul) untuk grup replikasi Redis (mode cluster dinonaktifkan) atau Redis (mode cluster diaktifkan) gunakan AWS CLI operasi `describe-replication-groups` dengan parameter opsional berikut.

- **`--replication-group-id`**—Parameter opsional yang jika digunakan akan membatasi output pada detail grup replikasi yang ditentukan. Jika parameter ini dihilangkan, detail hingga 100 grup replikasi akan ditampilkan.

#### Example

Perintah ini akan menampilkan detail untuk `sample-repl-group`.

Untuk Linux, macOS, atau Unix:

```
aws elasticache describe-replication-groups \
```

```
    --replication-group-id sample-repl-group
```

Untuk Windows:

```
aws elasticache describe-replication-groups ^
    --replication-group-id sample-repl-group
```

Output dari perintah ini akan terlihat seperti ini. ID serpihan (grup simpul) *disorot* di sini agar lebih mudah ditemukan.

```
{
    "ReplicationGroups": [
        {
            "Status": "available",
            "Description": "2 shards, 2 nodes (1 + 1 replica)",
            "NodeGroups": [
                {
                    "Status": "available",
                    "Slots": "0-8191",
                    "NodeGroupId": "0001",
                    "NodeGroupMembers": [
                        {
                            "PreferredAvailabilityZone": "us-west-2c",
                            "CacheNodeId": "0001",
                            "CacheClusterId": "sample-repl-group-0001-001"
                        },
                        {
                            "PreferredAvailabilityZone": "us-west-2a",
                            "CacheNodeId": "0001",
                            "CacheClusterId": "sample-repl-group-0001-002"
                        }
                    ]
                },
                {
                    "Status": "available",
                    "Slots": "8192-16383",
                    "NodeGroupId": "0002",
                    "NodeGroupMembers": [
                        {
                            "PreferredAvailabilityZone": "us-west-2b",
                            "CacheNodeId": "0001",
                            "CacheClusterId": "sample-repl-group-0002-001"
                        },
```

```
                        {
                            "PreferredAvailabilityZone": "us-west-2a",
                            "CacheNodeId": "0001",
                            "CacheClusterId": "sample-repl-group-0002-002"
                        }
                    ]
                }
            ],
            "ConfigurationEndpoint": {
                "Port": 6379,
                "Address": "sample-repl-
group.9dcv5r.clustercfg.usw2.cache.amazonaws.com"
            },
            "ClusterEnabled": true,
            "ReplicationGroupId": "sample-repl-group",
            "SnapshotRetentionLimit": 1,
            "AutomaticFailover": "enabled",
            "SnapshotWindow": "13:00-14:00",
            "MemberClusters": [
                "sample-repl-group-0001-001",
                "sample-repl-group-0001-002",
                "sample-repl-group-0002-001",
                "sample-repl-group-0002-002"
            ],
            "CacheNodeType": "cache.m3.medium",
            "DataTiering": "disabled",
            "PendingModifiedValues": {}
        }
    ]
}
```

Menggunakan ElastiCache API

Untuk menemukan id pecahan (grup simpul) untuk grup replikasi Redis (mode cluster dinonaktifkan) atau Redis (mode cluster diaktifkan) gunakan AWS CLI operasi `describe-replication-groups` dengan parameter opsional berikut.

- **`ReplicationGroupId`**—Parameter opsional yang jika digunakan akan membatasi output pada detail grup replikasi yang ditentukan. Jika parameter ini dihilangkan, detail hingga *`xxx`* grup replikasi akan ditampilkan.

Example

Perintah ini akan menampilkan detail untuk `sample-repl-group`.

Untuk Linux, macOS, atau Unix:

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeReplicationGroup
   &ReplicationGroupId=sample-repl-group
   &Version=2015-02-02
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

# Membandingkan cache yang dirancang sendiri dari Memcached dan Redis

Amazon ElastiCache mendukung mesin cache Memcached dan Redis. Setiap mesin menyediakan beberapa kelebihan. Gunakan informasi dalam topik ini untuk membantu Anda memilih mesin dan versi yang paling sesuai dengan kebutuhan Anda.

> **Important**
>
> Setelah Anda membuat klaster cache atau grup replikasi, Anda dapat memutakhirkan versi mesin ke versi lebih baru, tetapi Anda tidak dapat menurunkan ke versi mesin yang lebih lama. Jika Anda ingin menggunakan versi mesin yang lama, Anda harus menghapus klaster cache atau grup replikasi yang sudah ada dan membuatnya lagi dengan versi mesin yang lama tersebut.

Secara umum, mesin yang ada terlihat serupa. Setiap mesin merupakan penyimpanan nilai-kunci dalam memori. Namun, dalam praktiknya terdapat perbedaan yang signifikan.

Pilih Memcached jika hal berikut berlaku untuk Anda:

- Anda membutuhkan model yang paling sederhana.
- Anda perlu menjalankan simpul besar dengan beberapa inti atau thread.

- Anda membutuhkan kemampuan untuk menskalakan ke luar dan ke dalam, yakni menambahkan dan menghapus simpul seiring peningkatan dan penurunan permintaan pada sistem.
- Anda perlu menyimpan obyek ke dalam cache.

Pilih Redis dengan versi ElastiCache for Redis jika hal berikut berlaku untuk Anda:

- ElastiCache for Redis versi 7.0 (Ditingkatkan)

  Anda ingin menggunakan Redis Functions, Sharded Pub/Sub, atau Redis ACL improvements. Untuk informasi selengkapnya, lihat Redis Versi 7.0 (Ditingkatkan).
- ElastiCache for Redis versi 6.2 (Ditingkatkan)

  Anda ingin kemampuan untuk mengatur tingkatan data antara memori dan SSD menggunakan tipe node r6gd. Untuk informasi selengkapnya, lihat Tingkatan data.
- ElastiCache for Redis versi 6.0 (Ditingkatkan)

  Anda ingin mengautentikasi pengguna dengan kontrol akses berbasis peran.

  Untuk informasi selengkapnya, lihat Redis Versi 6.0 (Ditingkatkan).
- ElastiCache for Redis versi 5.0.0 (Ditingkatkan)

  Anda ingin menggunakan Redis streams, yakni struktur data log yang memungkinkan produsen untuk menambahkan item baru secara real-time dan juga memungkinkan konsumen untuk mengonsumsi pesan baik dalam mode blok atau non-blok.

  Untuk informasi selengkapnya, lihat Redis Versi 5.0.0 (Ditingkatkan).
- ElastiCache for Redis versi 4.0.10 (Ditingkatkan)

  Mendukung enkripsi serta secara dinamis menambahkan atau menghapus serpihan dari klaster Redis (mode klaster aktif).

  Untuk informasi selengkapnya, lihat Redis Versi 4.0.10 (Ditingkatkan).

Versi berikut tidak digunakan lagi, telah mencapai atau segera mencapai akhir masa pakainya.

- ElastiCache for Redis versi 3.2.10 (Ditingkatkan)

Mendukung kemampuan untuk secara dinamis menambahkan atau menghapus serpihan dari klaster Redis (mode klaster aktif).

> **Important**
>
> Saat ini ElastiCache for Redis 3.2.10 tidak mendukung enkripsi.

Untuk informasi selengkapnya, lihat berikut ini:

- Redis versi 3.2.10 (Ditingkatkan)
- Praktik terbaik resharding online untuk Redis, Untuk informasi selengkapnya, lihat referensi berikut:
  - Praktik Terbaik: Resharding Online
  - Resharding dan Penyeimbangan Ulang Serpihan Secara Online untuk Redis (Mode Klaster Aktif)
- Untuk informasi selengkapnya tentang penskalaan klaster Redis, lihat Penskalaan.

- ElastiCache for Redis versi 3.2.6 (Ditingkatkan)

  Jika Anda membutuhkan fungsionalitas dari versi Redis sebelumnya ditambah fitur berikut, pilih ElastiCache for Redis 3.2.6:

  - Enkripsi saat dalam pengiriman. Untuk informasi selengkapnya, lihat Enkripsi Saat Dalam Pengiriman Amazon ElastiCache for Redis.
  - Enkripsi saat dalam penyimpanan. Untuk informasi selengkapnya, lihat Enkripsi Saat Transit Amazon ElastiCache for Redis.
- ElastiCache for Redis (Mode klaster aktif) versi 3.2.4

  Jika Anda membutuhkan fungsionalitas Redis 2.8.x ditambah fitur berikut, pilih Redis 3.2.4 (mode berklaster):

  - Anda perlu membuat partisi data Anda di dua hingga 500 grup simpul (khusus mode berklaster).
  - Anda membutuhkan pengindeksan geospasial (mode berklaster atau mode tanpa klaster).
  - Anda tidak perlu mendukung beberapa basis data.
- ElastiCache for Redis (mode tanpa klaster) 2.8.x dan 3.2.4 (Ditingkatkan)

  Jika hal berikut berlaku untuk Anda, pilih Redis 2.8.x atau Redis 3.2.4 (mode tanpa klaster):

- Anda memerlukan jenis data yang kompleks, seperti string, hash, list, set, sorted set, dan bitmap.
- Anda perlu mengurutkan atau membuat peringkat set data dalam memori.
- Anda perlu persistensi pada penyimpanan kunci.
- Anda perlu mereplikasi data Anda dari primer ke satu atau beberapa replika baca untuk aplikasi dengan operasi baca intensif.
- Anda perlu melakukan failover otomatis jika simpul primer Anda gagal.
- Anda perlu mempublikasikan dan berlangganan kemampuan (pub/sub) — untuk memberitahu klien peristiwa di server.
- Anda memerlukan kemampuan pencadangan dan pemulihan.
- Anda perlu mendukung beberapa basis data.

Ringkasan perbandingan Memcached, Redis (mode klaster nonaktif), dan Redis (mode klaster aktif)

| | Memcached | Redis (mode klaster nonaktif) | Redis (mode klaster aktif) |
|---|---|---|---|
| Engine versions+ | 1.4.5 and later | 4.0.10 and later | 4.0.10 and later |
| Data types | Simple | 2.8.x - Complex * | 3.2.x and later - Complex |
| | | Complex | |
| Data partitioning | Yes | No | Yes |
| Cluster is modifiable | Yes | Yes | 3.2.10 and later - Limited |
| Online resharding | No | No | 3.2.10 and later |
| Encryption | in-transit 1.6.12 and later | 4.0.10 and later | 4.0.10 and later |
| Data tiering | No | 6.2 and later | 6.2 and later |
| Sertifikasi kepatuhan | | | |
| Compliance Certification | | | |
| FedRAMP | Ya - 1.6.12 dan yang lebih baru | 4.0.10 dan yang lebih baru | 4.0.10 dan yang lebih baru |
| HIPAA | Ya - 1.6.12 dan yang lebih baru | 4.0.10 dan yang lebih baru | 4.0.10 dan yang lebih baru |
| PCI DSS | Ya | 4.0.10 dan yang lebih baru | 4.0.10 dan yang lebih baru |
| Multi-threaded | Yes | No | No |
| Node type upgrade | No | Yes | Yes |
| Engine upgrading | Yes | Yes | Yes |

| | Memcached | Redis (mode klaster nonaktif) | Redis (mode klaster aktif) |
|---|---|---|---|
| High availability (replication) | No | Yes | Yes |
| Automatic failover | No | Optional | Required |
| Pub/Sub capabilities | No | Yes | Yes |
| Sorted sets | No | Yes | Yes |
| Backup and restore | No | Yes | Yes |
| Geospatial indexing | No | 4.0.10 and later | Yes |

Catatan:

string, objects (like databases)

* string, sets, sorted sets, lists, hashes, bitmaps, hyperloglog

string, sets, sorted sets, lists, hashes, bitmaps, hyperloglog, geospatial indexes

+ Excludes versions which are deprecated, have reached or soon to reach end of life.

Setelah Anda memilih mesin untuk klaster Anda, sebaiknya gunakan versi terbaru mesin tersebut. Untuk informasi selengkapnya, lihat Versi ElastiCache for Memcached yang Didukung atau Versi ElastiCache for Redis yang Didukung.

# Migrasi online ke ElastiCache

Dengan Migrasi Online, Anda dapat memigrasikan data Anda dari Redis sumber terbuka yang di-hosting sendiri di Amazon EC2 ke Amazon ElastiCache.

> **Note**
>
> Migrasi online tidak didukung ke cache nirserver ElastiCache atau klaster yang berjalan pada tipe simpul r6gd.

## Gambaran Umum

Migrasi data dari Redis sumber terbuka yang berjalan di Amazon EC2 ke Amazon ElastiCache memerlukan deployment Amazon ElastiCache yang sudah ada atau yang baru dibuat. Deployment harus memiliki konfigurasi yang siap untuk migrasi. Ini juga harus selaras dengan konfigurasi yang Anda inginkan, termasuk atribut seperti jenis instans, jumlah serpihan, dan jumlah replika.

Migrasi online dirancang untuk migrasi data dari Redis sumber terbuka yang di-hosting di Amazon EC2 ke ElastiCache for Redis dan bukan di antara klaster ElastiCache for Redis.

> **Important**
>
> Sangat direkomendasikan agar Anda membaca bagian berikut secara keseluruhan sebelum memulai proses migrasi online.

Migrasi dimulai saat Anda memanggil operasi API `StartMigration` atau perintah AWS CLI. Untuk mode klaster Redis dinonaktifkan, proses migrasi menjadikan simpul primer klaster ElastiCache for Redis sebagai replika Redis primer sumber Anda. Untuk mode klaster Redis diaktifkan, proses migrasi menjadikan simpul primer setiap serpihan ElastiCache sebagai replika serpihan klaster sumber Anda yang memiliki slot yang sama.

Setelah perubahan sisi klien siap, panggil operasi API `CompleteMigration`. Operasi API ini mempromosikan deployment ElastiCache Anda ke deployment Redis primer Anda dengan simpul primer dan replika (sebagaimana berlaku). Sekarang Anda dapat mengalihkan aplikasi klien Anda untuk mulai menulis data ke ElastiCache. Dalam proses migrasi, Anda dapat memeriksa status replikasi dengan menjalankan perintah redis-cli INFO pada simpul Redis Anda dan pada simpul primer ElastiCache.

## Langkah migrasi

Topik berikut menguraikan proses untuk melakukan migrasi data Anda:

- Mempersiapkan simpul Redis sumber dan target Anda untuk migrasi
- Menguji migrasi data
- Memulai migrasi
- Memverifikasi kemajuan migrasi data

- Menyelesaikan migrasi data

## Mempersiapkan simpul Redis sumber dan target Anda untuk migrasi

Anda harus memastikan bahwa keempat prasyarat yang disebutkan berikut ini terpenuhi sebelum Anda memulai migrasi dari konsol ElastiCache, API, atau AWS CLI.

Untuk mempersiapkan simpul Redis sumber dan target Anda untuk migrasi

1. Identifikasi deployment ElastiCache target dan pastikan Anda dapat memigrasikan data ke target.

   Deployment ElastiCache yang sudah ada atau yang baru dibuat harus memenuhi persyaratan berikut untuk migrasi:

   - Simpul menggunakan mesin Redis versi 5.0.6 atau lebih tinggi.
   - Simpul tidak mengaktifkan baik enkripsi data bergerak ataupun enkripsi data diam.
   - Simpul mengaktifkan Multi-AZ.
   - Memori simpul mencukupi untuk memuat data dari klaster Redis Anda. Untuk mengonfigurasi pengaturan memori cadangan yang tepat, lihat Mengelola Memori Terpesan.
   - Untuk mode klaster dinonaktifkan, Anda dapat bermigrasi langsung dari Redis versi 2.8.21 dan seterusnya ke Redis versi 5.0.6 jika Anda menggunakan CLI atau Redis konsol. Untuk mode klaster diaktifkan, Anda dapat bermigrasi langsung dari Redis dengan mode klaster aktif ke Redis versi 5.0.6 dan seterusnya jika Anda menggunakan CLI atau Redis versi 5.0.6 jika menggunakan CLI atau konsol.
   - Jumlah serpihan dalam sumber dan target cocok.
   - Bukan bagian dari penyimpanan data global.
   - Menonaktifkan tingkat data.
2. Pastikan bahwa konfigurasi Redis sumber terbuka Anda dan deployment ElastiCache for Redis kompatibel.

   Minimal, semua hal berikut dalam deployment ElastiCache target harus kompatibel dengan konfigurasi Redis Anda untuk replikasi Redis:

   - Klaster Redis Anda tidak boleh mengaktifkan AUTH Redis.
   - Config Redis `protected-mode` harus diatur ke `no`.

- Jika Anda memiliki konfigurasi `bind` di config Redis Anda, konfigurasi harus diperbarui untuk mengizinkan permintaan dari simpul ElastiCache.
- Jumlah basis data logis harus sama antara simpul ElastiCache dan klaster Redis Anda. Nilai ini diatur menggunakan `databases` dalam config Redis.
- Perintah Redis yang melakukan modifikasi data tidak boleh diganti namanya untuk memungkinkan berhasilnya replikasi data. misalnya `sync`, `psync`, `info`, `config`, `command`, dan `cluster`.
- Untuk mereplikasi data dari klaster Redis Anda ke ElastiCache, pastikan ada CPU dan memori yang mencukupi untuk menangani beban tambahan ini. Beban ini berasal dari file RDB yang dibuat oleh klaster Redis Anda dan ditransfer melalui jaringan ke simpul ElastiCache.
- Semua instans redis di klaster sumber harus berjalan di port yang sama.

3. Pastikan instans Anda dapat tersambung ke ElastiCache dengan melakukan hal berikut:
   - Pastikan bahwa alamat IP instans Anda adalah privat.
   - Tugaskan atau buat deployment ElastiCache di cloud privat virtual (VPC) yang sama dengan Redis pada instans Anda (direkomendasikan).
   - Jika VPC berbeda, siapkan peering VPC untuk mengizinkan akses antara simpul. Untuk informasi selengkapnya tentang peering VPC, lihat Pola Akses untuk Mengakses Cache ElastiCache di Amazon VPC.
   - Grup keamanan yang terlampir pada instans Redis Anda harus mengizinkan lalu lintas masuk dari simpul ElastiCache.
4. Pastikan aplikasi Anda dapat mengarahkan lalu lintas ke simpul ElastiCache setelah migrasi data selesai. Untuk informasi selengkapnya, lihat Pola Akses untuk Mengakses Cache ElastiCache di Amazon VPC.

## Menguji migrasi data

Setelah semua prasyarat selesai, Anda dapat memvalidasi pengaturan migrasi menggunakan AWS Management Console, API ElastiCache, atau AWS CLI. Contoh berikut menunjukkan penggunaan CLI.

Uji migrasi dengan memanggil perintah `test-migration` dengan parameter berikut:

- `--replication-group-id`— ID grup replikasi yang menjadi destinasi migrasi data.

- `--customer-node-endpoint-list`— Daftar titik akhir sumber data yang akan dimigrasikan. Daftar harus memiliki hanya satu elemen.

Contoh berikut menunjukkan penggunaan CLI.

```
aws elasticache test-migration --replication-group-id test-cluster --customer-node-
endpoint-list "Address='10.0.0.241',Port=6379"
```

ElastiCache akan memvalidasi pengaturan migrasi tanpa migrasi data aktual.

## Memulai migrasi

Setelah semua prasyarat selesai, Anda dapat memulai migrasi data menggunakan AWS Management Console, API ElastiCache, atau AWS CLI. Untuk mode klaster diaktifkan, jika migrasi slot berbeda, resharding akan dilakukan sebelum migrasi langsung. Contoh berikut menunjukkan penggunaan CLI.

> **Note**
>
> Kami merekomendasikan untuk menggunakan API `TestMigration` untuk memvalidasi penyiapan migrasi. Tapi ini benar-benar opsional.

Mulai migrasi dengan memanggil perintah `start-migration` dengan parameter berikut:

- `--replication-group-id` – Pengidentifikasi grup replikasi ElastiCache target
- `--customer-node-endpoint-list` – Daftar titik akhir baik dengan DNS atau alamat IP dan port tempat klaster Redis sumber Anda berjalan. Daftar ini hanya dapat mengambil satu elemen baik untuk mode klaster dinonaktifkan dan mode klaster diaktifkan. Jika Anda telah mengaktifkan replikasi berantai, titik akhir dapat menunjuk ke replika, alih-alih simpul primer pada klaster Redis Anda.

Contoh berikut menunjukkan penggunaan CLI.

```
aws elasticache start-migration --replication-group-id test-cluster --customer-node-
endpoint-list "Address='10.0.0.241',Port=6379"
```

Saat Anda menjalankan perintah ini, simpul primer ElastiCache (di setiap serpihan) mengonfigurasi dirinya sendiri untuk menjadi replika dari instans Redis Anda (di serpihan yang memiliki slot yang sama di Redis dengan klaster aktif). Status klaster ElastiCache berubah ke bermigrasi dan data mulai bermigrasi dari instans Redis Anda ke simpul primer ElastiCache. Tergantung pada ukuran data dan beban pada instans Redis Anda, penyelesaian migrasi dapat memakan waktu. Anda dapat memeriksa kemajuan migrasi dengan menjalankan perintah redis-cli INFO pada instans Redis dan simpul primer ElastiCache Anda.

Setelah replikasi berhasil, semua penulisan ke instans Redis Anda akan disebarkan ke klaster ElastiCache. Anda dapat menggunakan simpul ElastiCache untuk proses baca. Namun, Anda tidak dapat menulis ke klaster ElastiCache. Jika simpul primer ElastiCache tersambung ke simpul replika lain, simpul replika ini terus mereplikasi dari simpul primer ElastiCache. Dengan cara ini, semua data dari klaster Redis Anda akan direplikasikan ke semua simpul di klaster ElastiCache.

Jika simpul primer ElastiCache tidak dapat menjadi replika instans Redis Anda, simpul tersebut mencoba beberapa kali sebelum akhirnya mempromosikan dirinya kembali ke primer. Status klaster ElastiCache lalu berubah menjadi tersedia, dan peristiwa grup replikasi tentang kegagalan untuk memulai migrasi akan dikirim. Untuk memecahkan masalah seperti kegagalan tersebut, periksa hal berikut:

- Lihat peristiwa grup replikasi. Gunakan informasi spesifik dari peristiwa untuk memperbaiki kegagalan migrasi.
- Jika peristiwa tidak memberikan informasi spesifik apa pun, pastikan bahwa Anda telah mengikuti pedoman di Mempersiapkan simpul Redis sumber dan target Anda untuk migrasi.
- Pastikan bahwa konfigurasi perutean untuk VPC dan subnet Anda mengizinkan lalu lintas antara simpul ElastiCache dan instans Redis Anda.
- Pastikan grup keamanan yang terlampir pada instans Redis Anda mengizinkan lalu lintas masuk dari simpul ElastiCache.
- Periksa log Redis untuk instans Redis Anda untuk informasi selengkapnya tentang kegagalan khusus replikasi.

## Memverifikasi kemajuan migrasi data

Setelah migrasi data dimulai, Anda dapat melakukan hal berikut untuk melacak kemajuannya:

- Verifikasi bahwa `master_link_status` Redis adalah up dalam perintah `INFO` pada simpul primer ElastiCache. Anda juga dapat menemukan informasi ini di konsol ElastiCache. Pilih klaster

dan di bawah Metrik CloudWatch, amati Status Kondisi Tautan Primer. Setelah nilainya mencapai 1, itu berarti data sudah sinkron.

- Anda dapat memeriksa bahwa replika ElastiCache memiliki status online dengan menjalankan perintah `INFO` di instans Redis Anda. Melakukan hal ini juga menyediakan informasi tentang lag replikasi.
- Verifikasi buffer output klien rendah menggunakan perintah CLIENT LIST pada instans Redis.

Setelah migrasi data selesai, data sinkron dengan segala proses tulis baru yang datang ke simpul primer pada klaster Redis Anda.

## Menyelesaikan migrasi data

Saat Anda siap untuk beralih ke klaster ElastiCache, gunakan perintah `complete-migration` CLI dengan parameter berikut:

- `--replication-group-id` – Pengidentifikasi untuk grup replikasi.
- `--force` – Nilai yang memaksa migrasi untuk berhenti tanpa memastikan bahwa data sudah sinkron.

Berikut adalah contohnya.

```
aws elasticache complete-migration --replication-group-id test-cluster
```

Saat Anda menjalankan perintah ini, simpul primer ElastiCache (di setiap serpihan) berhenti mereplikasi dari instans Redis Anda dan mempromosikannya menjadi primer. Promosi ini biasanya selesai dalam beberapa menit. Untuk mengonfirmasi promosi ke primer, periksa peristiwa `Complete Migration successful for test-cluster`. Pada titik ini, Anda dapat mengarahkan aplikasi Anda ke baca dan tulis ElastiCache. Status klaster ElastiCache harus berubah dari bermigrasi menjadi tersedia.

Jika promosi ke primer gagal, simpul primer ElastiCache terus mereplikasi dari instans Redis Anda. Klaster ElastiCache terus berada dalam status bermigrasi, dan olahpesan peristiwa grup replikasi tentang kegagalan dikirim. Untuk memecahkan masalah kegagalan ini, lihat hal berikut:

- Periksa peristiwa grup replikasi. Gunakan informasi spesifik dari peristiwa itu untuk memperbaiki kegagalan.

- Anda mungkin mendapatkan pesan peristiwa tentang data yang tidak sinkron. Jika demikian, pastikan bahwa primer ElastiCache dapat mereplikasi dari instans Redis Anda dan keduanya sudah sinkron. Jika Anda masih ingin menghentikan migrasi, Anda dapat menjalankan perintah sebelumnya dengan opsi `–force`.
- Anda mungkin mendapatkan olahpesan peristiwa jika salah satu simpul ElastiCache sedang dalam penggantian. Anda dapat mencoba lagi menyelesaikan langkah migrasi setelah penggantian selesai.

## Melakukan migrasi data online menggunakan Konsol

Anda dapat menggunakan AWS Management Console untuk memigrasikan data Anda dari klaster Anda ke klaster Redis.

Melakukan migrasi data online menggunakan Konsol

1. Masuk ke konsol dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Buat klaster Redis baru atau pilih klaster yang telah ada. Pastikan bahwa klaster memenuhi persyaratan berikut:
   - Versi mesin Redis Anda harus 5.0.6 atau lebih tinggi.
   - Klaster Redis Anda tidak boleh mengaktifkan AUTH Redis.
   - Config Redis `protected-mode` harus diatur ke `no`.
   - Jika Anda memiliki konfigurasi `bind` di config Redis Anda, konfigurasi harus diperbarui untuk mengizinkan permintaan dari simpul ElastiCache.
   - Jumlah basis data harus sama antara simpul ElastiCache dan klaster Redis Anda. Nilai ini diatur menggunakan `databases` dalam config Redis.
   - Perintah Redis yang melakukan modifikasi data tidak boleh diganti namanya untuk memungkinkan berhasilnya replikasi data.
   - Untuk mereplikasi data dari klaster Redis Anda ke ElastiCache, pastikan ada CPU dan memori mencukupi untuk menangani beban tambahan ini. Beban ini berasal dari file RDB yang dibuat oleh klaster Redis Anda dan ditransfer melalui jaringan ke simpul ElastiCache.
   - Klaster berada dalam status tersedia.
3. Setelah memilih klaster, pilih Migrasikan Data dari Titik Akhiruntuk Tindakan.
4. Pada kotak dialog Migrasikan Data dari Titik Akhir, masukkan alamat IP, dan port tempat klaster Redis Anda tersedia.

> **Important**
>
> Alamat IP harus sama persis. Jika Anda salah memasukkan alamat, migrasi akan gagal.

5. Pilih Mulai Migrasi.

   Saat klaster memulai migrasi, statusnya berubah ke Mengubah lalu ke Bermigrasi.

6. Pantau kemajuan migrasi dengan memilih Peristiwa pada panel navigasi.

Anda dapat menghentikan migrasi kapan pun. Untuk melakukannya, pilih klaster Anda dan pilih Hentikan Migrasi Data untuk Tindakan. Klaster akan berubah status menjadi Tersedia.

Jika migrasi berhasil, status klaster akan menjadi Tersedia dan log peristiwa menunjukkan hal berikut:

`Migration operation succeeded for replication group` *`ElastiCacheClusterName`*`.`

Jika migrasi gagal, status klaster akan menjadi Tersedia dan log peristiwa menunjukkan hal berikut:

`Migration operation failed for replication group` *`ElastiCacheClusterName`*`.`

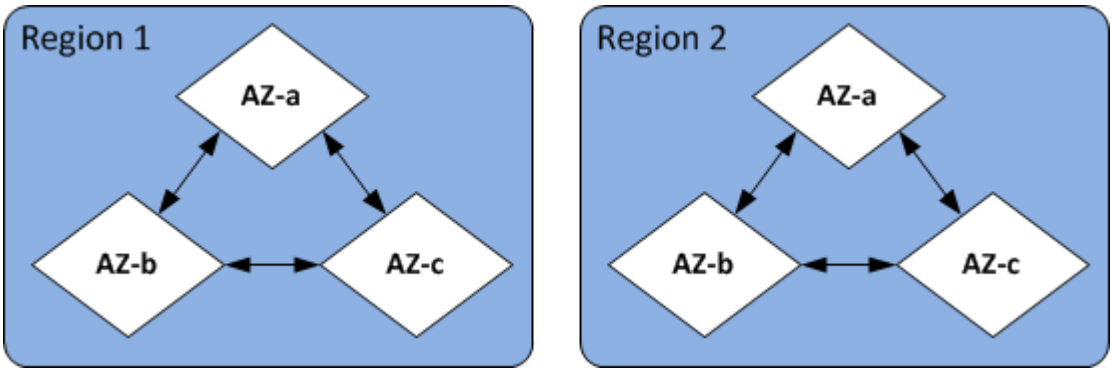
## Memilih wilayah dan zona ketersediaan

AWS Sumber daya komputasi awan ditempatkan di fasilitas pusat data yang sangat tersedia. Untuk memberikan skalabilitas dan keandalan tambahan, fasilitas pusat data ini ditempatkan di beberapa lokasi fisik yang berbeda. Lokasi ini dikategorikan berdasarkan wilayah dan Zona Ketersediaan.

AWS Daerah besar dan tersebar luas ke lokasi geografis yang terpisah. Availability Zone adalah lokasi berbeda dalam AWS Wilayah yang dirancang untuk diisolasi dari kegagalan di Availability Zone lainnya. Mereka menyediakan konektivitas jaringan latensi rendah yang murah ke Availability Zone lainnya di Wilayah yang sama AWS .

> **Important**
>
> Setiap wilayah adalah independen sepenuhnya. ElastiCache Aktivitas apa pun yang Anda mulai (misalnya, membuat kluster) hanya berjalan di wilayah default Anda saat ini.

Untuk membuat atau bekerja dengan klaster di wilayah tertentu, gunakan titik akhir layanan wilayah tersebut. Untuk titik akhir layanan, lihat Wilayah & titik akhir yang didukung.




Wilayah dan Zona Ketersediaan

Topik

- Menempatkan simpul Anda
- Wilayah & titik akhir yang didukung
- Menggunakan zona lokal dengan ElastiCache
- Menggunakan Outposts

## Menempatkan simpul Anda

Amazon ElastiCache mendukung lokasi semua node cluster dalam satu atau beberapa Availability Zones (AZ). Selanjutnya, jika Anda memilih untuk menemukan node Anda di beberapa AZ (disarankan), ElastiCache memungkinkan Anda untuk memilih AZ untuk setiap node, atau memungkinkan untuk memilihnya ElastiCache untuk Anda.

Dengan menempatkan simpul di AZ yang berbeda, Anda menghilangkan kemungkinan bahwa kegagalan, seperti pemadaman listrik, di satu AZ akan menyebabkan seluruh sistem Anda gagal. Pengujian telah menunjukkan bahwa tidak ada perbedaan latensi yang signifikan antara menempatkan semua simpul di satu AZ atau menyebarkannya di beberapa AZ.

Anda dapat menentukan AZ untuk setiap simpul saat Anda membuat klaster atau dengan menambahkan simpul saat Anda mengubah klaster yang telah ada. Untuk informasi selengkapnya, lihat informasi berikut:

- Membuat klaster
- Memodifikasi sebuah cluster ElastiCache
- Menambahkan simpul ke klaster

## Wilayah & titik akhir yang didukung

Amazon ElastiCache tersedia di beberapa AWS Wilayah. Ini berarti Anda dapat meluncurkan ElastiCache cluster di lokasi yang memenuhi kebutuhan Anda. Misalnya, Anda dapat meluncurkan di AWS Wilayah terdekat dengan pelanggan Anda, atau meluncurkan di AWS Wilayah tertentu untuk memenuhi persyaratan hukum tertentu.

Setiap Wilayah dirancang untuk terisolasi sepenuhnya dari Wilayah lainnya. Di dalam setiap wilayah terdapat beberapa Zona Ketersediaan. ElastiCache Cache tanpa server secara otomatis mereplikasi data di beberapa zona ketersediaan (kecuali`us-west-1`, di mana data direplikasi dalam dua zona ketersediaan) untuk ketersediaan tinggi. Saat mendesain ElastiCache cluster Anda sendiri, Anda dapat memilih untuk meluncurkan node Anda di AZ yang berbeda untuk mencapai toleransi kesalahan. Untuk informasi selengkapnya tentang wilayah dan Zona Ketersediaan, lihat Memilih wilayah dan zona ketersediaan di bagian atas topik ini.

Daerah di ElastiCache mana didukung

| Nama Wilayah/Wilayah | Titik Akhir | Protokol | |
|---|---|---|---|
| Wilayah AS Timur (Ohio)<br>`us-east-2` | `elasticache.us-east-2.amazonaws.com` | HTTPS | |
| Wilayah AS Timur (Virginia Utara)<br>`us-east-1` | `elasticache.us-east-1.amazonaws.com` | HTTPS | |
| Wilayah AS Barat (California Utara)<br>`us-west-1` | `elasticache.us-west-1.amazonaws.com` | HTTPS | |
| Wilayah AS Barat (Oregon)<br>`us-west-2` | `elasticache.us-west-2.amazonaws.com` | HTTPS | |
| Wilayah Kanada (Pusat)<br>`ca-central-1` | `elasticache.ca-central-1.amazonaws.com` | HTTPS | |
| Wilayah Kanada (Barat)<br>`ca-west-1` | `elasticache.ca-west-1.amazonaws.com` | HTTPS | |
| Asia Pasifik (Jakarta)<br>`ap-southeast-3` | `elasticache.ap-southeast-3.amazonaws.com` | HTTPS | |

| Nama Wilayah/Wilayah | Titik Akhir | Protokol | |
|---|---|---|---|
| Wilayah Asia Pasifik (Mumbai)<br>`ap-south-1` | `elasticache.ap-south-1.amazonaws.com` | HTTPS | |
| Wilayah Asia Pasifik (Hyderabad)<br>`ap-south-2` | `elasticache.ap-south-2.amazonaws.com` | HTTPS | |
| Wilayah Asia Pasifik (Tokyo)<br>`ap-northeast-1` | `elasticache.ap-northeast-1.amazonaws.com` | HTTPS | |
| Wilayah Asia Pasifik (Seoul)<br>`ap-northeast-2` | `elasticache.ap-northeast-2.amazonaws.com` | HTTPS | |
| Wilayah Asia Pasifik (Osaka)<br>`ap-northeast-3` | `elasticache.ap-northeast-3.amazonaws.com` | HTTPS | |
| Wilayah Asia Pasifik (Singapura)<br>`ap-southeast-1` | `elasticache.ap-southeast-1.amazonaws.com` | HTTPS | |
| Wilayah Asia Pasifik (Sydney)<br>`ap-southeast-2` | `elasticache.ap-southeast-2.amazonaws.com` | HTTPS | |

| Nama Wilayah/Wilayah | Titik Akhir | Protokol | |
|---|---|---|---|
| Wilayah Eropa (Frankfurt)<br>`eu-central-1` | `elasticache.eu-central-1.amazonaws.com` | HTTPS | |
| Wilayah Eropa (Zürich)<br>`eu-central-2` | `elasticache.eu-central-2.amazonaws.com` | HTTPS | |
| Wilayah Eropa (Stockholm)<br>`eu-north-1` | `elasticache.eu-north-1.amazonaws.com` | HTTPS | |
| Wilayah Timur Tengah (Bahrain)<br>`me-south-1` | `elasticache.me-south-1.amazonaws.com` | HTTPS | |
| Wilayah Timur Tengah (UEA)<br>`me-central-1` | `elasticache.me-central-1.amazonaws.com` | HTTPS | |
| Wilayah Eropa (Irlandia)<br>`eu-west-1` | `elasticache.eu-west-1.amazonaws.com` | HTTPS | |
| Wilayah Eropa (London)<br>`eu-west-2` | `elasticache.eu-west-2.amazonaws.com` | HTTPS | |

| Nama Wilayah/Wilayah | Titik Akhir | Protokol | |
|---|---|---|---|
| Wilayah Eropa (Paris)<br>`eu-west-3` | `elasticache.eu-west-3.amazonaws.com` | HTTPS | |
| Wilayah Eropa (Milan)<br>`eu-south-1` | `elasticache.eu-south-1.amazonaws.com` | HTTPS | |
| Wilayah Eropa (Spanyol)<br>`eu-south-2` | `elasticache.eu-south-2.amazonaws.com` | HTTPS | |
| Wilayah Amerika Selatan (Sao Paulo)<br>`sa-east-1` | `elasticache.sa-east-1.amazonaws.com` | HTTPS | |
| Wilayah Tiongkok (Beijing)<br>`cn-north-1` | `elasticache.cn-north-1.amazonaws.com.cn` | HTTPS | |
| Wilayah Tiongkok (Ningxia)<br>`cn-northwest-1` | `elasticache.cn-northwest-1.amazonaws.com.cn` | HTTPS | |
| Wilayah Asia Pacific (Hong Kong)<br>`ap-east-1` | `elasticache.ap-east-1.amazonaws.com` | HTTPS | |
| Wilayah Afrika (Cape Town)<br>`af-south-1` | `elasticache.af-south-1.amazonaws.com` | HTTPS | |

| Nama Wilayah/Wilayah | Titik Akhir | Protokol | |
|---|---|---|---|
| Wilayah Israel (Tel Aviv)<br><br>`il-central-1` | `elasticache.il-central-1.amazonaws.com` | HTTPS | |
| AWS GovCloud (AS-Barat)<br><br>`us-gov-west-1` | `elasticache.us-gov-west-1.amazonaws.com` | HTTPS | |
| AWS GovCloud (AS-Timur)<br><br>`us-gov-east-1` | `elasticache.us-gov-east-1.amazonaws.com` | HTTPS | |
| Untuk informasi tentang penggunaan AWS GovCloud (AS) dengan ElastiCache, lihat Layanan di wilayah AWS GovCloud (AS): ElastiCache. | | | |

Beberapa wilayah mendukung subset dari jenis simpul. Untuk tabel tipe node yang didukung menurut AWS Region, lihatJenis simpul yang didukung oleh Wilayah AWS.

Untuk tabel AWS produk dan layanan menurut wilayah, lihat Produk dan Layanan menurut Wilayah.

## Menggunakan zona lokal dengan ElastiCache

Zona Lokal adalah perluasan dari Wilayah AWS yang secara geografis dekat dengan pengguna Anda. Anda dapat memperluas cloud privat virtual (VPC) apa pun dari Wilayah AWS induk menjadi Zona Lokal dengan membuat subnet baru dan menetapkannya di Zona Lokal tersebut. Saat Anda membuat subnet di Zona Lokal, VPC Anda diperluas ke Zona Lokal itu. Subnet di Zona Lokal beroperasi sama seperti subnet lain di VPC Anda.

Dengan menggunakan Zona Lokal, Anda dapat menempatkan sumber daya seperti klaster ElastiCache di beberapa lokasi yang dekat dengan pengguna Anda.

Saat Anda membuat klaster ElastiCache, Anda dapat memilih subnet di Zona Lokal. Zona Lokal memiliki koneksinya sendiri ke internet dan mendukung AWS Direct Connect. Oleh karena itu,

sumber daya yang dibuat di Zona Lokal dapat melayani pengguna lokal dengan komunikasi berlatensi sangat rendah. Untuk informasi selengkapnya, lihat Zona Lokal AWS.

Zona Lokal ditunjukkan oleh kode Wilayah AWS yang diikuti oleh pengidentifikasi yang menunjukkan lokasinya, misalnya `us-west-2-lax-1a`.

Saat ini, Zona Lokal yang tersedia adalah `us-west-2-lax-1a` dan `us-west-2-lax-1b`.

Batasan berikut berlaku untuk ElastiCache untuk Zona Lokal:

- Penyimpanan data global tidak didukung.
- Migrasi online tidak didukung.
- Jenis simpul berikut didukung oleh Zona Lokal saat ini:
  - Generasi saat ini:

    Jenis simpul M5: `cache.m5.large`, `cache.m5.xlarge`, `cache.m5.2xlarge`, `cache.m5.4xlarge`, `cache.m5.12xlarge`, `cache.m5.24xlarge`

    Jenis simpul R5: `cache.r5.large`, `cache.r5.xlarge`, `cache.r5.2xlarge`, `cache.r5.4xlarge`, `cache.r5.12xlarge`, `cache.r5.24xlarge`

    Jenis simpul T3: `cache.t3.micro`, `cache.t3.small`, `cache.t3.medium`

## Mengaktifkan zona lokal

1. Aktifkan Zona Lokal di konsol Amazon EC2.

   Untuk informasi selengkapnya, lihat Mengaktifkan Zona Lokal di Panduan Pengguna Amazon EC2.

2. Buat subnet di Zona Lokal.

   Untuk informasi selengkapnya, lihat Membuat subnet di VPC Anda di Panduan Pengguna Amazon VPC.

3. Membuat grup subnet ElastiCache di Zona Lokal.

   Saat Anda membuat grup subnet ElastiCache, pilih grup Zona Ketersediaan untuk Zona Lokal.

   Untuk informasi selengkapnya, lihat Membuat grup subnet di Panduan Pengguna ElastiCache.

4. Buat klaster ElastiCache for Memcached yang menggunakan subnet ElastiCache di Zona Lokal. Untuk informasi selengkapnya, lihat salah satu topik berikut:

   - Membuat klaster Redis (Mode Klaster Dinonaktifkan) (Konsol)
   - Membuat klaster Redis (mode klaster diaktifkan) (Konsol)

## Menggunakan Outposts

AWS Outposts adalah layanan terkelola penuh yang memperluas infrastruktur AWS, layanan, API, dan alat ke lokasi pelanggan. Dengan menyediakan akses lokal ke infrastruktur yang dikelola AWS, AWS Outposts memungkinkan pelanggan untuk membangun dan menjalankan aplikasi secara on-premise menggunakan antarmuka pemrograman yang sama seperti di Wilayah AWS, sekaligus menggunakan sumber daya komputasi dan penyimpanan lokal untuk latensi yang lebih rendah dan kebutuhan pemrosesan data lokal. Outpost adalah sebuah kumpulan komputasi dan kapasitas penyimpanan AWS yang di-deploy di situs pelanggan. AWS mengoperasikan, memantau, dan mengelola kapasitas ini sebagai bagian dari Wilayah AWS. Anda dapat membuat subnet di Outposts Anda dan menentukannya saat Anda membuat sumber daya AWS misalnya klaster ElastiCache.

> **Note**
>
> Dalam versi ini, berlaku batasan berikut:
>
> - ElastiCache untuk Outposts hanya mendukung keluarga simpul M5 dan R5.
> - Migrasi langsung tidak didukung.
> - Multi-AZ (replikasi lintas Outpost tidak didukung).
> - Snapshot lokal tidak didukung.
> - ElastiCache untuk Outposts tidak didukung di wilayah berikut: cn-north-1, cn-northwest-1 dan ap-northeast-3.

### Menggunakan Outposts dengan konsol Redis

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Pada panel navigasi, pilih Redis.
3. Pada Mesin Klaster, pilih Redis.

4. Di bawah Lokasi, pilih On-Premise - Buat instans ElastiCache Anda di AWS Outposts.

Mengonfigurasi pilihan on-premise

Anda dapat memilih Outpost yang tersedia untuk menambahkan klaster cache atau, jika tidak ada Outposts yang tersedia, buat Outpost baru menggunakan langkah berikut:

Di bawah Opsi on-premise:

1. Di bawah Pengaturan Redis:

    a. Nama: Masukkan nama untuk klaster Redis

    b. Deskripsi: Masukkan deskripsi untuk klaster Redis.

    c. Kompatibilitas versi mesin: Versi mesin didasarkan pada wilayah AWS Outpost

    d. Port: Terima port default, 6379. Jika Anda ingin menggunakan port yang berbeda, tuliskan nomor port tersebut.

    e. Grup parameter: Gunakan drop-down untuk memilih grup parameter default atau khusus.

    f. Jenis Simpul: Instans yang tersedia didasarkan pada ketersediaan Outposts. Porting Assistant for .NET untuk Outposts hanya mendukung keluarga simpul M5 dan R5. Dari daftar drop-down, pilih Outposts lalu pilih jenis simpul tersedia yang ingin Anda gunakan untuk klaster ini. Kemudian pilih Simpan.

    g. Jumlah Replika: Masukkan jumlah replika baca yang ingin dibuat untuk grup replikasi ini. Anda harus memiliki minimal satu dan maksimal lima replika baca. Nilai default-nya adalah 2.

       Nama-nama replika baca yang dihasilkan otomatis mengikuti pola yang sama seperti nama klaster primer, dengan tanda hubung dan tiga digit nomor berurutan ditambahkan ke akhir, dimulai dengan `-002`. Misalnya, jika grup replikasi Anda bernama `MyGroup`, maka nama-nama berikutnya adalah `MyGroup-002`, `MyGroup-003`, `MyGroup-004`, `MyGroup-005`, `MyGroup-006`.

2. Di bawah Pengaturan lanjutan Redis:

    a. Grup Subnet: Dari daftar, pilih Buat baru.

    - Nama: Masukkan nama untuk grup subnet
    - Deskripsi: Masukkan deskripsi untuk grup subnet

- ID VPC: ID VPC harus cocok dengan VPC Outpost. Jika Anda memilih VPC yang tidak memiliki ID subnet pada Outposts, daftar yang dikembalikan akan kosong.
- Zona Ketersediaan atau Outpost: Pilih Outpost yang Anda gunakan.
- ID Subnet: Pilih ID subnet yang tersedia untuk Outpost. Jika tidak ada ID subnet yang tersedia, Anda perlu membuatnya. Untuk informasi selengkapnya, silakan lihat Membuat Subnet.

b. Pilih Buat.

Melihat detail klaster Outpost

Pada halaman daftar Redis, pilih sebuah klaster yang dimiliki oleh AWS Outpost dan perhatikan hal-hal berikut ketika melihat Detail klaster:

- Zona Ketersediaan: Ini akan mewakili Outpost, menggunakan ARN (Amazon Resource Name) dan Nomor Sumber Daya AWS.
- Nama Outpost: Nama dari AWS Outpost.

## Menggunakan Outposts dengan AWS CLI

Anda dapat menggunakan AWS Command Line Interface (AWS CLI) untuk mengontrol beberapa layanan AWS dari baris perintah dan mengotomatiskan layanan tersebut melalui skrip. Anda dapat menggunakan AWS CLI untuk operasi ad hoc (satu kali).

Mengunduh dan mengonfigurasi AWS CLI

AWS CLI berjalan di Windows, macOS, atau Linux. Gunakan prosedur berikut untuk mengunduh dan mengonfigurasinya.

Mengunduh, menginstal, dan mengonfigurasi CLI

1. Unduh AWS CLI pada halaman web AWS Command Line Interface.
2. Ikuti petunjuk untuk Menginstal AWS CLI dan Mengonfigurasi AWS CLI pada Panduan Pengguna AWS Command Line Interface.

Menggunakan AWS CLI dengan Outposts

Gunakan operasi CLI berikut untuk membuat klaster cache yang menggunakan Outposts:

- create-cache-cluster – Dengan operasi ini, parameter `outpost-mode` menerima nilai yang menentukan apakah simpul dalam klaster cache dibuat dalam satu Outposts atau beberapa Outposts.

> **Note**
>
> Pada saat ini, hanya mode `single-outpost` yang didukung.

```
aws elasticache create-cache-cluster \
   --cache-cluster-id cache cluster id \
   --outpost-mode single-outpost \
```

# Bekerja dengan ElastiCache

Di bagian ini Anda dapat menemukan detail tentang cara mengelola berbagai komponen ElastiCache implementasi Anda.

Topik



## Melakukan snapshot dan pemulihan

Cache Amazon ElastiCache yang menjalankan Redis dapat mencadangkan datanya dengan membuat snapshot. Anda dapat menggunakan cadangan untuk memulihkan cache atau menyemai data ke cache baru. Cadangan terdiri dari metadata cache, beserta semua data di dalam cache. Semua cadangan ditulis ke Amazon Simple Storage Service (Amazon S3), yang menyediakan penyimpanan tahan lama. Anda dapat memulihkan data kapan saja dengan membuat cache Redis baru dan mengisinya dengan data dari cadangan. Dengan ElastiCache, Anda dapat mengelola cadangan menggunakan AWS Management Console, AWS Command Line Interface, (AWS CLI), dan API ElastiCache.

Jika Anda ingin menghapus cache dan penting untuk menyimpan datanya, Anda dapat mengambil tindakan pencegahan tambahan. Untuk melakukannya, buat cadangan manual terlebih dahulu, pastikan bahwa statusnya tersedia, lalu hapus cache. Dengan melakukannya, Anda dapat memastikan bahwa jika cadangan gagal, Anda masih memiliki data cache tersedia. Anda dapat mencoba lagi membuat cadangan, dengan mengikuti praktik terbaik yang diuraikan sebelumnya.

Topik



## Batasan pencadangan

Pertimbangkan kendala berikut saat merencanakan atau membuat cadangan:

- Untuk saat ini, pencadangan dan pemulihan hanya didukung untuk cache yang berjalan di Redis.
- Untuk klaster Redis (mode klaster dinonaktifkan), pencadangan dan pemulihan tidak didukung di simpul `cache.t1.micro`. Semua jenis simpul cache lain didukung.
- Untuk klaster Redis (mode klaster diaktifkan), pencadangan dan pemulihan didukung untuk semua jenis simpul.
- Dalam jangka waktu 24 jam, Anda dapat membuat maksimal 20 cadangan manual per simpul di klaster.
- Redis (mode klaster diaktifkan) hanya mendukung pencadangan di tingkat klaster (untuk API atau CLI, tingkat grup replikasi). Redis (mode klaster diaktifkan) tidak mendukung pencadangan di tingkat serpihan (untuk API atau CLI, tingkat grup simpul).
- Selama proses pencadangan, Anda tidak dapat menjalankan operasi API atau CLI apa pun lainnya di klaster.
- Jika menggunakan klaster dengan tingkatan data, Anda tidak dapat mengekspor cadangan ke Amazon S3.

- Anda dapat memulihkan cadangan klaster menggunakan jenis simpul r6gd hanya untuk klaster menggunakan jenis simpul r6gd.

## Dampak performa pencadangan klaster yang dirancang sendiri

Pencadangan di cache nirserver transparan untuk aplikasi tanpa dampak performa. Namun, saat membuat cadangan untuk klaster yang dirancang sendiri, mungkin ada beberapa dampak performa bergantung pada memori cadangan yang tersedia.

Berikut ini adalah panduan untuk meningkatkan kinerja performa untuk klaster yang dirancang sendiri.

- Atur parameter `reserved-memory-percent` – Untuk mengurangi paging yang berlebihan, sebaiknya tetapkan parameter reserved-memory-percent. Parameter ini mencegah Redis mengonsumsi semua memori simpul yang tersedia, dan dapat membantu mengurangi jumlah paging. Anda mungkin juga melihat peningkatan performa hanya dengan menggunakan simpul yang lebih besar. Untuk informasi selengkapnya tentang parameter reserved-memory dan reserved-memory-percent, lihat Mengelola Memori Terpesan.

- Membuat cadangan dari replika baca – Jika Anda menjalankan Redis di grup simpul dengan lebih dari satu simpul, Anda dapat melakukan pencadangan dari simpul primer atau salah satu replika baca. Karena sumber daya sistem yang diperlukan selama BGSAVE, sebaiknya buat cadangan dari salah satu replika baca. Pada saat cadang sedang dibuat dari replika, simpul primer tetap tidak terpengaruh oleh kebutuhan sumber daya BGSAVE. Simpul primer dapat terus melayani permintaan tanpa menjadi lambat.

  Untuk melakukannya, lihat Membuat cadangan manual (Konsol) dan di bidang Nama Klaster di jendela Membuat Cadangan, pilih replika, bukan simpul primer default.

Jika Anda menghapus grup replikasi dan meminta cadangan akhir, ElastiCache selalu mengambil cadangan dari simpul primer. Hal ini memastikan bahwa Anda menangkap data Redis yang paling baru, sebelum grup replikasi dihapus.

# Menjadwalkan pencadangan otomatis

Anda dapat mengaktifkan pencadangan otomatis untuk cache ElastiCache for Redis. Saat pencadangan otomatis diaktifkan, ElastiCache membuat cadangan dari cache setiap hari. Tidak ada dampak pada cache dan perubahan terjadi seketika. Pencadangan otomatis dapat membantu mencegah kehilangan data. Jika terjadi kegagalan, Anda dapat membuat cache baru, memulihkan data Anda dari cadangan terakhir. Hasilnya adalah cache dengan proses warm start, yang dimuat di awal dengan data Anda dan siap digunakan. Untuk informasi selengkapnya, lihat Melakukan pemulihan dari cadangan ke dalam cache baru.

Saat Anda menjadwalkan backup otomatis, Anda harus merencanakan pengaturan berikut:

- Waktu mulai pencadangan - Waktu hari ketika ElastiCache mulai membuat cadangan. Anda dapat mengatur periode pencadangan setiap saat pada waktu yang paling sesuai. Jika Anda tidak menentukan periode pencadangan, ElastiCache menetapkannya secara otomatis.

- Batas retensi cadangan — Jumlah hari cadangan dipertahankan di Amazon S3. Contohnya, jika Anda menetapkan batas retensi ke 5, cadangan yang diambil hari ini akan dipertahankan selama 5 hari. Jika batas retensi berakhir, cadangan akan dihapus secara otomatis.

  Batas retensi cadangan maksimum adalah 35 hari. Jika batas retensi cadangan ditetapkan ke 0, cadangan otomatis akan dinonaktifkan untuk cache.

Anda dapat mengaktifkan atau menonaktifkan pencadangan otomatis saat membuat cache baru atau memperbarui cache Redis yang ada, dengan menggunakan konsol ElastiCache, AWS CLI, atau API ElastiCache dengan mencentang kotak Aktifkan Pencadangan Otomatis di bagian Pengaturan Redis Lanjutan.

# Mengambil cadangan manual

Selain cadangan otomatis, Anda dapat membuat cadangan manual kapan saja. Tidak seperti cadangan otomatis, yang secara otomatis dihapus setelah berakhirnya periode penyimpanan yang ditentukan, cadangan manual tidak memiliki periode penyimpanan untuk penghapusan secara otomatis. Bahkan jika Anda menghapus cache, setiap cadangan manual dari cache akan tetap tersimpan. Jika Anda tidak ingin lagi menyimpan cadangan manual, Anda harus menghapusnya sendiri secara eksplisit.

Selain membuat cadangan manual secara langsung, Anda dapat membuat cadangan manual dengan salah satu cara berikut:

- Menyalin cadangan Tidak menjadi masalah apakah backup sumber dibuat secara otomatis atau manual.
- Membuat cadangan akhir Membuat backup dengan segera sebelum menghapus klaster atau simpul.

Anda dapat membuat cadangan manual dari cache menggunakan AWS Management Console,AWS CLI, atau API ElastiCache.

Membuat cadangan manual (Konsol)

Untuk membuat cadangan dari cache (konsol)

1. Masuk ke AWS Management Console lalu buka konsol Amazon EC2 di https://console.aws.amazon.com/ec2/.
2. Dari panel navigasi, pilih Cache Redis.
3. Pilih kotak di sebelah kiri nama klaster Redis yang ingin Anda cadangkan.
4. Pilih Cadangkan.
5. Di dialog Buat Cadangan, ketik nama untuk cadangan Anda di kotak Nama Cadangan. Sebaiknya berikan nama yang menunjukkan klaster yang dicadangakan serta tanggal dan waktu cadangan dibuat.

   Batasan penamaan klaster adalah sebagai berikut:

   - Harus berisi 1–40 karakter alfanumerik atau tanda hubung.
   - Harus dimulai dengan huruf.

- Tidak dapat berisi dua tanda hubung berturut-turut.
- Tidak boleh diakhiri dengan tanda hubung.

6. Pilih Buat Cadangan.

   Status klaster berubah menjadi membuat snapshot.

### Membuat cadangan manual (AWS CLI)

#### Cadangan manual dari cache nirserver dengan AWS CLI

Untuk membuat cadangan manual cache menggunakan AWS CLI, gunakan operasi `create-serverless-snapshot` AWS CLI dengan parameter berikut:

- `--serverless-cache-name` – Nama cache nirserver yang Anda cadangkan.
- `--serverless-cache-snapshot-name` – Nama snapshot yang akan dibuat.

Untuk Linux, macOS, atau Unix:

```
aws elasticache create-serverless-snapshot \
                        --serverless-cache-name CacheName \
                        --serverless-cache-snapshot-name bkup-20231127
```

Untuk Windows:

```
aws elasticache create-serverless-snapshot ^
    --serverless-cache-name CacheName ^
    --serverless-cache-snapshot-name bkup-20231127
```

#### Cadangan manual dari klaster yang dirancang sendiri dengan AWS CLI

Untuk membuat cadangan manual dari klaster yang dirancang sendiri menggunakan AWS CLI, gunakan operasi `create-snapshot` AWS CLI dengan parameter berikut:

- `--cache-cluster-id`
  - Jika klaster yang Anda cadangkan tidak memiliki simpul replika, `--cache-cluster-id` adalah nama klaster yang Anda cadangakan, misalnya *`mycluster`*.

- Jika klaster yang Anda cadangkan memiliki satu atau beberapa simpul replika, `--cache-cluster-id` adalah nama simpul di klaster yang dapat Anda gunakan untuk cadangan. Sebagai contoh, namanya mungkin *mycluster-002*.

  Gunakan parameter ini hanya ketika membuat cadangan sebuah klaster Redis (mode klaster dinonaktifkan).

- `--replication-group-id` – Nama klaster Redis (mode klaster diaktifkan) (CLI/API: grup replikasi) untuk digunakan sebagai sumber untuk cadangan. Gunakan parameter ini hanya ketika mencadangkan klaster Redis (mode klaster diaktifkan).

- `--snapshot-name` – Nama snapshot yang akan dibuat.

  Batasan penamaan klaster adalah sebagai berikut:

  - Harus berisi 1–40 karakter alfanumerik atau tanda hubung.
  - Harus dimulai dengan huruf.
  - Tidak dapat berisi dua tanda hubung berturut-turut.
  - Tidak boleh diakhiri dengan tanda hubung.

Contoh 1: Mencadangkan klaster Redis (Mode Klaster Dinonaktifkan) yang tidak memiliki replika simpul

Operasi AWS CLI berikut membuat backup `bkup-20150515` dari klaster Redis(mode klaster dinonaktifkan) `myNonClusteredRedis` yang tidak memiliki replika baca.

Untuk Linux, macOS, atau Unix:

```
aws elasticache create-snapshot \
    --cache-cluster-id myNonClusteredRedis \
    --snapshot-name bkup-20150515
```

Untuk Windows:

```
aws elasticache create-snapshot ^
    --cache-cluster-id myNonClusteredRedis ^
    --snapshot-name bkup-20150515
```

Contoh 2: Mencadangkan klaster Redis (Mode Klaster Dinonaktifkan) yang tidak memiliki replika simpul

Operasi AWS CLI berikut membuat cadangan `bkup-20150515` dari klaster Redis (mode klaster dinonaktifkan) `myNonClusteredRedis`. Cadangan ini memiliki satu atau beberapa replika baca.

Untuk Linux, macOS, atau Unix:

```
aws elasticache create-snapshot \
    --cache-cluster-id myNonClusteredRedis-001 \
    --snapshot-name bkup-20150515
```

Untuk Windows:

```
aws elasticache create-snapshot ^
    --cache-cluster-id myNonClusteredRedis-001 ^
    --snapshot-name bkup-20150515
```

Contoh Output: Mencadangkan Klaster Redis (Mode Klaster Dinonaktifkan) dengan Simpul Replika

Output dari operasi ini akan terlihat seperti berikut.

```
{
    "Snapshot": {
        "Engine": "redis",
        "CacheParameterGroupName": "default.redis6.x",
        "VpcId": "vpc-91280df6",
        "CacheClusterId": "myNonClusteredRedis-001",
        "SnapshotRetentionLimit": 0,
        "NumCacheNodes": 1,
        "SnapshotName": "bkup-20150515",
        "CacheClusterCreateTime": "2017-01-12T18:59:48.048Z",
        "AutoMinorVersionUpgrade": true,
        "PreferredAvailabilityZone": "us-east-1c",
        "SnapshotStatus": "creating",
        "SnapshotSource": "manual",
        "SnapshotWindow": "08:30-09:30",
        "EngineVersion": "6.0",
        "NodeSnapshots": [
            {
                "CacheSize": "",
                "CacheNodeId": "0001",
```

```
                "CacheNodeCreateTime": "2017-01-12T18:59:48.048Z"
            }
        ],
        "CacheSubnetGroupName": "default",
        "Port": 6379,
        "PreferredMaintenanceWindow": "wed:07:30-wed:08:30",
        "CacheNodeType": "cache.m3.2xlarge",
        "DataTiering": "disabled"
    }
}
```

Contoh 3: Mencadangkan klaster untuk Redis (Mode Klaster Diaktifkan)

Operasi AWS CLI berikut membuat cadangan `bkup-20150515` dari klaster Redis `myClusteredRedis` (mode klaster diaktifkan). Perhatikan penggunaan `--replication-group-id` alih-alih `--cache-cluster-id` untuk mengidentifikasi sumber.

Untuk Linux, macOS, atau Unix:

```
aws elasticache create-snapshot \
    --replication-group-id myClusteredRedis \
    --snapshot-name bkup-20150515
```

Untuk Windows:

```
aws elasticache create-snapshot ^
    --replication-group-id myClusteredRedis ^
    --snapshot-name bkup-20150515
```

Contoh Output: Mencadangkan Klaster Redis (Mode Klaster Diaktifkan)

Output dari operasi ini akan terlihat seperti berikut.

```
{
    "Snapshot": {
        "Engine": "redis",
        "CacheParameterGroupName": "default.redis6.x.cluster.on",
        "VpcId": "vpc-91280df6",
        "NodeSnapshots": [
            {
                "CacheSize": "",
                "NodeGroupId": "0001"
```

```
            },
            {
                "CacheSize": "",
                "NodeGroupId": "0002"
            }
        ],
        "NumNodeGroups": 2,
        "SnapshotName": "bkup-20150515",
        "ReplicationGroupId": "myClusteredRedis",
        "AutoMinorVersionUpgrade": true,
        "SnapshotRetentionLimit": 1,
        "AutomaticFailover": "enabled",
        "SnapshotStatus": "creating",
        "SnapshotSource": "manual",
        "SnapshotWindow": "10:00-11:00",
        "EngineVersion": "6.0",
        "CacheSubnetGroupName": "default",
        "ReplicationGroupDescription": "2 shards 2 nodes each",
        "Port": 6379,
        "PreferredMaintenanceWindow": "sat:03:30-sat:04:30",
        "CacheNodeType": "cache.r3.large",
        "DataTiering": "disabled"
    }
}
```

Topik terkait

Untuk informasi lebih lanjut, lihat create-snapshot dalam Referensi Perintah AWS CLI.

# Membuat cadangan akhir

Anda dapat membuat cadangan akhir menggunakan konsol ElastiCache, AWS CLI, atau API ElastiCache.

### Membuat cadangan akhir (konsol)

Anda dapat membuat cadangan akhir saat menghapus cache Redis menggunakan konsol ElastiCache.

Untuk membuat cadangan akhir saat menghapus cache Redis, pada kotak dialog hapus pilih Ya di bagian Buat cadangan dan beri nama cadangan.

Topik terkait

- Menggunakan AWS Management Console
- Menghapus Grup Replikasi (Konsol)

### Membuat cadangan akhir (AWS CLI)

Anda dapat membuat cadangan akhir saat menghapus cache Redis menggunakan AWS CLI.

Topik

- Saat menghapus cache nirserver Redis
- Ketika menghapus klaster Redis yang tidak memiliki replika baca
- Saat menghapus klaster Redis dengan replika baca

#### Saat menghapus cache nirserver Redis

Untuk membuat cadangan akhir, gunakan operasi `delete-serverless-cache` AWS CLI dengan parameter berikut.

- `--serverless-cache-name` – Nama cache yang dihapus.
- `--final-snapshot-name` – Nama cadangan.

Kode berikut membuat cadangan akhir `bkup-20231127-final` saat menghapus cache `myserverlesscache`.

Untuk Linux, macOS, atau Unix:

```
aws elasticache delete-serverless-cache \
        --serverless-cache-name myserverlesscache \
        --final-snapshot-name bkup-20231127-final
```

Untuk Windows:

```
aws elasticache delete-serverless-cache ^
        --serverless-cache-name myserverlesscache ^
        --final-snapshot-name bkup-20231127-final
```

Untuk informasi selengkapnya, lihat delete-serverless-cache di Referensi Perintah AWS CLI.

Ketika menghapus klaster Redis yang tidak memiliki replika baca

Untuk membuat cadangan akhir, gunakan operasi `delete-cache-cluster` AWS CLI dengan parameter berikut.

- `--cache-cluster-id` – Nama klaster yang dihapus.
- `--final-snapshot-identifier` – Nama cadangan.

Kode berikut membuat backup cadangan `bkup-20150515-final` saat menghapus klaster `myRedisCluster`.

Untuk Linux, macOS, atau Unix:

```
aws elasticache delete-cache-cluster \
        --cache-cluster-id myRedisCluster \
        --final-snapshot-identifier bkup-20150515-final
```

Untuk Windows:

```
aws elasticache delete-cache-cluster ^
        --cache-cluster-id myRedisCluster ^
        --final-snapshot-identifier bkup-20150515-final
```

Untuk informasi selengkapnya, lihat delete-cache-cluster di Referensi Perintah AWS CLI.

Saat menghapus klaster Redis dengan replika baca

Untuk membuat cadangan akhir saat menghapus grup replika, gunakan operasi `delete-replication-group` AWS CLI dengan parameter berikut:

- `--replication-group-id` – Nama grup replikasi yang dihapus.
- `--final-snapshot-identifier` – Nama cadangan akhir.

Kode berikut mengambil cadangan akhir `bkup-20150515-final` saat menghapus grup replikasi `myReplGroup`.

Untuk Linux, macOS, atau Unix:

```
aws elasticache delete-replication-group \
        --replication-group-id myReplGroup \
        --final-snapshot-identifier bkup-20150515-final
```

Untuk Windows:

```
aws elasticache delete-replication-group ^
        --replication-group-id myReplGroup ^
        --final-snapshot-identifier bkup-20150515-final
```

Untuk informasi selengkapnya, lihat describe-replication-groups dalam Referensi Perintah AWS CLI.

# Menjelaskan cadangan

Prosedur berikut menunjukkan cara menampilkan daftar cadangan Anda. Jika ingin, Anda juga dapat melihat detail cadangan tertentu.

### Menjelaskan Cadangan (Konsol)

Untuk menampilkan cadangan menggunakan AWS Management Console

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Dari panel navigasi, pilih Cadangan.
3. Untuk melihat detail cadangan tertentu, pilih kotak di sebelah kiri nama cadangan.

### Menjelaskan cadangan nirserver (AWS CLI)

Untuk menampilkan daftar cadangan nirserver dan secara opsional detail tentang cadangan tertentu, gunakan operasi CLI `describe-serverless-cache-snapshots`.

#### Contoh

Operasi berikut menggunakan parameter `--max-records` untuk menampilkan hingga 20 cadangan yang terkait dengan akun Anda. Menghilangkan parameter `--max-records` akan menampilkan hingga 50 cadangan.

```
aws elasticache describe-serverless-cache-snapshots --max-records 20
```

Operasi berikut menggunakan parameter `--serverless-cache-name` untuk menampilkan hanya cadangan yang terkait dengan cache `my-cache`.

```
aws elasticache describe-serverless-cache-snapshots --serverless-cache-name my-cache
```

Operasi berikut menggunakan parameter `--serverless-cache-snapshot-name` untuk menampilkan detail cadangan `my-backup`.

```
aws elasticache describe-serverless-cache-snapshots --serverless-cache-snapshot-
name my-backup
```

Untuk informasi selengkapnya, lihat describe-serverless-cache-snapshots di Referensi Perintah AWS CLI.

Menjelaskan cadangan klaster yang dirancang sendiri (AWS CLI)

Untuk menampilkan daftar cadangan klaster yang dirancang sendiri dan secara opsional detail tentang cadangan tertentu, gunakan operasi CLI `describe-snapshots`.

Contoh

Operasi berikut menggunakan parameter `--max-records` untuk menampilkan hingga 20 cadangan yang terkait dengan akun Anda. Menghilangkan parameter `--max-records` akan menampilkan hingga 50 cadangan.

```
aws elasticache describe-snapshots --max-records 20
```

Operasi berikut menggunakan parameter `--cache-cluster-id` untuk menampilkan hanya cadangan yang terkait dengan klaster `my-cluster`.

```
aws elasticache describe-snapshots --cache-cluster-id my-cluster
```

Operasi berikut menggunakan parameter `--snapshot-name` untuk menampilkan detail cadangan `my-backup`.

```
aws elasticache describe-snapshots --snapshot-name my-backup
```

Untuk informasi selengkapnya, lihat describe-snapshots di Referensi Perintah AWS CLI.

# Menyalin cadangan

Anda dapat membuat salinan cadangan apa pun, baik dibuat secara otomatis maupun manual. Anda juga dapat mengekspor cadangan agar Anda dapat mengaksesnya dari luar ElastiCache. Untuk panduan tentang cara mengekspor cadangan, lihat Mengekspor cadangan.

Langkah-langkah berikut menunjukkan cara menyalin cadangan.

Menyalin cadangan (Konsol)

Untuk menyalin cadangan (konsol)

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Untuk melihat daftar cadangan Anda, dari panel navigasi sebelah kiri, pilih Cadangan.
3. Dari daftar cadangan, pilih kotak di sebelah kiri nama cadangan yang ingin Anda salin.
4. Pilih Tindakan lalu Salin.
5. Pada kotak Nama cadangan baru, ketikkan nama untuk cadangan baru Anda.
6. Pilih Salin.

Menyalin cadangan nirserver (AWS CLI)

Untuk menyalin cadangan cache nirserver, gunakan operasi `copy-serverless-cache-snapshot`.

Parameter

- `--source-serverless-cache-snapshot-name` – Nama cadangan yang akan disalin.
- `--target-serverless-cache-snapshot-name` – Nama salinan cadangan.

Contoh berikut membuat salinan dari cadangan otomatis.

Untuk Linux, macOS, atau Unix:

```
aws elasticache copy-serverless-cache-snapshot \
    --source-serverless-cache-snapshot-name automatic.my-cache-2023-11-27-03-15 \
    --target-serverless-cache-snapshot-name my-backup-copy
```

Untuk Windows:

```
aws elasticache copy-serverless-cache-snapshot ^
    --source-serverless-cache-snapshot-name automatic.my-cache-2023-11-27-03-15 ^
    --target-serverless-cache-snapshot-name my-backup-copy
```

Untuk informasi selengkapnya, lihat copy-serverless-cache-snapshot di AWS CLI.

Menyalin cadangan klaster yang dirancang sendiri (AWS CLI)

Untuk menyalin cadangan klaster yang dirancang sendiri, gunakan operasi `copy-snapshot`.

Parameter

- `--source-snapshot-name` – Nama cadangan yang akan disalin.
- `--target-snapshot-name` – Nama salinan cadangan.
- `--target-bucket` – Dicadangkan untuk mengekspor cadangan. Jangan menggunakan parameter ini saat membuat salinan cadangan. Untuk informasi selengkapnya, lihat Mengekspor cadangan.

Contoh berikut membuat salinan dari cadangan otomatis.

Untuk Linux, macOS, atau Unix:

```
aws elasticache copy-snapshot  \
    --source-snapshot-name automatic.my-redis-primary-2014-03-27-03-15 \
    --target-snapshot-name my-backup-copy
```

Untuk Windows:

```
aws elasticache copy-snapshot ^
    --source-snapshot-name automatic.my-redis-primary-2014-03-27-03-15 ^
    --target-snapshot-name my-backup-copy
```

Untuk informasi selengkapnya, lihat copy-snapshot di AWS CLI.

# Mengekspor cadangan

Amazon ElastiCache mendukung ekspor ElastiCache cadangan Anda ke bucket Amazon Simple Storage Service (Amazon S3), yang memberi Anda akses ke sana dari luar. ElastiCache Anda dapat mengekspor cadangan menggunakan ElastiCache konsol, file AWS CLI, atau ElastiCache API.

Mengekspor cadangan dapat membantu jika Anda perlu meluncurkan cluster di AWS Wilayah lain. Anda dapat mengekspor data Anda dalam satu AWS Wilayah, menyalin file.rdb ke AWS Wilayah baru, dan kemudian menggunakan file.rdb itu untuk menyemai cluster baru alih-alih menunggu klaster baru diisi melalui penggunaan. Untuk informasi tentang menyemai klaster baru, lihat Melakukan seeding klaster yang dirancang sendiri dengan cadangan yang dibuat secara eksternal. Alasan lain Anda ingin mengekspor data dari klaster Anda adalah untuk menggunakan file .rdb untuk pemrosesan offline.

> **Important**
>
> - ElastiCache Cadangan dan ember Amazon S3 yang ingin Anda salin harus berada di Wilayah yang sama AWS .
>
>  Meskipun cadangan disalin ke bucket Amazon S3 dalam keadaan terenkripsi, sebaiknya jangan memberi orang lain akses ke bucket Amazon S3 tempat Anda ingin menyimpan cadangan Anda.
> - Mengekspor cadangan ke Amazon S3 tidak didukung untuk klaster yang menggunakan tingkatan data. Untuk informasi selengkapnya, lihat Tingkatan data.

Sebelum Anda dapat mengekspor cadangan ke bucket Amazon S3, Anda harus memiliki bucket Amazon S3 di Wilayah AWS yang sama dengan cadangan. Berikan ElastiCache akses ke ember. Dua langkah pertama menunjukkan cara melakukannya.

## Langkah 1: Buat bucket Amazon S3.

Langkah-langkah berikut menggunakan konsol Amazon S3 untuk membuat bucket Amazon S3 tempat Anda mengekspor dan menyimpan cadangan. ElastiCache

Untuk membuat bucket Amazon S3

1. Masuk ke AWS Management Console dan buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.

2. Pilih Buat Bucket.
3. Di Buat Bucket - Pilih Nama Bucket dan Wilayah, lakukan hal berikut:

    a. Di Nama Bucket, ketikkan nama untuk bucket Amazon S3 Anda.

    Nama bucket Amazon S3 Anda harus sesuai dengan DNS. Jika tidak, tidak ElastiCache dapat mengakses file cadangan Anda. Aturan untuk kepatuhan DNS adalah:

    - Nama harus minimal 3 dan tidak lebih dari 63 karakter.
    - Nama harus serangkaian satu atau beberapa label yang dipisahkan oleh titik (.) dengan setiap label:
        - Dimulai dengan huruf kecil atau angka.
        - Diakhiri dengan huruf kecil atau angka.
        - Hanya berisi huruf kecil, dan tanda hubung.
    - Nama tidak dapat diformat sebagai alamat IP (misalnya, 192.0.2.0).

    b. Dari daftar Wilayah, pilih AWS Wilayah untuk bucket Amazon S3 Anda. AWS Wilayah ini harus AWS Wilayah yang sama dengan ElastiCache cadangan yang ingin Anda ekspor.

    c. Pilih Buat.

Untuk informasi selengkapnya tentang cara membuat bucket Amazon S3, lihat Membuat bucket di Panduan Pengguna Amazon Simple Storage Service.

## Langkah 2: Berikan ElastiCache akses ke bucket Amazon S3 Anda

ElastiCache Agar dapat menyalin snapshot ke bucket Amazon S3, Anda harus memperbarui kebijakan bucket untuk ElastiCache memberikan akses ke bucket.

> **Warning**
>
> Meskipun cadangan yang disalin ke bucket Amazon S3 sudah terenkripsi, data Anda dapat diakses oleh siapa saja dengan akses ke bucket Amazon S3 Anda. Oleh karena itu, sebaiknya siapkan kebijakan IAM untuk mencegah akses tidak sah ke bucket Amazon S3 ini. Untuk informasi selengkapnya, lihat Mengelola akses di Panduan Pengguna Amazon S3.

Untuk membuat izin yang tepat di bucket Amazon S3, lakukan langkah-langkah yang dijelaskan sebagai berikut.

Untuk memberikan ElastiCache akses ke bucket S3

1. Masuk ke AWS Management Console dan buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Pilih nama bucket Amazon S3 yang menjadi tujuan penyalinan cadangan. Bucket tersebut harus berupa bucket S3 yang Anda buat di Langkah 1: Buat bucket Amazon S3..
3. Pilih tab Izin dan di bagian Izin, pilih Daftar kontrol akses (ACL), lalu pilih Edit.
4. Tambahkan ID Canonical penerima `540804c33a284a299d2547575ce1010f2312ef3da9b3a053c8bc45bf233e4353` dengan opsi berikut:

   - Objek: Daftar, Tulis
   - Bucket ACL: Baca, Tulis

   > **Note**
   > - Untuk GovCloud Wilayah PDT, Id Kanonik adalah. `40fa568277ad703bd160f66ae4f83fc9dfdfd06c2f1b5060ca22442ac3ef8be6`
   > - Untuk GovCloud Wilayah OSU, Id Kanonik adalah. `c54286759d2a83da9c480405349819c993557275cf37d820d514b42da6893f5c`

5. Pilih Simpan.

## Langkah 3: Ekspor ElastiCache cadangan

Sekarang Anda telah membuat bucket S3 dan memberikan ElastiCache izin untuk mengaksesnya. Selanjutnya, Anda dapat menggunakan ElastiCache konsol, AWS CLI, atau ElastiCache API untuk mengekspor snapshot Anda ke sana. Contoh berikut mengasumsikan bahwa identitas IAM pemanggil memiliki izin IAM khusus S3 tambahan berikut.

```
{
 "Version": "2012-10-17",
 "Statement": [{
  "Effect": "Allow",
  "Action": [
   "s3:GetBucketLocation",
   "s3:ListAllMyBuckets",
   "s3:PutObject",
```

```
   "s3:GetObject",
   "s3:DeleteObject",
   "s3:ListBucket"
  ],
  "Resource": "arn:aws:s3:::*"
 }]
}
```

Untuk Wilayah keikutsertaan, berikut adalah contoh tampilan kebijakan yang sudah diperbarui untuk bucket S3. (Contoh ini menggunakan Wilayah Asia Pasifik (Hong Kong).

```
{
    "Version": "2012-10-17",
    "Id": "Policy15397346",
    "Statement": [
        {
            "Sid": "Stmt15399483",
            "Effect": "Allow",
            "Principal": {
                "Service": "elasticache.amazonaws.com"
            },
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::hkg-elasticache-backup",
                "arn:aws:s3:::hkg-elasticache-backup/*"
            ]
        },
        {
            "Sid": "Stmt15399484",
            "Effect": "Allow",

            "Principal": {
                "Service": "ap-east-1.elasticache-snapshot.amazonaws.com"
            },
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::hkg-elasticache-backup",
                "arn:aws:s3:::hkg-elasticache-backup/*"
            ]
        }
    ]
}
```

## Mengekspor ElastiCache cadangan (Konsol)

Langkah-langkah berikut menggunakan ElastiCache konsol untuk mengekspor cadangan ke bucket Amazon S3 sehingga Anda dapat mengaksesnya dari luar. ElastiCache Bucket Amazon S3 harus berada di AWS Wilayah yang sama dengan cadangan. ElastiCache

Untuk mengekspor ElastiCache cadangan ke bucket Amazon S3

1. Masuk ke AWS Management Console dan buka ElastiCache konsol di https://console.aws.amazon.com/elasticache/.
2. Untuk melihat daftar cadangan Anda, dari panel navigasi sebelah kiri, pilih Cadangan.
3. Dari daftar cadangan, pilih kotak di sebelah kiri nama cadangan yang ingin Anda ekspor.
4. Pilih Salin.
5. Di Buat Salinan Backup?, lakukan hal berikut:

   a. Di kotak Nama cadangan baru, ketikkan nama untuk cadangan baru Anda.

      Nama itu harus di antara 1 dan 1.000 karakter serta dapat dienkode ke UTF-8.

      ElastiCache menambahkan pengidentifikasi instance dan nilai `.rdb` yang Anda masukkan di sini. Misalnya, jika Anda memasukkan `my-exported-backup`, ElastiCache membuat `my-exported-backup-0001.rdb`.

   b. Dari daftar Lokasi S3 Target, pilih nama bucket Amazon S3 yang menjadi tujuan Anda menyalin cadangan (bucket yang telah Anda buat di Langkah 1: Buat bucket Amazon S3.).

      Lokasi Target S3 harus berupa bucket Amazon S3 di Wilayah AWS cadangan dengan izin berikut agar proses ekspor berhasil.

      - Akses objek – Baca dan Tulis.
      - Akses izin – Baca.

      Untuk informasi selengkapnya, lihat Langkah 2: Berikan ElastiCache akses ke bucket Amazon S3 Anda.

   c. Pilih Salin.

> **Note**
>
> Jika bucket S3 Anda tidak memiliki izin yang diperlukan untuk mengekspor cadangan ElastiCache ke sana, Anda menerima salah satu pesan galat berikut. Kembali ke Langkah 2: Berikan ElastiCache akses ke bucket Amazon S3 Anda untuk menambahkan izin yang ditentukan dan mencoba mengekspor cadangan Anda kembali.
>
> - ElastiCache belum diberikan izin BACA %s pada Bucket S3.
>
>   Solusi: Tambahkan izin Baca pada bucket.
> - ElastiCache belum diberikan izin WRITE %s pada Bucket S3.
>
>   Solusi: Tambahkan izin Tulis pada bucket.
> - ElastiCache belum diberikan izin READ_ACP %s pada Bucket S3.
>
>   Solusi: Tambahkan Baca untuk akses Izin pada bucket.

Jika Anda ingin menyalin cadangan Anda ke AWS Wilayah lain, gunakan Amazon S3 untuk menyalinnya. Untuk informasi selengkapnya, lihat Mengunduh objek di Panduan Pengguna Amazon Simple Storage.

Mengekspor cadangan ElastiCache tanpa server ()AWS CLI

Mengekspor cadangan cache nirserver

Ekspor backup ke bucket Amazon S3 menggunakan operasi CLI `export-serverless-cache-snapshot` dengan parameter berikut:

Parameter

- `--serverless-cache-snapshot-name` – Nama cadangan yang akan disalin.
- `--s3-bucket-name` – Nama bucket Amazon S3 tempat tujuan Anda mengekspor cadangan. Salinan cadangan dibuat dalam bucket yang ditentukan.

  `--s3-bucket-name`Harus berupa bucket Amazon S3 di AWS Wilayah cadangan dengan izin berikut agar proses ekspor berhasil.
  - Akses objek – Baca dan Tulis.
  - Akses izin – Baca.

Operasi berikut menyalin cadangan ke my-s3-bucket.

Untuk Linux, macOS, atau Unix:

```
aws elasticache export-serverless-cache-snapshot \
    --serverless-cache-snapshot-name automatic.my-redis-2023-11-27 \
    --s3-bucket-name my-s3-bucket
```

Untuk Windows:

```
aws elasticache export-serverless-cache-snapshot ^
    --serverless-cache-snapshot-name automatic.my-redis-2023-11-27 ^
    --s3-bucket-name my-s3-bucket
```

Mengekspor cadangan ElastiCache cluster yang dirancang sendiri ()AWS CLI

Mengekspor cadangan klaster yang dirancang sendiri

Ekspor backup ke bucket Amazon S3 menggunakan operasi CLI `copy-snapshot` dengan parameter berikut:

Parameter

- `--source-snapshot-name` – Nama cadangan yang akan disalin.
- `--target-snapshot-name` – Nama salinan cadangan.

  Nama tersebut harus di antara 1 dan 1.000 karakter serta berenkode UTF-8.

  ElastiCache menambahkan pengidentifikasi instance dan nilai `.rdb` yang Anda masukkan di sini. Misalnya, jika Anda memasukkan `my-exported-backup`, ElastiCache membuat `my-exported-backup-0001.rdb`.
- `--target-bucket` – Nama bucket Amazon S3 tempat tujuan Anda mengekspor cadangan. Salinan cadangan dibuat dalam bucket yang ditentukan.

  `--target-bucket`Harus berupa bucket Amazon S3 di AWS Wilayah cadangan dengan izin berikut agar proses ekspor berhasil.

  - Akses objek – Baca dan Tulis.
  - Akses izin – Baca.

Untuk informasi selengkapnya, lihat Langkah 2: Berikan ElastiCache akses ke bucket Amazon S3 Anda.

Operasi berikut menyalin cadangan ke my-s3-bucket.

Untuk Linux, macOS, atau Unix:

```
aws elasticache copy-snapshot \
    --source-snapshot-name automatic.my-redis-primary-2016-06-27-03-15 \
    --target-snapshot-name my-exported-backup \
    --target-bucket my-s3-bucket
```

Untuk Windows:

```
aws elasticache copy-snapshot ^
    --source-snapshot-name automatic.my-redis-primary-2016-06-27-03-15 ^
    --target-snapshot-name my-exported-backup ^
    --target-bucket my-s3-bucket
```

# Melakukan pemulihan dari cadangan ke dalam cache baru

Anda dapat memulihkan cadangan yang ada ke cache Nirserver baru atau klaster yang dirancang sendiri.

Memulihkan cadangan ke dalam cache nirserver (Konsol)

> **Note**
>
> ElastiCache Serverless mendukung file RDB yang kompatibel dengan Redis versi antara 5.0 dan versi terbaru yang tersedia.

Untuk memulihkan cadangan ke cache nirserver (konsol)

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Dari panel navigasi, pilih Cadangan.
3. Dalam daftar cadangan, pilih kotak di sebelah kiri nama cadangan yang ingin Anda pulihkan.
4. Pilih Tindakan, lalu pilih Pulihkan.
5. Masukkan nama untuk cache nirserver baru dan deskripsi opsional.
6. Klik Buat untuk membuat cache baru dan mengimpor data dari cadangan Anda.

Memulihkan cadangan ke dalam klaster yang dirancang sendiri (Konsol)

Untuk memulihkan cadangan ke klaster yang dirancang sendiri (konsol)

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Dari panel navigasi, pilih Cadangan.
3. Di daftar cadangan, pilih kotak di sebelah kiri nama cadangan yang ingin Anda pulihkan.
4. Pilih Tindakan, lalu pilih Pulihkan.
5. Pilih Rancang cache sendiri dan sesuaikan pengaturan klaster, seperti jenis simpul, ukuran, jumlah serpihan, replika, penempatan AZ, dan pengaturan keamanan.
6. Pilih Buat untuk membuat cache baru dan mengimpor data dari cadangan Anda.

Memulihkan cadangan ke dalam cache nirserver (AWS CLI)

> **Note**
>
> ElastiCache Serverless mendukung file RDB yang kompatibel dengan Redis versi antara 5.0 dan versi terbaru yang tersedia.

Untuk memulihkan cadangan ke cache nirserver baru (AWS CLI)

Contoh AWS CLI berikut membuat cache baru menggunakan `create-serverless-cache` dan mengimpor data dari cadangan.

Untuk Linux, macOS, atau Unix:

```
aws elasticache create-serverless-cache \
    --serverless-cache-name CacheName \
    --engine redis
    --snapshot-arns-to-restore Snapshot-ARN
```

Untuk Windows:

```
aws elasticache create-serverless-cache ^
    --serverless-cache-name CacheName ^
    --engine redis ^
    --snapshot-arns-to-restore Snapshot-ARN
```

Memulihkan cadangan ke dalam klaster yang dirancang sendiri (AWS CLI)

Untuk memulihkan cadangan ke klaster yang dirancang sendiri (AWS CLI)

Anda dapat memulihkan cadangan Redis (mode klaster dinonaktifkan) dengan dua cara.

- Anda dapat memulihkan ke klaster Redis (mode klaster dinonaktifkan) simpul tunggal menggunakan operasi AWS CLI create-cache-cluster.
- Atau, Anda dapat memulihkan ke klaster Redis dengan replika baca (grup replikasi). Untuk melakukannya, Anda dapat menggunakan Redis (mode cluster dinonaktifkan) atau Redis (mode cluster diaktifkan) dengan operasi AWS CLI create-replication-group. Dalam hal ini, Anda menyemai pemulihan dengan file .rdb Redis. Untuk informasi lebih lanjut tentang penyemaian klaster baru yang dirancang sendiri, lihat. Melakukan seeding klaster yang dirancang sendiri dengan cadangan yang dibuat secara eksternal

Saat menggunakan operasi `create-cache-cluster` atau `create-replication-group`, pastikan untuk memasukkan parameter `--snapshot-name` atau `--snapshot-arn` untuk menyemai klaster atau grup replikasi baru dengan data dari cadangan.

## Menghapus cadangan

Cadangan otomatis akan dihapus secara otomatis jika batas retensinya berakhir. Jika Anda menghapus klaster, semua cadangan otomatis juga dihapus. Jika Anda menghapus grup replikasi, semua cadangan otomatis dari klaster di grup tersebut juga dihapus.

ElastiCache menyediakan operasi API penghapusan yang memungkinkan Anda menghapus cadangan kapan saja, terlepas dari apakah cadangan dibuat secara otomatis atau manual. Karena cadangan manual tidak memiliki batas penyimpanan, penghapusan manual adalah satu-satunya cara untuk menghapusnya.

Anda dapat menghapus cadangan menggunakan konsol ElastiCache, AWS CLI, atau ElastiCache.

Menghapus cadangan (Konsol)

Prosedur berikut menghapus cadangan menggunakan konsol ElastiCache.

Untuk menghapus cadangan

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Di panel navigasi, pilih Cadangan.

   Layar Cadangan muncul dengan daftar cadangan Anda.
3. Pilih kotak di sebelah kiri nama cadangan yang ingin Anda hapus.
4. Pilih Hapus.
5. Jika Anda ingin menghapus cadangan ini, pilih Hapus di layar konfirmasi Hapus Cadangan. Status berubah menjadi menghapus.

Menghapus cadangan nirserver (AWS CLI)

Gunakan operasi AWS CLI delete-snapshot dengan parameter berikut untuk menghapus cadangan nirserver.

- `--serverless-cache-snapshot-name` – Nama cadangan yang akan dihapus.

Kode berikut menghapus cadangan `myBackup`.

```
aws elasticache delete-serverless-cache-snapshot --serverless-cache-snapshot-
name myBackup
```

Untuk informasi selengkapnya, lihat delete-serverless-cache-snapshot dalam Referensi Perintah AWS CLI.

Menghapus cadangan klaster yang dirancang sendiri (AWS CLI)

Gunakan operasi delete-snapshot AWS CLI dengan parameter berikut untuk menghapus cadangan klaster yang dirancang sendiri.

- `--snapshot-name` – Nama cadangan yang akan dihapus.

Kode berikut menghapus cadangan `myBackup`.

```
aws elasticache delete-snapshot --snapshot-name myBackup
```

Untuk informasi selengkapnya, lihat delete-snapshot di Referensi Perintah AWS CLI.

## Menandai cadangan

Anda dapat memberikan metadata Anda sendiri ke setiap cadangan dalam bentuk tanda. Tanda memungkinkan Anda mengategorikan cadangan dengan berbagai cara, misalnya, berdasarkan tujuan, pemilik, atau lingkungan. Hal ini berguna ketika Anda memiliki banyak sumber daya dengan tipe yang sama—Anda dapat dengan cepat mengidentifikasi sumber daya tertentu berdasarkan tag yang telah Anda tetapkan. Untuk informasi selengkapnya, lihat Sumber daya yang dapat Anda tandai.

Tanda alokasi biaya adalah sarana untuk melacak biaya Anda di berbagai layanan AWS dengan mengelompokkan pengeluaran pada faktur berdasarkan nilai tanda. Untuk mempelajari lebih lanjut tentang tanda alokasi biaya, lihat Menggunakan tanda alokasi biaya.

Menggunakan konsol ElastiCache, AWS CLI, atau API ElastiCache, Anda dapat menambahkan, mendaftarkan, memodifikasi, menghapus, atau menyalin tanda alokasi biaya di cadangan Anda. Untuk informasi selengkapnya, lihat Memantau biaya dengan tanda alokasi biaya.

# Melakukan seeding klaster yang dirancang sendiri dengan cadangan yang dibuat secara eksternal

Saat Anda membuat klaster Redis yang dirancang sendiri, Anda dapat melakukan seeding data dari file cadangan .rdb Redis. Seeding klaster berguna jika Anda saat ini mengelola instans Redis di luar ElastiCache dan ingin mengisi klaster ElastiCache for Redis baru yang dirancang sendiri dengan data Redis yang ada.

Untuk melakukan seeding klaster Redis yang dirancang sendiri dari cadangan Redis yang dibuat di Amazon ElastiCache, lihat Melakukan pemulihan dari cadangan ke dalam cache baru.

Saat Anda menggunakan file .rdb Redis untuk melakukan seeding klaster Redis baru yang dirancang sendiri, Anda dapat melakukan hal berikut:

- Lakukan peningkatan dari klaster non-partisi ke klaster Redis yang dirancang sendiri (mode klaster) yang menjalankan Redis versi 3.2.4.
- Tentukan jumlah serpihan (disebut grup simpul di API dan CLI) di klaster yang dirancang sendiri. Jumlah ini dapat berbeda dari jumlah serpihan di klaster yang dirancang sendiri yang digunakan untuk membuat file cadangan.
- Tentukan jenis simpul yang berbeda untuk klaster yang dirancang sendiri — lebih besar atau lebih kecil dari yang digunakan di klaster yang membuat cadangan. Jika Anda menskalakan ke jenis simpul yang lebih kecil, pastikan bahwa jenis simpul baru memiliki memori yang cukup untuk data Anda dan overhead Redis. Untuk informasi selengkapnya, lihat Memastikan bahwa Anda memiliki cukup memori untuk membuat snapshot Redis.
- Distribusikan kunci Anda di dalam slot klaster Redis (mode klaster diaktifkan) baru secara berbeda dengan klaster yang digunakan untuk membuat file cadangan.

> **Note**
>
> Anda tidak dapat melakukan seeding klaster Redis (mode klaster dinonaktifkan) dari file .rdb yang dibuat dari klaster Redis (mode klaster diaktifkan).

**Important**

- Anda harus memastikan bahwa data cadangan Redis Anda tidak melebihi sumber daya dari simpul. Misalnya, Anda tidak dapat mengunggah file .rdb dengan data Redis sebesar 5 GB ke simpul cache.m3.medium yang memiliki memori sebesar 2,9 GB.

  Jika cadangan terlalu besar, klaster yang dihasilkan akan memiliki status `restore-failed`. Jika hal ini terjadi, Anda harus menghapus klaster tersebut dan memulai dari awal.

  Untuk daftar lengkap jenis simpul dan spesifikasi, lihat Parameter khusus jenis simpul Redis serta fitur dan detail produk Amazon ElastiCache.
- Anda dapat mengenkripsi file .rdb Redis dengan enkripsi sisi server Amazon S3 (SSE-S3) saja. Untuk informasi selengkapnya, lihat Melindungi data menggunakan enkripsi sisi server.

Pada bagian berikut, Anda dapat menemukan topik yang memandu Anda melalui migrasi klaster Redis Anda dari luar ElastiCache for Redis ke ElastiCache for Redis.

Bermigrasi ke ElastiCache for Redis

- Langkah 1: Buat cadangan Redis
- Langkah 2: Buat folder dan bucket Amazon S3
- Langkah 3: Unggah backup Anda ke Amazon S3
- Langkah 4: Beri ElastiCache akses baca ke file .rdb

## Langkah 1: Buat cadangan Redis

Untuk membuat cadangan Redis untuk melakukan seeding instans ElastiCache for Redis Anda

1. Hubungkan ke instans Redis yang telah ada.
2. Jalankan baik operasi Redis `BGSAVE` ataupun `SAVE` untuk membuat cadangan. Perhatikan tempat file .rdb Anda berada.

   `BGSAVE` bersifat asinkron dan tidak memblokir klien lain saat melakukan pemrosesan. Untuk informasi selengkapnya, lihat BGSAVE di situs web Redis.

SAVE bersifat sinkron dan memblokir proses lainnya hingga selesai. Untuk informasi selengkapnya, lihat SAVE di situs web Redis.

Untuk informasi tambahan tentang cara membuat backup, lihat Redis persistence di situs web Redis.

## Langkah 2: Buat folder dan bucket Amazon S3

Saat Anda telah membuat file backup, Anda perlu mengunggahnya ke folder di dalam bucket Amazon S3. Untuk melakukannya, Anda harus memiliki bucket Amazon S3 dan folder terlebih dahulu di dalam bucket tersebut. Jika Anda sudah memiliki bucket Amazon S3 dan folder dengan izin yang sesuai, Anda dapat melanjutkan ke Langkah 3: Unggah backup Anda ke Amazon S3.

Untuk membuat bucket Amazon S3

1. Masuk ke AWS Management Console dan buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Ikuti petunjuk untuk membuat bucket Amazon S3 di Membuat bucket dalam Panduan Pengguna Amazon Simple Storage Service.

   Nama bucket Amazon S3 Anda harus sesuai dengan DNS. Jika tidak, ElastiCache tidak dapat mengakses file cadangan Anda. Aturan untuk kepatuhan DNS adalah:

   - Nama harus minimal 3 dan tidak lebih dari 63 karakter.
   - Nama harus serangkaian satu atau beberapa label yang dipisahkan oleh titik (.) dengan setiap label:
     - Dimulai dengan huruf kecil atau angka.
     - Diakhiri dengan huruf kecil atau angka.
     - Hanya berisi huruf kecil, dan tanda hubung.
   - Nama tidak dapat diformat sebagai alamat IP (misalnya, 192.0.2.0).

   Anda harus membuat bucket Amazon S3 Anda di Wilayah AWS yang sama dengan klaster ElastiCache for Redis baru Anda. Pendekatan ini memastikan kecepatan transfer data tertinggi saat ElastiCache membaca file .rdb Anda dari Amazon S3.

> **Note**
>
> Untuk menjaga data Anda seaman mungkin, buat izin di bucket Amazon S3 Anda seketat mungkin. Pada saat yang sama, izin masih perlu memperbolehkan bucket dan isinya untuk digunakan menyemai klaster Redis baru Anda.

Untuk menambahkan folder ke bucket Amazon S3

1. Masuk ke AWS Management Console dan buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Pilih nama bucket untuk mengunggah file .rdb Anda.
3. Pilih Buat folder.
4. Masukkan nama untuk folder baru Anda.
5. Pilih Simpan.

   Catat nama dari bucket dan folder.

## Langkah 3: Unggah backup Anda ke Amazon S3

Sekarang saatnya mengunggah file .rdb yang Anda buat di Langkah 1: Buat cadangan Redis. Anda mengunggahnya ke bucket Amazon S3 dan folder yang Anda buat di Langkah 2: Buat folder dan bucket Amazon S3. Untuk informasi lain mengenai tugas ini, lihat Menambahkan Objek ke Bucket. Di antara langkah 2 dan 3, pilih nama folder yang Anda buat.

Untuk mengunggah file .rdb Anda ke folder Amazon S3

1. Masuk ke AWS Management Console dan buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Pilih nama bucket Amazon S3 yang Anda buat di Langkah 2.
3. Pilih nama folder yang Anda buat di Langkah 2.
4. Pilih Unggah.
5. Pilih Tambah file.
6. Telusuri untuk mencari file atau beberapa file yang ingin diunggah, lalu pilih file atau beberapa file itu. Untuk memilih beberapa file, tahan tombol Ctrl saat memilih setiap nama file.

7. Pilih Buka.
8. Konfirmasikan kebenaran file atau beberapa file yang tercantum dalam kotak dialog Unggah, lalu pilih Unggah.

Perhatikan jalur ke file .rdb Anda. Misalnya, jika nama bucket Anda `myBucket` dan jalurnya adalah `myFolder/redis.rdb`, masukkan `myBucket/myFolder/redis.rdb`. Anda memerlukan jalur ini untuk menyemai klaster baru dengan data di dalam cadangan ini.

Untuk informasi tambahan, lihat Pembatasan dan batasan bucket dalam Panduan Pengguna Amazon Simple Storage Service.

### Langkah 4: Beri ElastiCache akses baca ke file .rdb

Sekarang, beri ElastiCache akses baca ke file cadangan .rdb Anda. Anda memberi ElastiCache akses ke file cadangan Anda dengan cara yang berbeda tergantung apakah bucket Anda berada di dalam Wilayah AWS default atau di Wilayah AWS pilihan.

Wilayah AWS yang diperkenalkan sebelum 20 Maret 2019, diaktifkan secara default. Anda dapat langsung mulai bekerja di Wilayah AWS ini. Wilayah yang diperkenalkan setelah 20 Maret 2019, seperti Asia Pasific (Hong Kong) dan Timur Tengah (Bahrain) dinonaktifkan secara default. Anda harus mengaktifkan, atau memilih ikut serta, ke Wilayah ini sebelum dapat menggunakannya, seperti dijelaskan pada Mengelola Wilayah AWS di Referensi Umum AWS.

Pilih pendekatan Anda bergantung pada Wilayah AWS Anda:

- Untuk Wilayah default, gunakan prosedur di Memberi ElastiCache akses baca ke file .rdb di Wilayah default.
- Untuk Wilayah pilihan, gunakan prosedur di Beri ElastiCache akses baca ke file .rdb di Wilayah pilihan.

#### Memberi ElastiCache akses baca ke file .rdb di Wilayah default

Wilayah AWS yang diperkenalkan sebelum 20 Maret 2019, diaktifkan secara default. Anda dapat langsung mulai bekerja di Wilayah AWS ini. Wilayah yang diperkenalkan setelah 20 Maret 2019, seperti Asia Pasific (Hong Kong) dan Timur Tengah (Bahrain) dinonaktifkan secara default. Anda harus mengaktifkan, atau memilih ikut serta, ke Wilayah ini sebelum dapat menggunakannya, seperti dijelaskan pada Mengelola Wilayah AWS di Referensi Umum AWS.

Untuk memberi ElastiCache akses baca ke file backup di Wilayah AWS yang diaktifkan secara default

1. Masuk ke AWS Management Console dan buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Pilih nama bucket S3 yang berisi file .rdb Anda.
3. Pilih nama folder yang berisi file .rdb Anda.
4. Pilih nama file cadang .rdb Anda. Nama file yang terpilih muncul di atas tab di bagian atas halaman.
5. Pilih Izin.
6. Jika aws-scs-s3-readonly atau salah satu ID kanonik dalam daftar berikut ini tidak tercantum sebagai pengguna, lakukan hal berikut:

   a. Di bagian Akses untuk akun AWS lainnya, pilih Tambah penerima.

   b. Pada kotak itu, tambahkan ID kanonik dari Wilayah AWS, seperti ditunjukkan berikut:

      - Wilayah AWS GovCloud (AS-Barat):

        ```
        40fa568277ad703bd160f66ae4f83fc9dfdfd06c2f1b5060ca22442ac3ef8be6
        ```

        **Important**

        Backup harus ditempatkan di bucket S3 di AWS GovCloud (US) agar Anda dapat mengunduhnya ke klaster Redis di AWS GovCloud (US).

      - Wilayah AWS diaktifkan secara default:

        ```
        540804c33a284a299d2547575ce1010f2312ef3da9b3a053c8bc45bf233e4353
        ```

   c. Tetapkan izin pada bucket dengan memilih Ya untuk berikut ini:

      - Objek daftar/tulis
      - Objek baca/tulis izin ACL

   d. Pilih Simpan.

7. Pilih Gambaran Umum, lalu pilih Unduh.

Beri ElastiCache akses baca ke file .rdb di Wilayah pilihan

Wilayah AWS yang diperkenalkan sebelum 20 Maret 2019, diaktifkan secara default. Anda dapat langsung mulai bekerja di Wilayah AWS ini. Wilayah yang diperkenalkan setelah 20 Maret 2019, seperti Asia Pasific (Hong Kong) dan Timur Tengah (Bahrain) dinonaktifkan secara default. Anda harus mengaktifkan, atau memilih ikut serta, ke Wilayah ini sebelum dapat menggunakannya, seperti dijelaskan pada Mengelola Wilayah AWS di Referensi Umum AWS.

Sekarang, beri ElastiCache akses baca ke file cadangan .rdb Anda.

Untuk memberi ElastiCache akses baca ke file cadangan

1. Masuk ke AWS Management Console dan buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Pilih nama bucket S3 yang berisi file .rdb Anda.
3. Pilih nama folder yang berisi file .rdb Anda.
4. Pilih nama file cadang .rdb Anda. Nama file yang terpilih muncul di atas tab di bagian atas halaman.
5. Pilih tab Izin.
6. Di bagian Izin, pilih Kebijakan Bucket, lalu pilih Edit.
7. Perbarui kebijakan ini untuk memberikan kepada ElastiCache izin yang diperlukan untuk melakukan operasi:

   - Tambahkan `[ "Service" : "region-full-name.elasticache-snapshot.amazonaws.com" ]` ke `Principal`.
   - Menambahkan izin yang diperlukan berikut untuk mengekspor snapshot ke bucket Amazon S3:
     - `"s3:GetObject"`
     - `"s3:ListBucket"`
     - `"s3:GetBucketAcl"`

   Berikut adalah contoh tampilan kebijakan yang sudah diperbarui.

```
{
    "Version": "2012-10-17",
    "Id": "Policy15397346",
```

```
    "Statement": [
        {
            "Sid": "Stmt15399483",
            "Effect": "Allow",
            "Principal": {
                "Service": "ap-east-1.elasticache-snapshot.amazonaws.com"
            },
            "Action": [
                "s3:GetObject",
                "s3:ListBucket",
                "s3:GetBucketAcl"
            ],
            "Resource": [
                "arn:aws:s3:::example-bucket",
                "arn:aws:s3:::example-bucket/backup1.rdb",
                "arn:aws:s3:::example-bucket/backup2.rdb"
            ]
        }
    ]
}
```

8. Pilih Simpan perubahan.

Langkah 5: Melakukan seeding klaster ElastiCache dengan data file .rdb

Sekarang Anda siap untuk membuat klaster ElastiCache dan melakukan seeding dengan data dari file .rdb. Untuk membuat klaster, ikuti petunjuk di Membuat klaster atau Membuat grup replikasi Redis dari awal. Pastikan untuk memilih Redis sebagai mesin klaster Anda.

Metode yang Anda gunakan untuk memberitahukan ElastiCache tentang tempat mencari backup Redis yang Anda unggah ke Amazon S3 akan bergantung pada metode yang Anda gunakan untuk membuat klaster:

Semai klaster atau grup replikasi ElastiCache for Redis dengan data file .rdb

- Menggunakan konsol ElastiCache

  Saat memilih Pengaturan klaster, pilih Pulihkan dari cadangan sebagai metode pembuatan klaster Anda, lalu pilih Cadangan lain sebagai Sumber Anda di bagian Sumber cadangan. Di kotak Seeding lokasi S3 file RDB, ketikkan jalur Amazon S3 untuk file tersebut. Jika Anda memiliki beberapa file .rdb, ketikkan jalur untuk setiap file di dalam daftar yang dipisahkan koma. Jalur Amazon S3 terlihat seperti *myBucket*/*myFolder*/*myBackupFilename*.rdb.

- Menggunakan AWS CLI

  Jika Anda menggunakan operasi `create-cache-cluster` atau `create-replication-group`, gunakan parameter `--snapshot-arns` untuk menentukan ARN yang memenuhi syarat sepenuhnya untuk setiap file .rdb. Misalnya, `arn:aws:s3:::myBucket/myFolder/myBackupFilename.rdb`. ARN harus dapat diterjemahkan ke file backup yang Anda simpan di Amazon S3.

- Menggunakan API ElastiCache

  Jika Anda menggunakan `CreateCacheCluster` atau operasi API ElastiCache `CreateReplicationGroup`, gunakan parameter `SnapshotArns` untuk menentukan ARN yang memenuhi syarat sepenuhnya untuk setiap file .rdb. Misalnya, `arn:aws:s3:::myBucket/myFolder/myBackupFilename.rdb`. ARN harus dapat diterjemahkan ke file backup yang Anda simpan di Amazon S3.

> **Important**
>
> Saat melakukan seeding klaster Redis (mode klaster diaktifkan), Anda harus mengonfigurasi setiap grup simpul (serpihan) di dalam klaster atau grup replikasi baru. Gunakan parameter `--node-group-configuration` (API: `NodeGroupConfiguration`) untuk melakukannya. Untuk informasi selengkapnya tentang IAM, lihat hal berikut:
>
> - CLI: create-replication-group di Referensi AWS CLI
> - API: CreateReplicationGroup dalam Referensi API ElastiCache

Selama proses pembuatan klaster Anda, data dalam cadangan Redis Anda dituliskan ke klaster. Anda dapat memantau kemajuan dengan melihat pesan peristiwa ElastiCache. Untuk melakukan ini, lihat konsol ElastiCache dan pilih Peristiwa Cache. Anda juga dapat menggunakan antarmuka baris perintah ElastiCache AWS atau API ElastiCache untuk memperoleh pesan peristiwa. Untuk informasi selengkapnya, lihat Melihat peristiwa ElastiCache.

# Versi mesin dan peningkatannya

Bagian ini mencakup versi mesin Redis yang didukung dan cara untuk meningkatkannya.

Topik

# Versi mesin dan pemutakhiran

Versi ElastiCache for Redis diidentifikasi dengan versi semantik yang terdiri dari komponen UTAMA dan MINOR. Misalnya, di Redis 6.2, versi utamanya adalah 6, dan versi minornya adalah 2. Saat mengoperasikan klaster yang dirancang sendiri, ElastiCache for Redis juga mengekspos komponen PATCH, misalnya Redis 6.2.1, dan versi patch-nya adalah 1.

Versi UTAMA adalah untuk perubahan API yang tidak kompatibel dan versi MINOR untuk fungsionalitas baru yang ditambahkan dengan cara yang kompatibel dengan versi lama. Versi PATCH adalah untuk perbaikan bug yang kompatibel dengan versi lama dan perubahan non-fungsional.

## Manajemen versi untuk ElastiCache Serverless

ElastiCache Serverless secara otomatis menerapkan versi perangkat lunak MINOR dan PATCH terbaru ke cache Anda, tanpa dampak atau waktu henti apa pun ke aplikasi Anda. Anda tidak perlu melakukan tindakan apa pun.

Ketika versi UTAMA baru tersedia, ElastiCache Serverless akan mengirimkan notifikasi kepada Anda di konsol dan peristiwa di EventBridge. Anda dapat memilih untuk meningkatkan cache Anda ke versi utama terbaru dengan memodifikasi cache menggunakan Konsol, CLI, atau API, dan memilih versi mesin terbaru.

## Manajemen versi untuk klaster ElastiCache yang dirancang sendiri

Saat bekerja dengan klaster ElastiCache yang dirancang sendiri, Anda dapat mengontrol waktu peningkatan perangkat lunak yang menjalankan klaster cache Anda ke versi baru yang didukung oleh ElastiCache. Anda dapat mengontrol kapan harus meningkatkan cache Anda ke versi UTAMA, MINOR, dan PATCH terbaru yang tersedia. Anda dapat memulai peningkatan versi mesin pada grup klaster atau replikasi Anda dengan mengubahnya dan menentukan versi mesin baru.

Anda dapat mengontrol syarat dan waktu perangkat lunak yang mematuhi protokol dan menjalankan klaster cache Anda dimutakhirkan ke versi baru yang didukung oleh ElastiCache. Dengan tingkat kontrol ini, Anda dapat memelihara kompatibilitas dengan versi tertentu, menguji versi baru dengan aplikasi Anda sebelum di-deploy ke sistem produksi, dan melakukan peningkatan versi sesuai syarat dan waktu Anda sendiri.

Karena peningkatan versi mungkin menimbulkan beberapa risiko kompatibilitas, peningkatan tidak dilakukan secara otomatis. Anda sendiri yang harus memulai prosesnya.

Anda dapat memulai peningkatan versi mesin pada grup klaster atau replikasi Anda dengan mengubahnya dan menentukan versi mesin baru. Untuk informasi selengkapnya, lihat berikut ini:

- Mengubah cluster
- Mengubah grup replikasi

## Pertimbangan peningkatan saat bekerja dengan klaster yang dirancang sendiri

> **Note**
>
> Pertimbangan berikut hanya berlaku saat meningkatkan klaster yang dirancang sendiri. Pertimbangan ini tidak berlaku untuk ElastiCache Serverless.

Saat meningkatkan klaster yang dirancang sendiri, pertimbangkan hal berikut

- Pengelolaan versi mesin dirancang agar Anda dapat memiliki kontrol sebanyak mungkin terkait cara melakukan patching. Namun, ElastiCache berhak untuk melakukan patching pada klaster atas nama Anda jika terjadi kerentanan keamanan yang penting dalam perangkat lunak cache atau sistem.
- Mulai Redis 6.0, ElastiCache for Redis akan menawarkan versi tunggal untuk setiap rilis minor Redis OSS, alih-alih menawarkan beberapa versi patch.
- Mulai mesin Redis versi 5.0.6, Anda dapat meningkatkan versi klaster Anda dengan waktu henti yang minimal. Klaster tersedia untuk proses baca selama keseluruhan proses peningkatan dan tersedia untuk proses tulis untuk sebagian besar durasi peningkatan, kecuali selama operasi failover yang berlangsung beberapa detik.
- Anda juga dapat meningkatkan klaster ElastiCache Anda dengan versi yang lebih lama dari 5.0.6. Tindakan ini melibatkan proses yang sama tetapi mungkin mengalami waktu failover lebih lama selama penyebaran DNS (30 detik - 1 menit).
- Mulai Redis 7, ElastiCache for Redis mendukung peralihan antara Redis (mode klaster dinonaktifkan) dan Redis (mode klaster diaktifkan).
- Proses peningkatan mesin Amazon ElastiCache for Redis dirancang untuk melakukan upaya terbaik dalam mempertahankan data yang ada dan memerlukan keberhasilan replikasi Redis.
- Saat meningkatkan mesin, ElastiCache untuk Redis akan menghentikan koneksi klien yang ada. Untuk meminimalkan waktu henti selama peningkatan mesin, sebaiknya terapkan praktik terbaik

untuk klien Redis dengan percobaan ulang kesalahan dan mundur eksponensial, serta praktik terbaik untuk meminimalkan waktu henti selama pemeliharaan.

- Anda tidak dapat melakukan peningkatan secara langsung dari Redis (mode klaster dinonaktifkan) ke Redis (mode klaster diaktifkan) saat Anda meningkatkan mesin Anda. Prosedur berikut ini menunjukkan cara melakukan peningkatan dari Redis (mode klaster dinonaktifkan) ke Redis (mode klaster diaktifkan).

  Untuk melakukan peningkatan dari versi mesin Redis (mode klaster dinonaktifkan) ke Redis (mode klaster diaktifkan)

  1. Buat cadangan klaster atau grup replikasi Redis (mode klaster dinonaktifkan). Untuk informasi selengkapnya, lihat Mengambil cadangan manual.
  2. Gunakan cadangan untuk membuat dan menyemai klaster Redis (mode klaster diaktifkan) dengan satu serpihan (grup simpul). Tentukan versi baru mesin dan aktifkan mode klaster saat membuat klaster atau grup replikasi. Untuk informasi selengkapnya, lihat Melakukan seeding klaster yang dirancang sendiri dengan cadangan yang dibuat secara eksternal.
  3. Hapus klaster atau grup replikasi Redis (mode klaster dinonaktifkan). Untuk informasi lebih lanjut, lihat Menghapus klaster atau Menghapus grup replikasi.
  4. Skalakan klaster atau grup replikasi Redis (mode klaster diaktifkan) yang baru sesuai jumlah serpihan (grup simpul) yang Anda butuhkan. Untuk informasi selengkapnya, lihat Penskalaan klaster di Redis (Mode Klaster Diaktifkan)

- Saat meningkatkan versi utama mesin, misalnya dari 5.0.6 ke 6.0, Anda juga harus memilih grup parameter baru yang kompatibel dengan versi mesin yang baru tersebut.
- Untuk klaster Redis tunggal dan klaster dengan Multi-AZ dinonaktifkan, sebaiknya sediakan memori yang cukup untuk Redis seperti dijelaskan di Memastikan bahwa Anda memiliki cukup memori untuk membuat snapshot Redis. Dalam kasus ini, klaster primer tidak tersedia untuk melayani permintaan selama proses pemutakhiran.
- Untuk klaster Redis yang mengaktifkan Multi-AZ, sebaiknya jadwalkan pemutakhiran mesin saat lalu lintas operasi tulis masuk rendah. Saat memutakhirkan ke Redis 5.0.6 atau versi leboh tinggi, klaster primer terus tersedia untuk melayani permintaan selama proses pemutakhiran.

  Klaster dan grup replikasi dengan beberapa serpihan diproses dan di-patch sebagai berikut:

  - Semua serpihan diproses secara paralel. Hanya satu operasi pemutakhiran dilakukan pada satu serpihan kapan saja.

  - Di setiap serpihan, semua replika diproses sebelum primer diproses. Jika terdapat lebih sedikit replika dalam serpihan, primer di dalam serpihan itu mungkin diproses sebelum replika di serpihan lainnya selesai diproses.
  - Di semua serpihan, simpul primer diproses secara berurutan. Hanya satu simpul primer yang dimutakhirkan dalam satu waktu.
- Jika enkripsi diaktifkan di klaster atau grup replikasi Anda saat ini, Anda tidak dapat melakukan pemutakhiran ke versi mesin yang tidak mendukung enkripsi, seperti dari 3.2.6 ke 3.2.10.

## Cara memutakhirkan versi mesin

Anda memulai pemutakhiran versi mesin pada klaster atau grup replikasi Anda dengan mengubahnya menggunakan konsol ElastiCache, AWS CLI, atau API ElastiCache dan menetapkan versi mesin yang baru. Untuk informasi selengkapnya, lihat topik berikut.

| Cara mengubah klaster dan grup replikasi | |
|---|---|
| Klaster | Grup replikasi |
| Menggunakan AWS Management Console | Menggunakan AWS Management Console |
| Menggunakan AWS CLI | Menggunakan AWS CLI |
| Menggunakan ElastiCache API | Menggunakan API ElastiCache |

## Menyelesaikan peningkatan mesin Redis yang diblokir

Seperti ditunjukkan pada tabel berikut, operasi peningkatan mesin Redis Anda diblokir jika Anda memiliki operasi menaikkan skala yang tertunda.

| Operasi yang tertunda | Operasi yang diblokir |
|---|---|
| Menaikkan skala | Peningkatan mesin segera |
| Peningkatan mesin | Menaikkan skala segera |
| Menaikkan skala dan pemutakhiran mesin | Menaikkan skala segera |

| Operasi yang tertunda | Operasi yang diblokir |
| --- | --- |
| | Peningkatan mesin segera |

Untuk menyelesaikan peningkatan mesin Redis yang diblokir

- Lakukan salah satu hal berikut:
  - Jadwalkan operasi peningkatan mesin Redis Anda untuk periode pemeliharaan berikutnya dengan menghapus kotak centang Terapkan segera.

    Dengan CLI, gunakan `--no-apply-immediately`. Dengan API, gunakan `ApplyImmediately=false`.
  - Tunggu hingga periode pemeliharaan berikutnya (atau setelahnya) untuk melakukan operasi peningkatan mesin Redis Anda.
  - Tambahkan operasi penaikan skala Redis pada perubahan klaster ini dengan memilih kotak centang Terapkan Segera.

    Dengan CLI, gunakan `--apply-immediately`. Dengan API, gunakan `ApplyImmediately=true`.

    Pendekatan ini secara efektif membatalkan peningkatan mesin selama jendela pemeliharaan berikutnya dengan melakukannya segera.

# Versi ElastiCache for Redis yang Didukung

Cache ElastiCache Nirserver mendukung versi Redis berikut:

- ElastiCache for Redis versi 7.1 (Ditingkatkan)

Klaster ElastiCache yang dirancang sendiri mendukung versi Redis berikut:

- ElastiCache for Redis versi 7.1 (Ditingkatkan)
- ElastiCache for Redis versi 7.0 (Ditingkatkan)
- ElastiCache for Redis versi 6.2 (ditingkatkan)
- ElastiCache for Redis versi 6.0 (ditingkatkan)
- ElastiCache for Redis versi 5.0.6 (Ditingkatkan)
- ElastiCache for Redis versi 5.0.5 (usang, gunakan versi 5.0.6)
- ElastiCache for Redis versi 5.0.4 (usang, gunakan versi 5.0.6)
- ElastiCache for Redis versi 5.0.3 (usang, gunakan versi 5.0.6)
- ElastiCache for Redis versi 5.0.0 (usang, gunakan versi 5.0.6)
- ElastiCache for Redis versi 4.0.10 (Ditingkatkan)
- Akhir Hidup Sebelumnya (EOL) versi (3.x)
- Akhir Hidup Sebelumnya (EOL) versi (2.x)

## ElastiCache for Redis versi 7.1 (Ditingkatkan)

Rilis ini berisi peningkatan kinerja yang memungkinkan beban kerja mendorong throughput yang lebih tinggi dan latensi operasi yang lebih rendah. ElastiCache 7.1 memperkenalkan dua penyempurnaan utama:

Kami memperluas fungsionalitas thread I/O yang disempurnakan untuk juga menangani logika lapisan presentasi. Dengan lapisan presentasi, yang kami maksud adalah thread Enhanced I/O yang sekarang tidak hanya membaca input klien, tetapi juga mengurai input ke dalam format perintah biner Redis. Ini kemudian diteruskan ke thread utama untuk eksekusi yang memberikan peningkatan kinerja. Peningkatan pola akses memori Redis. Langkah-langkah eksekusi dari banyak operasi struktur data disisipkan, untuk memastikan akses memori paralel dan mengurangi latensi akses memori. Saat menjalankan ElastiCache pada `R7g.4xlarge` berbasis Graviton3 atau lebih besar, pelanggan dapat mencapai lebih dari 1 juta permintaan per detik per simpul. Dengan peningkatan kinerja ElastiCache for Redis v7.1, pelanggan dapat mencapai throughput hingga 100% lebih banyak

dan latensi P99 50% lebih rendah dibandingkan dengan ElastiCache for Redis v7.0. Peningkatan ini diaktifkan pada ukuran simpul dengan setidaknya 8 inti fisik (`2xlarge` pada Graviton, dan `4xlarge` pada x86), terlepas dari jenis CPU dan tidak memerlukan perubahan klien.

> **Note**
>
> ElastiCache v7.1 kompatibel dengan OSS Redis v7.0.

### ElastiCache for Redis versi 7.0 (Ditingkatkan)

ElastiCache for Redis 7.0 menambahkan sejumlah perbaikan dan dukungan untuk fungsionalitas baru:

- Fungsi Redis: ElastiCache for Redis 7 menambahkan dukungan untuk Fungsi Redis, dan memberikan pengalaman terkelola yang memungkinkan developer untuk mengeksekusi skrip LUA dengan logika aplikasi yang disimpan di klaster ElastiCache, tanpa mengharuskan klien untuk mengirim ulang skrip ke server dengan setiap koneksi.
- Perbaikan ACL: ElastiCache for Redis 7 menambahkan dukungan untuk versi Redis Daftar Kontrol Akses (ACL) berikutnya. Dengan ElastiCache for Redis 7, klien sekarang dapat menentukan beberapa set izin pada kunci atau ruang kunci tertentu di Redis.
- Serpihan Pub/Sub: ElastiCache for Redis 7 menambahkan dukungan untuk menjalankan fungsionalitas Redis Pub/Sub dengan cara pembuatan serpihan saat menjalankan ElastiCache di Mode Klaster Diaktifkan (CME). Kemampuan Redis Pub/Sub memungkinkan penerbit untuk mengeluarkan pesan ke sejumlah pelanggan di saluran. Dengan Amazon ElastiCache for Redis 7, saluran terikat pada serpihan di klaster ElastiCache, sehingga menghilangkan kebutuhan untuk menyebarkan informasi saluran di seluruh serpihan untuk meningkatkan skalabilitas.
- Multiplexing I/O yang disempurnakan: ElastiCache for Redis versi 7 memperkenalkan multiplexing I/O yang disempurnakan, yang memberikan peningkatan throughput dan mengurangi latensi untuk beban kerja throughput tinggi yang memiliki banyak koneksi klien bersamaan ke klaster ElastiCache. Misalnya, saat menggunakan klaster simpul r6g.xlarge dan menjalankan 5200 klien bersamaan, Anda dapat mencapai hingga 72% peningkatan throughput (operasi baca dan tulis per detik) dan latensi P99 dengan penurunan hingga 71%, dibandingkan dengan ElastiCache for Redis versi 6.

Untuk informasi selengkapnya tentang rilis Redis 7.0, lihat Catatan Rilis Redis 7.0 pada Redis di GitHub.

## ElastiCache for Redis versi 6.2 (ditingkatkan)

ElastiCache for Redis 6.2 menyertakan peningkatan performa untuk klaster berkemampuan TLS menggunakan tipe simpul x86 dengan 8 vCPU atau lebih atau tipe simpul Graviton2 dengan 4 vCPU atau lebih. Peningkatan ini meningkatkan throughput dan mengurangi waktu pembentukan koneksi klien dengan membongkar enkripsi ke vCPU lain. Dengan Redis 6.2, Anda juga dapat mengelola akses ke saluran Pub/Sub dengan aturan Daftar Kontrol Akses (ACL).

Dengan versi ini, kami juga memperkenalkan dukungan untuk tingkatan data pada simpul klaster yang berisi NVMe SSD yang terpasang secara lokal. Untuk informasi selengkapnya, lihat Tingkatan data.

Mesin Redis versi 6.2.6 juga memperkenalkan dukungan format JavaScript Object Notation (JSON) asli, yang merupakan cara sederhana dan tanpa skema untuk menyandikan set data yang kompleks di dalam klaster Redis. Dengan dukungan JSON, Anda dapat memanfaatkan performa dan API Redis untuk aplikasi yang beroperasi melalui JSON. Untuk informasi selengkapnya, silakan lihat Memulai dengan JSON. Juga turut disertakan adalah metrik `JsonBasedCmds` dan `JsonBasedCmdsLatency` yang terkait JSON yang dimasukkan ke dalam CloudWatch untuk memantau penggunaan jenis data ini. Untuk informasi selengkapnya, lihat Metrik untuk Redis.

Anda menentukan versi mesin dengan menggunakan 6.2. ElastiCache for Redis secara otomatis akan menginvokasi versi patch pilihan dari Redis 6.2 yang tersedia. Sebagai contoh, ketika Anda membuat/mengubah klaster cache, Anda menetapkan parameter `--engine-version` ke 6.2. Klaster akan diluncurkan dengan versi patch pilihan Redis 6.2 yang tersedia saat ini pada saat pembuatan/modifikasi. Menentukan mesin versi 6.x di API akan menghasilkan versi minor terbaru dari Redis 6.

Untuk klaster 6.0 yang ada, Anda dapat memilih untuk meningkatkan versi minor otomatis berikutnya dengan menetapkan parameter `AutoMinorVersionUpgrade` ke `yes` di API `CreateCacheCluster`, `ModifyCacheCluster`, `CreateReplicationGroup` atau `ModifyReplicationGroup`. ElastiCache for Redis akan meningkatkan versi minor dari klaster 6.0 yang ada menjadi 6.2 menggunakan pembaruan mandiri. Untuk informasi selengkapnya, lihat Pembaruan mandiri di Amazon ElastiCache.

Saat memanggil API DescribeCacheEngineVersions, nilai parameter `EngineVersion` akan ditetapkan menjadi 6.2 dan versi mesin yang sebenarnya dengan versi patch akan dikembalikan pada bidang `CacheEngineVersionDescription`.

Untuk informasi selengkapnya tentang rilis Redis 6.2, lihat Catatan Rilis Redis 6.2 pada Redis di GitHub.

## ElastiCache for Redis versi 6.0 (ditingkatkan)

Amazon ElastiCache for Redis memperkenalkan versi berikutnya dari mesin Redis, yang mencakup Melakukan Autentikasi Pengguna dengan Kontrol Akses Berbasis Peran, caching sisi klien dan perbaikan operasional yang signifikan.

Mulai dari Redis 6.0, ElastiCache for Redis akan menawarkan versi tunggal untuk setiap rilis minor Redis OSS, bukannya menawarkan beberapa versi patch. ElastiCache for Redis secara otomatis akan mengelola versi patch klaster cache Anda yang sedang berjalan, sehingga memastikan peningkatan performa dan keamanan.

Anda juga dapat memilih untuk peningkatan versi minor otomatis berikutnya dengan menetapkan parameter `AutoMinorVersionUpgrade` ke `yes` dan ElastiCache for Redis akan mengelola peningkatan versi minor, melalui pembaruan mandiri. Untuk informasi selengkapnya, lihat Pembaruan layanan di ElastiCache.

Anda menentukan versi mesin dengan menggunakan `6.0`. ElastiCache for Redis secara otomatis akan menginvokasi versi patch pilihan dari Redis 6.0 yang tersedia. Sebagai contoh, ketika Anda membuat/mengubah klaster cache, Anda menetapkan parameter `--engine-version` ke 6.0. Klaster akan diluncurkan dengan versi patch pilihan Redis 6.0 yang tersedia saat ini pada saat pembuatan/modifikasi. Permintaan apa pun dengan nilai versi patch tertentu akan ditolak, pengecualian akan dikeluarkan dan proses akan gagal.

Saat memanggil API DescribeCacheEngineVersions, nilai parameter `EngineVersion` akan ditetapkan menjadi 6.0 dan versi mesin yang sebenarnya dengan versi patch akan dikembalikan pada bidang `CacheEngineVersionDescription`.

Untuk informasi selengkapnya tentang rilis Redis 6.0, lihat Catatan Rilis Redis 6.0 pada Redis di GitHub.

## ElastiCache for Redis versi 5.0.6 (Ditingkatkan)

Amazon ElastiCache for Redis memperkenalkan versi berikutnya dari mesin Redis, yang mencakup perbaikan bug dan pembaruan kumulatif berikut:

- Jaminan kestabilan mesin dalam kondisi khusus.
- Peningkatan penanganan kesalahan Hyperloglog.
- Peningkatan perintah jabat tangan untuk replikasi yang andal.
- Pelacakan pengiriman pesan yang konsisten melalui perintah `XCLAIM`.

- Peningkatan pengelolaan bidang LFU dalam objek.
- Manajemen transaksi yang ditingkatkan saat menggunakan ZPOP.
- Kemampuan untuk mengubah nama perintah: Parameter yang disebut `rename-commands` memungkinkan Anda untuk mengubah nama perintah Redis yang berpotensi bahaya atau yang menghambat program yang dapat menyebabkan kehilangan data tanpa disengaja, seperti FLUSHALL atau FLUSHDB. Ini mirip dengan konfigurasi mengganti nama perintah pada Redis sumber terbuka. Namun, ElastiCache telah meningkatkan pengalaman program dengan menyediakan alur kerja yang terkelola sepenuhnya. Perubahan nama perintah diterapkan segera, dan secara otomatis disebarkan ke semua simpul di dalam klaster yang berisi daftar perintah. Tidak ada intervensi yang diperlukan di sisi Anda, seperti boot ulang simpul.

  Contoh berikut menunjukkan cara mengubah grup parameter yang telah ada. Yang termasuk di sini adalah parameter `rename-commands`, yang merupakan daftar dipisahkan spasi berisi perintah yang ingin diubah namanya:

```
aws elasticache modify-cache-parameter-group --cache-parameter-group-
name custom_param_group
--parameter-name-values "ParameterName=rename-commands,  ParameterValue='flushall
 restrictedflushall'" --region region
```

  Pada contoh ini, parameter rename-command digunakan untuk mengubah nama perintah `flushall` menjadi `restrictedflushall`.

  Untuk mengubah nama beberapa perintah, gunakan cara berikut:

```
aws elasticache modify-cache-parameter-group --cache-parameter-group-
name custom_param_group
--parameter-name-values "ParameterName=rename-commands,  ParameterValue='flushall
 restrictedflushall flushdb restrictedflushdb'" --region region
```

  Untuk membalikkan perubahan apapun, jalankan kembali perintah dan keluarkan semua nilai perubahan nama dari daftar `ParameterValue` yang ingin dipertahankan, seperti ditunjukkan berikut:

```
aws elasticache modify-cache-parameter-group --cache-parameter-group-
name custom_param_group
--parameter-name-values "ParameterName=rename-commands,  ParameterValue='flushall
 restrictedflushall'" --region region
```

Dalam hal ini, perintah `flushall` diubah namanya menjadi `restrictedflushall` dan perintah perubahan nama lainnya kembali ke nama perintah aslinya.

> **Note**
>
> Saat mengubah nama perintah, Anda dibatasi pada keterbatasan berikut:
>
> - Semua perintah ubah nama harus alfanumerik.
> - Panjang maksimum nama perintah baru adalah 20 karakter alfanumerik.
> - Ketika mengubah nama perintah, pastikan bahwa Anda memperbarui grup parameter yang terkait dengan klaster Anda.
> - Untuk mencegah penggunaan sebuah perintah sepenuhnya, gunakan kata kunci `blocked`, seperti yang ditunjukkan berikut ini:
>
> ```
> aws elasticache modify-cache-parameter-group --cache-parameter-group-
> name custom_param_group
> --parameter-name-values "ParameterName=rename-commands,
>  ParameterValue='flushall blocked'" --region region
> ```

Untuk informasi lain tentang perubahan parameter dan daftar perintah apa yang layak untuk perubahan nama, lihat Perubahan parameter Redis 5.0.3.

- Pengaliran Redis: Ini memodelkan struktur data log yang memungkinkan produsen menambahkan item baru secara waktu nyata. Ini juga memungkinkan pelanggan menerima pesan baik dalam mode memblokir ataupun tidak memblokir. Pengaliran juga memungkinkan grup pelanggan, yang mewakili sekelompok klien untuk secara kooperatif menggunakan bagian yang berbeda dari pengaliran pesan yang sama, mirip dengan Apache Kafka. Untuk informasi selengkapnya, lihat Pengantar Pengaliran Redis.
- Dukungan untuk keluarga perintah pengaliran, seperti `XADD`, `XRANGE` dan `XREAD`. Untuk informasi selengkapnya, lihat Perintah Pengaliran Redis.
- Sejumlah parameter baru dan perubahan nama. Untuk informasi selengkapnya, lihat Perubahan parameter Redis 5.0.0.
- Metrik baru Redis, `StreamBasedCmds`.
- Waktu snapshot yang sedikit lebih cepat untuk simpul Redis.

> **Important**
>
> Amazon ElastiCache for Redis telah melakukan porting balik dua perbaikan bug penting dari Redis sumber terbuka versi 5.0.1. Perbaikan tersebut tercantum sebagai berikut:
>
> - PULIHKAN balasan yang tidak sesuai ketika kunci tertentu telah kedaluwarsa.
> - Perintah `XCLAIM` dapat berpotensi menghasilkan entri yang salah atau mengganggu sinkronisasi protokol.
>
> Kedua perbaikan bug ini disertakan dalam dukungan ElastiCache for Redis untuk versi mesin Redis 5.0.0 dan digunakan pada pembaruan versi mendatang.

Untuk informasi selengkapnya, silakan lihat Catatan Rilis Redis 5.0.6 pada Redis di GitHub.

## ElastiCache for Redis versi 5.0.5 (usang, gunakan versi 5.0.6)

Amazon ElastiCache for Redis memperkenalkan versi berikutnya dari mesin Redis;. Ini mencakup perubahan konfigurasi online untuk ElastiCache for Redis dari klaster failover otomatis selama semua operasi yang direncanakan. Anda sekarang dapat menskalakan klaster Anda, meningkatkan versi mesin Redis serta menerapkan patch dan pembaruan pemeliharaan sementara klaster tetap online dan terus melayani permintaan yang masuk. Ini juga menyertakan perbaikan bug.

Untuk informasi selengkapnya, silakan lihat Catatan Rilis Redis 5.0.5 pada Redis di GitHub.

## ElastiCache for Redis versi 5.0.4 (usang, gunakan versi 5.0.6)

Amazon ElastiCache for Redis memperkenalkan versi berikutnya dari mesin Redis yang didukung oleh Amazon ElastiCache. Ini mencakup peningkatan berikut ini:

- Jaminan kestabilan mesin dalam kondisi khusus.
- Peningkatan penanganan kesalahan Hyperloglog.
- Peningkatan perintah jabat tangan untuk replikasi yang andal.
- Pelacakan pengiriman pesan yang konsisten melalui perintah `XCLAIM`.
- Peningkatan pengelolaan bidang `LFU` dalam objek.
- Manajemen transaksi yang ditingkatkan saat menggunakan `ZPOP`.

Untuk informasi selengkapnya, silakan lihat Catatan Rilis Redis 5.0.4 pada Redis di GitHub.

### ElastiCache for Redis versi 5.0.3 (usang, gunakan versi 5.0.6)

Amazon ElastiCache for Redis memperkenalkan versi berikutnya dari mesin Redis yang didukung oleh Amazon ElastiCache yang menyertakan perbaikan bug.

### ElastiCache for Redis versi 5.0.0 (usang, gunakan versi 5.0.6)

Amazon ElastiCache for Redis memperkenalkan versi utama berikutnya dari mesin Redis yang didukung oleh Amazon ElastiCache. ElastiCache for Redis 5.0.0 menghadirkan dukungan untuk perbaikan berikut:

- Pengaliran Redis: Ini memodelkan struktur data log yang memungkinkan produsen menambahkan item baru secara waktu nyata. Ini juga memungkinkan pelanggan menerima pesan baik dalam mode memblokir ataupun tidak memblokir. Pengaliran juga memungkinkan grup pelanggan, yang mewakili sekelompok klien untuk secara kooperatif menggunakan bagian yang berbeda dari pengaliran pesan yang sama, mirip dengan Apache Kafka. Untuk informasi selengkapnya, lihat Pengantar Pengaliran Redis.
- Dukungan untuk keluarga perintah pengaliran, seperti `XADD`, `XRANGE` dan `XREAD`. Untuk informasi selengkapnya, lihat Perintah Pengaliran Redis.
- Sejumlah parameter baru dan perubahan nama. Untuk informasi selengkapnya, lihat Perubahan parameter Redis 5.0.0.
- Metrik baru Redis, `StreamBasedCmds`.
- Waktu snapshot yang sedikit lebih cepat untuk simpul Redis.

### ElastiCache for Redis versi 4.0.10 (Ditingkatkan)

Amazon ElastiCache for Redis memperkenalkan versi utama berikutnya dari mesin Redis yang didukung oleh Amazon ElastiCache. ElastiCache for Redis 4.0.10 menghadirkan dukungan untuk perbaikan berikut:

- Perubahan ukuran dan enkripsi klaster secara online dalam satu versi ElastiCache for Redis tunggal. Untuk informasi selengkapnya tentang IAM, lihat hal berikut:
  - Penskalaan klaster di Redis (Mode Klaster Diaktifkan)
  - Resharding dan penyeimbangan ulang serpihan secara online untuk Redis (mode klaster diaktifkan)

  - Keamanan data di Amazon ElastiCache
- Sejumlah parameter baru. Untuk informasi selengkapnya, lihat Perubahan parameter Redis 4.0.10.
- Dukungan untuk keluarga perintah memori, seperti `MEMORY`. Untuk informasi selengkapnya, lihat Perintah Redis (cari di MEMO).
- Dukungan untuk defragmentasi memori saat online sehingga memungkinkan pemanfaatan memori yang lebih efisien dan lebih banyak memori yang tersedia untuk data Anda.
- Dukungan untuk flush dan penghapusan asinkron. ElastiCache for Redis mendukung perintah seperti `UNLINK`, `FLUSHDB` dan `FLUSHALL` untuk dijalankan dalam thread yang berbeda dari thread utama. Melakukan hal ini membantu meningkatkan kinerja dan waktu respons untuk aplikasi Anda dengan membebaskan memori secara asinkron.
- Metrik baru Redis, `ActiveDefragHits`. Untuk informasi selengkapnya, lihat Metrik untuk Redis.

Pengguna Redis (mode klaster dinonaktifkan) yang menjalankan Redis versi 3.2.10 dapat menggunakan konsol untuk meningkatkan klaster mereka melalui peningkatan online.

Membandingkan dukungan perubahan ukuran dan enkripsi klaster ElastiCache for Redis

| Fitur | 3.2.6 | 3.2.10 | 4.0.10 dan yang lebih baru |
|---|---|---|---|
| Perubahan ukuran klaster online * | Tidak | Ya | Ya |
| Enkripsi bergerak ** | Ya | Tidak | Ya |
| Enkripsi diam ** | Ya | Tidak | Ya |
| * Menambahkan, menghapus, dan menyeimbangkan kembali serpihan. | | | |
| ** Diwajibkan untuk aplikasi yang mematuhi FedRAMP, HIPAA, dan PCI DSS. Untuk informasi selengkapnya, lihat Validasi kepatuhan untuk Amazon ElastiCache. | | | |

## Akhir Hidup Sebelumnya (EOL) versi (3.x)

### ElastiCache for Redis versi 3.2.10 (Ditingkatkan)

Amazon ElastiCache for Redis memperkenalkan versi utama berikutnya dari mesin Redis yang didukung oleh Amazon ElastiCache. ElastiCache for Redis 3.2.10 memperkenalkan perubahan ukuran klaster online untuk menambah atau menghapus serpihan dari klaster sementara klaster terus melayani permintaan I/O yang masuk. Pengguna ElastiCache for Redis 3.2.10 memiliki semua fungsionalitas versi Redis sebelumnya kecuali kemampuan untuk mengenkripsi data mereka. Kemampuan ini saat ini hanya tersedia pada versi 3.2.6.

Membandingkan ElastiCache for Redis versi 3.2.6 dan 3.2.10

| Fitur | 3.2.6 | 3.2.10 |
|---|---|---|
| Perubahan ukuran klaster online * | Tidak | Ya |
| Enkripsi bergerak ** | Ya | Tidak |
| Enkripsi diam ** | Ya | Tidak |
| * Menambahkan, menghapus, dan menyeimbangkan kembali serpihan. | | |
| ** Diwajibkan untuk aplikasi yang mematuhi FedRAMP, HIPAA, dan PCI DSS. Untuk informasi selengkapnya, lihat Validasi kepatuhan untuk Amazon ElastiCache. | | |

Untuk informasi selengkapnya tentang IAM, lihat hal berikut:

- Resharding dan penyeimbangan ulang serpihan secara online untuk Redis (mode klaster diaktifkan)
- Perubahan ukuran klaster online

### ElastiCache for Redis versi 3.2.6 (Ditingkatkan)

Amazon ElastiCache for Redis memperkenalkan versi utama berikutnya dari mesin Redis yang didukung oleh Amazon ElastiCache. Pengguna ElastiCache for Redis 3.2.6 memiliki semua fungsionalitas versi Redis sebelumnya plus opsi untuk mengenkripsi data mereka. Untuk informasi selengkapnya tentang IAM, lihat hal berikut:

- Enkripsi bergerak ElastiCache (TLS)
- Enkripsi Diam di ElastiCache
- Validasi kepatuhan untuk Amazon ElastiCache

ElastiCache for Redis versi 3.2.4 (Ditingkatkan)

Amazon ElastiCache for Redis versi 3.2.4 memperkenalkan versi utama berikutnya dari mesin Redis yang didukung oleh Amazon ElastiCache. Pengguna ElastiCache for Redis 3.2.4 memiliki semua fungsionalitas versi Redis sebelumnya yang tersedia bagi mereka ditambah opsi untuk dijalankan dalam mode klaster atau mode non-klaster. Tabel berikut merangkum hal ini.

Membandingkan Redis 3.2.4 mode non-klaster dan mode klaster

| Fitur | Mode non-klaster | Mode klaster |
|---|---|---|
| Pembuatan Partisi Data | Tidak | Ya |
| Pengindeksan geospasial | Ya | Ya |
| Ubah jenis simpul | Ya | Ya * |
| Penskalaan replika | Ya | Ya * |
| Menskalakan ke luar | Tidak | Ya * |
| Dukungan basis data | Beberapa | Tunggal |
| Grup parameter | `default.redis3.2` ** | `default.redis3.2.cluster.on` ** |
| * Lihat Melakukan pemulihan dari cadangan ke dalam cache baru | | |
| ** Atau satu yang berasal dari itu. | | |

Catatan:

- Pembuatan partisi – kemampuan untuk membagi data Anda pada 2 hingga 500 grup simpul (serpihan) dengan dukungan replikasi untuk setiap grup simpul.

- Pengindeksan geospasial – Redis 3.2.4 memperkenalkan dukungan untuk pengindeksan geospasial melalui enam perintah GEO. Untuk informasi selengkapnya, lihat dokumentasi perintah GEO* Redis Perintah Redis: GEO pada halaman Perintah Redis (disaring untuk GEO).

Untuk informasi tentang fitur Redis 3 tambahan, lihat Catatan rilis Redis 3.2 dan Catatan rilis Redis 3.0.

Saat ini, Redis (mode klaster diaktifkan) yang dikelola ElastiCache tidak mendukung fitur Redis 3.2 berikut:

- Migrasi replika
- Penyeimbangan kembali klaster
- Lua debugger

ElastiCache menonaktifkan perintah pengelolaan Redis 3.2 berikut:

- `cluster meet`
- `cluster replicate`
- `cluster flushslots`
- `cluster addslots`
- `cluster delslots`
- `cluster setslot`
- `cluster saveconfig`
- `cluster forget`
- `cluster failover`
- `cluster bumpepoch`
- `cluster set-config-epoch`
- `cluster reset`

Untuk informasi tentang parameter Redis 3.2.4, lihat Perubahan parameter Redis 3.2.4.

## Akhir Hidup Sebelumnya (EOL) versi (2.x)

ElastiCache for Redis versi 2.8.24 (Ditingkatkan)

Perbaikan Redis ditambahkan sejak versi 2.8.23 termasuk perbaikan bug dan pencatatan log dari alamat akses memori yang buruk. Untuk informasi selengkapnya, lihat Catatan rilis Redis 2.8.

ElastiCache for Redis versi 2.8.23 (Ditingkatkan)

Perbaikan Redis ditambahkan sejak versi 2.8.22 termasuk perbaikan bug. Untuk informasi selengkapnya, lihat Catatan rilis Redis 2.8. Rilis ini mencakup dukungan untuk parameter baru `close-on-slave-write` yang, jika diaktifkan, memutuskan klien yang mencoba untuk menulis ke replika baca-saja.

Untuk informasi selengkapnya tentang parameter Redis 2.8.23, lihat Parameter yang ditambahkan dalam Redis 2.8.23 (ditingkatkan) pada Panduan Pengguna ElastiCache.

ElastiCache for Redis versi 2.8.22 (Ditingkatkan)

Perbaikan Redis ditambahkan sejak versi 2.8.21 meliputi hal berikut:

- Dukungan untuk pencadangan dan sinkronisasi forkless, yang memungkinkan Anda mengalokasikan lebih sedikit memori untuk overhead cadangan dan lebih banyak untuk aplikasi Anda. Untuk informasi selengkapnya, lihat Cara penerapan sinkronisasi dan pencadangan. Proses forkless dapat mempengaruhi latensi dan throughput. Saat ada throughput proses tulis yang tinggi, dan replika melakukan sinkronisasi ulang, proses tulis dapat menjadi tidak terjangkau selama keseluruhan waktu sinkronisasi itu.
- Jika ada failover, grup replikasi sekarang akan pulih lebih cepat karena replika melakukan sinkronisasi parsial dengan primer bukannya sinkronisasi penuh bila memungkinkan. Selain itu, primer dan replika tidak lagi menggunakan disk selama sinkronisasi, yang menyediakan keuntungan selanjutnya pada kecepatan.
- Dukungan untuk dua metrik CloudWatch baru.
  - `ReplicationBytes` – Jumlah byte yang dikirimkan klaster primer grup replikasi ke replika baca.
  - `SaveInProgress` – Nilai biner yang menunjukkan apakah ada proses simpan di latar belakang yang bekerja.

  Untuk informasi selengkapnya, lihat Memantau penggunaan dengan Metrik CloudWatch.

- Sejumlah perbaikan bug penting dalam perilaku replikasi PSYNC. Untuk informasi selengkapnya, lihat Catatan rilis Redis 2.8.
- Untuk memelihara performa replikasi yang ditingkatkan dalam grup replikasi Multi-AZ dan untuk meningkatkan stabilitas klaster, replika non-ElastiCache tidak lagi didukung.
- Untuk memperbaiki konsistensi data antara klaster primer dan replika dalam grup replikasi, replika tidak lagi mengosongkan kunci yang independen dari klaster primer.
- Variabel konfigurasi Redis `appendonly` dan `appendfsync` tidak didukung pada Redis versi 2.8.22 dan yang lebih baru.
- Dalam situasi memori rendah, klien dengan buffer keluaran yang besar mungkin terputus dari klaster replika. Jika terputus, klien perlu menyambung kembali. Situasi seperti itu kemungkinan besar terjadi untuk klien PUBSUB.

ElastiCache for Redis versi 2.8.21

Perbaikan Redis ditambahkan sejak versi 2.8.19 termasuk sejumlah perbaikan bug. Untuk informasi selengkapnya, lihat Catatan rilis Redis 2.8.

ElastiCache for Redis versi 2.8.19

Perbaikan Redis ditambahkan sejak versi 2.8.6 meliputi hal berikut:

- Dukungan untuk HyperLogLog. Untuk informasi selengkapnya, lihat Struktur data baru Redis: HyperLogLog.
- Jenis data sorted set sekarang memiliki dukungan untuk kueri kisaran leksikografis dengan perintah baru `ZRANGEBYLEX`, `ZLEXCOUNT`, dan `ZREMRANGEBYLEX`.
- Untuk mencegah simpul primer mengirimkan data usang untuk simpul replika, master SYNC gagal jika proses anak untuk menyimpan di latar belakang (`bgsave`) dibatalkan.
- Dukungan untuk metrik CloudWatch HyperLogLogBasedCommands. Untuk informasi selengkapnya, lihat Metrik untuk Redis.

ElastiCache for Redis versi 2.8.6

Perbaikan Redis ditambahkan sejak versi 2.6.13 meliputi hal berikut:

- Peningkatan ketahanan dan toleransi kesalahan untuk replika baca.
- Dukungan untuk resinkronisasi parsial.

- Dukungan untuk jumlah minimum replika baca yang ditentukan pengguna yang harus tersedia setiap saat.
- Dukungan penuh untuk pub/sub—memberitahukan klien tentang peristiwa di server.
- Deteksi otomatis untuk kegagalan simpul primer dan failover dari simpul primer Anda ke simpul sekunder.

ElastiCache for Redis versi 2.6.13

Redis versi 2.6.13 adalah versi awal Redis yang didukung oleh Amazon ElastiCache for Redis. Multi-AZ tidak didukung pada Redis 2.6.13.

## Jadwal akhir masa pakai versi Redis

Bagian ini mendefinisikan tanggal akhir masa pakai (EOL) untuk versi utama yang lebih lama saat diumumkan. Hal ini membantu Anda mengambil keputusan terkait versi dan peningkatan pada masa mendatang.

> **Note**
>
> Versi patch ElastiCache for Redis dari 5.0.0 hingga 5.0.5 tidak digunakan lagi. Gunakan versi 5.0.6 atau yang lebih baru.

Tabel berikut merangkum setiap versi dan tanggal EOL yang diumumkan, serta versi target peningkatan yang direkomendasikan.

EOL terdahulu

| V U S | Versi Minor Sumber | Target Peningkatan yang Disarankan | Tanggal EOL |
|---|---|---|---|
| V 3 | 3.2.4, 3.2.6 dan 3.2.10 | Versi 6.2 atau yang lebih baru<br><br>**Note**<br>Untuk Wilayah US-ISO-EAST-1, US- | 31 Juli 2023 |

| V U S | Versi Minor Sumber | Target Peningkatan yang Disarankan | Tanggal EOL |
|---|---|---|---|
| | | ISO-WEST-1, dan US-ISOB-EAST-1, kami merekomendasikan 5.0.6 atau yang lebih baru. | |
| V 2 | 2.8.24, 2.8.23, 2.8.22, 2.8.21, 2.8.19, 2.8.12, 2.8.6, 2.6.13 | Versi 6.2 atau yang lebih baru<br>**Note**<br>Untuk Wilayah US-ISO-EAST-1, US-ISO-WEST-1, dan US-ISOB-EAST-1, kami merekomendasikan 5.0.6 atau yang lebih baru. | 13 Januari 2023 |

## Cara meningkatkan versi mesin

Anda memulai peningkatan versi mesin pada klaster atau grup replikasi Anda dengan mengubahnya menggunakan konsol ElastiCache, AWS CLI, atau API ElastiCache dan menentukan versi mesin yang lebih baru. Untuk informasi selengkapnya, lihat topik berikut.

| Cara mengubah klaster dan grup replikasi | |
|---|---|
| Klaster | Grup replikasi |
| Menggunakan AWS Management Console | Menggunakan AWS Management Console |
| Menggunakan AWS CLI | Menggunakan AWS CLI |
| Menggunakan ElastiCache API | Menggunakan API ElastiCache |

## Menyelesaikan peningkatan mesin Redis yang diblokir

Seperti ditunjukkan pada tabel berikut, operasi peningkatan mesin Redis Anda diblokir jika Anda memiliki operasi menaikkan skala yang tertunda.

| Operasi yang tertunda | Operasi yang diblokir |
|---|---|
| Menaikkan skala | Peningkatan mesin segera |
| Peningkatan mesin | Menaikkan skala segera |
| Menaikkan skala dan pemutakhiran mesin | Menaikkan skala segera |
| | Peningkatan mesin segera |

Untuk menyelesaikan peningkatan mesin Redis yang diblokir

- Lakukan salah satu hal berikut:
  - Jadwalkan operasi peningkatan mesin Redis Anda untuk periode pemeliharaan berikutnya dengan menghapus kotak centang Terapkan segera.

Dengan CLI, gunakan `--no-apply-immediately`. Dengan API, gunakan `ApplyImmediately=false`.

- Tunggu hingga periode pemeliharaan berikutnya (atau setelahnya) untuk melakukan operasi peningkatan mesin Redis Anda.
- Tambahkan operasi penaikan skala Redis pada perubahan klaster ini dengan memilih kotak centang Terapkan Segera.

  Dengan CLI, gunakan `--apply-immediately`. Dengan API, gunakan `ApplyImmediately=true`.

  Pendekatan ini secara efektif membatalkan peningkatan mesin selama jendela pemeliharaan berikutnya dengan melakukannya segera.

## Perilaku versi utama dan perbedaan kompatibilitas

> **Important**
>
> Halaman berikut disusun untuk menandakan semua perbedaan inkompatibilitas antar versi dan memberi tahu Anda tentang pertimbangan apa pun yang harus Anda buat saat meningkatkan ke versi yang lebih baru. Daftar ini termasuk masalah inkompatibilitas versi apa pun yang mungkin Anda temui saat melakukan peningkatan.
> Anda dapat melakukan peningkatan langsung dari versi Redis Anda saat ini ke versi Redis terbaru yang tersedia, tanpa perlu peningkatan berurutan. Misalnya, Anda dapat melakukan peningkatan langsung dari Redis versi 3.0 ke versi 7.0.

Versi Redis diidentifikasi dengan versi semantik yang terdiri dari komponen UTAM, MINOR, dan PATCH. Misalnya, di Redis 4.0.10, versi utamanya adalah 4, versi minornya adalah 0, dan versi patch-nya adalah 10. Nilai-nilai ini umumnya bertambah berdasarkan konvensi berikut:

- Versi UTAMA adalah untuk perubahan API yang tidak kompatibel
- Versi MINOR adalah untuk fungsionalitas baru yang ditambahkan dengan cara yang kompatibel dengan versi lama
- Versi PATCH adalah untuk perbaikan bug yang kompatibel dengan versi lama dan perubahan non-fungsional

Sebaiknya selalu gunakan versi patch terbaru dalam versi UTAMA.MINOR tertentu untuk mendapatkan peningkatan kinerja dan stabilitas terbaru. Mulai Redis 6.0, ElastiCache for Redis akan menawarkan versi tunggal untuk setiap rilis minor Redis OSS, alih-alih menawarkan beberapa versi patch. ElastiCache untuk Redis secara otomatis akan mengelola versi patch klaster cache Anda yang sedang berjalan, sehingga memastikan peningkatan performa dan keamanan.

Kami juga merekomendasikan melakukan peningkatan secara berkala ke versi utama terbaru, karena sebagian besar perbaikan utama tidak kembali dipindahkan ke versi lama. Karena ElastiCache memperluas ketersediaan ke wilayah AWS baru, ElastiCache for Redis mendukung dua versi MAJOR.MINOR terbaru pada waktu itu untuk wilayah baru. Misalnya, jika wilayah AWS baru diluncurkan dan MAJOR.MINOR ElastiCache for Redis versi terbaru adalah 7.0 dan 6.2, ElastiCache for Redis akan mendukung versi 7.0 dan 6.2 di wilayah AWS baru tersebut. Karena versi MAJOR.MINOR yang lebih baru dari ElastiCache for Redis dirilis, ElastiCache akan terus menambahkan dukungan untuk versi ElastiCache for Redis yang baru dirilis. Untuk mempelajari lebih lanjut cara memilih wilayah untuk ElastiCache, lihat Memilih wilayah dan zona ketersediaan.

Saat melakukan peningkatan yang mencakup versi utama atau minor, harap pertimbangkan daftar berikut yang mencakup perilaku dan perubahan yang tidak kompatibel dengan versi lama yang dirilis dengan Redis dari waktu ke waktu.

## Perilaku Redis 7.0 dan perubahan yang tidak kompatibel dengan versi lama

Untuk daftar lengkap perubahan, lihat Catatan rilis Redis 7.0.

- `SCRIPT LOAD` dan `SCRIPT FLUSH` tidak lagi disebarkan ke replika. Jika Anda membutuhkan lebih banyak ketahanan untuk skrip, sebaiknya pertimbangkan untuk menggunakan fungsi Redis.
- Saluran Pubsub sekarang diblokir secara default untuk pengguna ACL baru.
- Perintah `STRALGO` diganti dengan perintah `LCS`.
- Format untuk `ACL GETUSER` telah berubah sehingga semua bidang menunjukkan pola string akses standar. Jika Anda menggunakan otomatisasi `ACL GETUSER`, Anda harus memverifikasi bahwa itu akan menangani salah satu format.
- Kategori ACL untuk `SELECT`, `WAIT`, `ROLE`, `LASTSAVE`, `READONLY`, `READWRITE`, dan `ASKING` telah berubah.
- Perintah `INFO` sekarang menunjukkan statistik perintah per sub-perintah, bukan di perintah kontainer tingkat atas.

- Nilai pengembalian perintah `LPOP`, `RPOP`, `ZPOPMIN` dan `ZPOPMAX` telah berubah dalam kasus ekstrem tertentu. Jika Anda menggunakan perintah ini, Anda harus memeriksa catatan rilis dan mengevaluasi apakah Anda terpengaruh.
- Perintah `SORT` dan `SORT_RO` sekarang memerlukan akses ke seluruh ruang kunci untuk menggunakan argumen `GET` dan `BY`.

## Perilaku Redis 6.2 dan perubahan yang tidak kompatibel dengan versi lama

Untuk daftar lengkap perubahan, lihat Catatan rilis Redis 6.2.

- Tanda ACL perintah `TIME`, `ECHO`, `ROLE`, dan `LASTSAVE` telah diubah. Hal ini dapat menyebabkan perintah yang sebelumnya diizinkan untuk ditolak dan sebaliknya.

> **Note**
>
> Tak satu pun dari perintah ini akan memodifikasi atau memberikan akses ke data.

- Saat melakukan peningkatan dari Redis 6.0, urutan pasangan kunci/nilai yang dikembalikan dari respons peta ke skrip lua akan diubah. Jika skrip Anda menggunakan `redis.setresp()` atau mengembalikan peta (baru di Redis 6.0), pertimbangkan implikasi bahwa skrip dapat rusak pada peningkatan.

## Perilaku Redis 6.0 dan perubahan yang tidak kompatibel dengan versi lama

Untuk daftar lengkap perubahan, lihat Catatan rilis Redis 6.0.

- Jumlah maksimum basis data yang diizinkan telah dikurangi dari 1,2 juta menjadi 10 ribu. Nilai defaultnya adalah 16, dan kami tidak menyarankan penggunaan nilai yang jauh lebih besar dari ini karena kami telah menemukan masalah performa dan memori.
- Setel `AutoMinorVersionUpgrade` parameter ke ya, dan ElastiCache for Redis akan mengelola peningkatan versi minor melalui pembaruan layanan mandiri. Hal ini akan ditangani lewat saluran notifikasi pelanggan standar melalui kampanye pembaruan secara mandiri. Untuk informasi selengkapnya, lihat Pembaruan secara mandiri di ElastiCache.

## Perilaku Redis 5.0 dan perubahan yang tidak kompatibel dengan versi lama

Untuk daftar lengkap perubahan, lihat Catatan rilis Redis 5.0.

- Skrip dengan direplikasi oleh efek bukannya mengeksekusi ulang skrip pada replika. Hal ini umumnya akan meningkatkan kinerja tetapi dapat meningkatkan jumlah data yang direplikasi antara primer dan replika. Ada opsi untuk kembali ke perilaku sebelumnya yang hanya tersedia di ElastiCache for Redis 5.0.
- Jika Anda melakukan peningkatan dari Redis 4.0, beberapa perintah dalam skrip LUA akan mengembalikan argumen dalam urutan yang berbeda dari yang dilakukan di versi sebelumnya. Dalam Redis 4.0, Redis akan memesan beberapa tanggapan secara leksografis untuk membuat respons deterministik, urutan ini tidak diterapkan ketika skrip direplikasi oleh efek.
- Di Redis 5.0.3 dan yang lebih baru, ElastiCache for Redis akan mengurangi beberapa pekerjaan IO ke inti latar belakang di tipe instans dengan lebih dari 4 VCPU. Hal ini dapat mengubah karakteristik performa Redis dan mengubah nilai beberapa metrik. Untuk informasi selengkapnya, lihat Metrik Apa Yang Harus Dipantau? untuk memahami apakah Anda perlu mengubah metrik yang Anda lihat.

### Perilaku Redis 4.0 dan perubahan yang tidak kompatibel dengan versi lama

Untuk daftar lengkap perubahan, lihat Catatan rilis Redis 4.0.

- Log lambat sekarang mencatat log dua argumen tambahan, nama klien dan alamat. Perubahan ini harus kompatibel dengan versi lama kecuali Anda secara eksplisit mengandalkan setiap entri log lambat yang berisi 3 nilai.
- Perintah `CLUSTER NODES` sekarang mengembalikan format yang sedikit berbeda, yang tidak kompatibel dengan versi lama. Sebaiknya klien tidak menggunakan perintah ini untuk mempelajari simpul yang ada di cluster, dan sebagai gantinya mereka harus menggunakan `CLUSTER SLOTS`.

## EOL terdahulu

Perilaku Redis 3.2 dan perubahan yang tidak kompatibel dengan versi lama

Untuk daftar lengkap perubahan, lihat Catatan rilis Redis 3.2.

- Tidak ada perubahan kompatibilitas untuk memanggil versi ini.

Untuk informasi selengkapnya, lihat Jadwal akhir masa pakai versi Redis.

Perilaku Redis 2.8 dan perubahan yang tidak kompatibel dengan versi lama

Untuk daftar lengkap perubahan, lihat Catatan rilis Redis 2.8.

- Mulai Redis 2.8.22, Redis AOF tidak lagi didukung di ElastiCache for Redis. Sebaiknya gunakan MemoryDB jika data perlu dipertahankan tahan lama.
- Mulai Redis 2.8.22, ElastiCache for Redis tidak lagi mendukung pelampiran replika ke primer yang dihosting dalam ElastiCache. Saat melakukan peningkatan, replika eksternal akan terputus dan tidak akan dapat terhubung kembali. Sebaiknya gunakan caching sisi klien, yang tersedia di Redis 6.0 sebagai alternatif replika eksternal.
- Perintah `TTL` dan `PTTL` sekarang mengembalikan -2 jika kunci tidak ada dan -1 jika ada tetapi tidak memiliki kedaluwarsa terkait. Redis 2.6 dan versi sebelumnya digunakan untuk mengembalikan -1 untuk kedua kondisi.
- `SORT` dengan `ALPHA` sekarang melakukan pengurutan berdasarkan lokal pengumpulan lokal jika tidak ada opsi `STORE` yang digunakan.

Untuk informasi selengkapnya, lihat Jadwal akhir masa pakai versi Redis.

# Praktik terbaik ElastiCache dan strategi caching

Di bawah ini, Anda dapat menemukan praktik terbaik yang direkomendasikan untuk Amazon ElastiCache. Mengikuti langkah ini akan meningkatkan performa dan keandalan cache Anda.

Topik

- Bekerja dengan Redis
- Praktik terbaik dengan klien Redis
- Praktik terbaik saat bekerja dengan klaster yang dirancang sendiri
- Praktik terbaik Redis
- Strategi Pembuatan Cache

## Bekerja dengan Redis

Di bawah ini, Anda dapat menemukan informasi tentang antarmuka Redis dalam ElastiCache.

Topik

- Perintah Redis yang didukung dan dibatasi
- Konfigurasi dan batas Redis

# Perintah Redis yang didukung dan dibatasi

Perintah Redis yang didukung

Perintah Redis yang didukung

Perintah Redis berikut didukung oleh cache nirserver. Selain perintah ini, Perintah JSON Redis yang didukung ini juga didukung.

Perintah Bitmap

- `BITCOUNT`

  Menghitung jumlah bit set (penghitungan populasi) dalam string.

  Pelajari selengkapnya
- `BITFIELD`

  Melakukan operasi bilangan bulat bitfield arbitrer pada string.

  Pelajari selengkapnya
- `BITFIELD_RO`

  Melakukan operasi bilangan bulat bitfield hanya-baca arbitrer pada string.

  Pelajari selengkapnya
- `BITOP`

  Melakukan operasi bitwise pada beberapa string, dan menyimpan hasilnya.

  Pelajari selengkapnya
- `BITPOS`

  Menemukan set pertama (1) atau menghapus (0) bit dalam string.

  Pelajari selengkapnya
- `GETBIT`

  Mengembalikan nilai bit dengan offset.

  Pelajari selengkapnya

- `SETBIT`

  Menetapkan atau menghapus bit pada offset dari nilai string. Membuat kunci jika tidak ada.

  Pelajari selengkapnya

Perintah Manajemen Klaster

- `CLUSTER COUNTKEYSINSLOT`

  Mengembalikan jumlah kunci dalam slot hash.

  Pelajari selengkapnya
- `CLUSTER GETKEYSINSLOT`

  Mengembalikan nama kunci dalam slot hash.

  Pelajari selengkapnya
- `CLUSTER INFO`

  Mengembalikan informasi tentang keadaan simpul. Dalam cache nirserver, mengembalikan status tentang “serpihan” virtual tunggal yang diekspos ke klien.

  Pelajari selengkapnya
- `CLUSTER KEYSLOT`

  Mengembalikan slot hash untuk kunci.

  Pelajari selengkapnya
- `CLUSTER MYID`

  Mengembalikan ID simpul. Dalam cache nirserver, mengembalikan status tentang “serpihan” virtual tunggal yang diekspos ke klien.

  Pelajari selengkapnya
- `CLUSTER NODES`

  Mengembalikan konfigurasi klaster untuk simpul. Dalam cache nirserver, mengembalikan status tentang “serpihan” virtual tunggal yang diekspos ke klien.

Pelajari selengkapnya

- `CLUSTER REPLICAS`

  Daftar node replika dari simpul utama. Dalam cache nirserver, mengembalikan status tentang "serpihan" virtual tunggal yang diekspos ke klien.

  Pelajari selengkapnya

- `CLUSTER SHARDS`

  Mengembalikan pemetaan slot klaster ke serpihan. Dalam cache nirserver, mengembalikan status tentang "serpihan" virtual tunggal yang diekspos ke klien.

  Pelajari selengkapnya

- `CLUSTER SLOTS`

  Mengembalikan pemetaan slot klaster ke simpul. Dalam cache nirserver, mengembalikan status tentang "serpihan" virtual tunggal yang diekspos ke klien.

  Pelajari selengkapnya

- `READONLY`

  Mengaktifkan kueri hanya-baca untuk koneksi ke simpul replika Redis Cluster.

  Pelajari selengkapnya

- `READWRITE`

  Mengaktifkan kueri baca-tulis untuk koneksi ke simpul replika Redis Cluster.

  Pelajari selengkapnya

Perintah Manajemen Koneksi

- `AUTH`

  Mengautentikasi koneksi.

  Pelajari selengkapnya

- `CLIENT GETNAME`

Mengembalikan nama koneksi.

Pelajari selengkapnya

- `CLIENT REPLY`

  Menginstruksikan server apakah akan membalas perintah.

  Pelajari selengkapnya

- `CLIENT SETNAME`

  Menetapkan nama koneksi.

  Pelajari selengkapnya

- `ECHO`

  Mengembalikan string yang diberikan.

  Pelajari selengkapnya

- `HELLO`

  Melakukan handshake dengan server Redis.

  Pelajari selengkapnya

- `PING`

  Mengembalikan respon keaktifan server.

  Pelajari selengkapnya

- `QUIT`

  Menutup koneksi.

  Pelajari selengkapnya

- `RESET`

  Mereset koneksi.

  Pelajari selengkapnya

- `SELECT`

Mengubah basis data yang dipilih.

Pelajari selengkapnya

Perintah Generik

- `COPY`

  Menyalin nilai kunci ke kunci baru.

  Pelajari selengkapnya
- `DEL`

  Menghapus satu atau beberapa tombol.

  Pelajari selengkapnya
- `DUMP`

  Mengembalikan representasi serial dari nilai yang disimpan pada kunci.

  Pelajari selengkapnya
- `EXISTS`

  Menentukan apakah ada satu kunci atau lebih.

  Pelajari selengkapnya
- `EXPIRE`

  Menetapkan waktu kedaluwarsa kunci dalam hitungan detik.

  Pelajari selengkapnya
- `EXPIREAT`

  Menetapkan waktu kedaluwarsa kunci ke stempel waktu Unix.

  Pelajari selengkapnya
- `EXPIRETIME`

  Mengembalikan waktu kedaluwarsa kunci sebagai stempel waktu Unix.

Pelajari selengkapnya

- `PERSIST`

  Menghapus waktu kedaluwarsa kunci.

  Pelajari selengkapnya

- `PEXPIRE`

  Menetapkan waktu kedaluwarsa kunci dalam milidetik.

  Pelajari selengkapnya

- `PEXPIREAT`

  Menetapkan waktu kedaluwarsa kunci ke stempel waktu milidetik Unix.

  Pelajari selengkapnya

- `PEXPIRETIME`

  Mengembalikan waktu kedaluwarsa kunci sebagai stempel waktu milidetik Unix.

  Pelajari selengkapnya

- `PTTL`

  Mengembalikan waktu kedaluwarsa dalam milidetik kunci.

  Pelajari selengkapnya

- `RANDOMKEY`

  Mengembalikan nama kunci acak dari basis data.

  Pelajari selengkapnya

- `RENAME`

  Mengganti nama kunci dan menimpa tujuan.

  Pelajari selengkapnya

- `RENAMENX`

  Mengganti nama kunci hanya jika nama kunci target tidak ada.

Pelajari selengkapnya

- `RESTORE`

  Membuat kunci dari representasi serial nilai.

  Pelajari selengkapnya

- `SCAN`

  Melakukan iterasi pada nama kunci dalam basis data.

  Pelajari selengkapnya

- `SORT`

  Mengurutkan elemen dalam daftar, set, atau set yang diurutkan, secara opsional menyimpan hasilnya.

  Pelajari selengkapnya

- `SORT_RO`

  Mengembalikan elemen diurutkan dari daftar, set, atau urutan set.

  Pelajari selengkapnya

- `TOUCH`

  Mengembalikan jumlah kunci yang ada dari yang ditentukan setelah memperbarui waktu kunci tersebut terakhir diakses.

  Pelajari selengkapnya

- `TTL`

  Mengembalikan waktu kedaluwarsa dalam detik kunci.

  Pelajari selengkapnya

- `TYPE`

  Menentukan jenis nilai yang disimpan pada kunci.

  Pelajari selengkapnya

- `UNLINK`

Menghapus satu kunci atau lebih secara asinkron.

Pelajari selengkapnya

Perintah Geospasial

- `GEOADD`

  Menambahkan satu atau beberapa anggota ke indeks geospasial. Kunci dibuat jika tidak ada.

  Pelajari selengkapnya
- `GEODIST`

  Mengembalikan jarak antara dua anggota indeks geospasial.

  Pelajari selengkapnya
- `GEOHASH`

  Mengembalikan anggota dari indeks geospasial sebagai string geohash.

  Pelajari selengkapnya
- `GEOPOS`

  Mengembalikan bujur dan lintang anggota dari indeks geospasial.

  Pelajari selengkapnya
- `GEORADIUS`

  Meminta indeks geospasial untuk anggota dalam jarak dari koordinat, secara opsional menyimpan hasilnya.

  Pelajari selengkapnya
- `GEORADIUS_RO`

  Mengembalikan anggota dari indeks geospasial yang berada dalam jarak dari koordinat.

  Pelajari selengkapnya
- `GEORADIUSBYMEMBER`

Meminta indeks geospasial untuk anggota dalam jarak dari anggota, secara opsional menyimpan hasilnya.

Pelajari selengkapnya

- `GEORADIUSBYMEMBER_RO`

  Mengembalikan anggota dari indeks geospasial yang berada dalam jarak dari anggota.

  Pelajari selengkapnya

- `GEOSEARCH`

  Kueri indeks geospasial untuk anggota di dalam area kotak atau lingkaran.

  Pelajari selengkapnya

- `GEOSEARCHSTORE`

  Kueri indeks geospasial untuk anggota di dalam area kotak atau lingkaran, secara opsional menyimpan hasil.

  Pelajari selengkapnya

Perintah Hash

- `HDEL`

  Menghapus satu atau beberapa bidang dan nilainya dari hash. Menghapus hash jika tidak ada bidang yang tersisa.

  Pelajari selengkapnya

- `HEXISTS`

  Menentukan apakah bidang ada dalam hash.

  Pelajari selengkapnya

- `HGET`

  Mengembalikan nilai bidang dalam hash.

  Pelajari selengkapnya

- `HGETALL`

  Mengembalikan semua bidang dan nilai dalam hash.

  Pelajari selengkapnya

- `HINCRBY`

  Menambah nilai bilangan bulat dari bidang dalam hash dengan angka. Menggunakan 0 sebagai nilai awal jika bidang tidak ada.

  Pelajari selengkapnya

- `HINCRBYFLOAT`

  Menambah nilai titik mengambang dari bidang dengan angka. Menggunakan 0 sebagai nilai awal jika bidang tidak ada.

  Pelajari selengkapnya

- `HKEYS`

  Mengembalikan semua bidang dalam hash.

  Pelajari selengkapnya

- `HLEN`

  Mengembalikan jumlah bidang dalam hash.

  Pelajari selengkapnya

- `HMGET`

  Mengembalikan nilai-nilai dari semua bidang dalam hash.

  Pelajari selengkapnya

- `HMSET`

  Menetapkan nilai dari beberapa bidang.

  Pelajari selengkapnya

- `HRANDFIELD`

  Mengembalikan satu atau beberapa bidang acak dari hash.

Pelajari selengkapnya

- HSCAN

  Melakukan iterasi pada bidang dan nilai hash.

  Pelajari selengkapnya

- HSET

  Membuat atau mengubah nilai bidang dalam hash.

  Pelajari selengkapnya

- HSETNX

  Menetapkan nilai bidang dalam hash hanya jika bidang tidak ada.

  Pelajari selengkapnya

- HSTRLEN

  Mengembalikan panjang nilai bidang.

  Pelajari selengkapnya

- HVALS

  Mengembalikan semua nilai dalam hash.

  Pelajari selengkapnya

Perintah HyperLogLog

- PFADD

  Menambahkan elemen ke kunci HyperLogLog. Membuat kunci jika tidak ada.

  Pelajari selengkapnya

- PFCOUNT

  Mengembalikan perkiraan kardinalitas himpunan yang diamati oleh kunci HyperLogLog.

  Pelajari selengkapnya

- PFMERGE

  Menggabungkan satu atau beberapa nilai HyperLogLog ke dalam satu kunci.

  Pelajari selengkapnya

Perintah List

- BLMOVE

  Memunculkan elemen dari daftar, mendorongnya ke daftar lain, dan mengembalikannya. Memblokir sampai elemen tersedia sebaliknya. Menghapus daftar jika elemen terakhir dipindahkan.

  Pelajari selengkapnya

- BLMPOP

  Memunculkan elemen pertama dari salah satu dari beberapa daftar. Memblokir sampai elemen tersedia sebaliknya. Menghapus daftar jika elemen terakhir muncul.

  Pelajari selengkapnya

- BLPOP

  Menghapus dan mengembalikan elemen pertama dalam daftar. Memblokir sampai elemen tersedia sebaliknya. Menghapus daftar jika elemen terakhir muncul.

  Pelajari selengkapnya

- BRPOP

  Menghapus dan mengembalikan elemen terakhir dalam daftar. Memblokir sampai elemen tersedia sebaliknya. Menghapus daftar jika elemen terakhir muncul.

  Pelajari selengkapnya

- BRPOPLPUSH

  Memunculkan elemen dari daftar, mendorongnya ke daftar lain, dan mengembalikannya. Memblokir sampai elemen tersedia sebaliknya. Menghapus daftar jika elemen terakhir muncul.

  Pelajari selengkapnya

- `LINDEX`

  Mengembalikan elemen dari daftar dengan indeks.

  Pelajari selengkapnya

- `LINSERT`

  Menyisipkan elemen sebelum atau sesudah elemen lain dalam daftar.

  Pelajari selengkapnya

- `LLEN`

  Mengembalikan panjang daftar.

  Pelajari selengkapnya

- `LMOVE`

  Mengembalikan elemen setelah muncul dari satu daftar dan mendorongnya ke yang lain. Menghapus daftar jika elemen terakhir dipindahkan.

  Pelajari selengkapnya

- `LMPOP`

  Mengembalikan beberapa elemen dari daftar setelah menghapusnya. Menghapus daftar jika elemen terakhir muncul.

  Pelajari selengkapnya

- `LPOP`

  Mengembalikan elemen pertama dalam daftar setelah menghapusnya. Menghapus daftar jika elemen terakhir muncul.

  Pelajari selengkapnya

- `LPOS`

  Mengembalikan indeks elemen yang cocok dalam daftar.

  Pelajari selengkapnya

- `LPUSH`

Menambahkan satu atau beberapa elemen ke daftar. Membuat kunci jika tidak ada.

Pelajari selengkapnya

- `LPUSHX`

  Menambahkan satu atau beberapa elemen ke daftar hanya jika daftar ada.

  Pelajari selengkapnya

- `LRANGE`

  Mengembalikan berbagai elemen dari daftar.

  Pelajari selengkapnya

- `LREM`

  Menghapus elemen dari daftar. Menghapus daftar jika elemen terakhir telah dihapus.

  Pelajari selengkapnya

- `LSET`

  Menetapkan nilai elemen dalam daftar dengan indeksnya.

  Pelajari selengkapnya

- `LTRIM`

  Menghapus elemen dari kedua ujung daftar. Menghapus daftar jika semua elemen dipangkas.

  Pelajari selengkapnya

- `RPOP`

  Mengembalikan dan menghapus elemen terakhir dari daftar. Menghapus daftar jika elemen terakhir muncul.

  Pelajari selengkapnya

- `RPOPLPUSH`

  Mengembalikan elemen terakhir dari daftar setelah menghapus dan mendorongnya ke daftar lain. Menghapus daftar jika elemen terakhir muncul.

Pelajari selengkapnya

- RPUSH

  Menambahkan satu atau beberapa elemen ke daftar. Membuat kunci jika tidak ada.

  Pelajari selengkapnya

- RPUSHX

  Menambahkan elemen ke daftar hanya jika daftar ada.

  Pelajari selengkapnya

Perintah Pub/Sub

> **Note**
>
> Perintah PUBSUB secara internal menggunakan PUBSUB serpihan, sehingga nama saluran akan dicampur.

- PUBLISH

  Memposting pesan ke saluran.

  Pelajari selengkapnya

- PUBSUB CHANNELS

  Mengembalikan saluran yang aktif.

  Pelajari selengkapnya

- PUBSUB NUMSUB

  Mengembalikan jumlah pelanggan ke saluran.

  Pelajari selengkapnya

- PUBSUB SHARDCHANNELS

  Mengembalikan saluran serpihan aktif.

PUBSUB-SHARDCHANNELS

- `PUBSUB SHARDNUMSUB`

  Mengembalikan jumlah pelanggan saluran serpihan.

  PUBSUB-SHARDNUMSUB

- `SPUBLISH`

  Memposting pesan ke saluran serpihan

  Pelajari selengkapnya

- `SSUBSCRIBE`

  Mendengarkan pesan yang dipublikasikan ke saluran serpihan.

  Pelajari selengkapnya

- `SUBSCRIBE`

  Mendengarkan pesan yang dipublikasikan ke saluran.

  Pelajari selengkapnya

- `SUNSUBSCRIBE`

  Berhenti mendengarkan pesan yang diposting ke saluran serpihan.

  Pelajari selengkapnya

- `UNSUBSCRIBE`

  Berhenti mendengarkan pesan yang diposting ke saluran.

  Pelajari selengkapnya

### Perintah Scripting

- `EVAL`

  Mengeksekusi skrip Lua sisi server.

  Pelajari selengkapnya

- `EVAL_RO`

  Mengeksekusi skrip Lua sisi server hanya-baca.

  Pelajari selengkapnya

- `EVALSHA`

  Mengeksekusi skrip Lua sisi server oleh digest SHA1.

  Pelajari selengkapnya

- `EVALSHA_RO`

  Mengeksekusi skrip Lua sisi server hanya-baca oleh digest SHA1.

  Pelajari selengkapnya

- `SCRIPT EXISTS`

  Menentukan apakah skrip Lua sisi server ada di cache skrip.

  Pelajari selengkapnya

- `SCRIPT FLUSH`

  Saat ini cache skrip no-op dikelola oleh layanan.

  Pelajari selengkapnya

- `SCRIPT LOAD`

  Memuat skrip Lua sisi server ke cache skrip.

  Pelajari selengkapnya

Perintah Manajemen Server

- `ACL CAT`

  Daftar kategori ACL, atau perintah di dalam kategori.

  Pelajari selengkapnya

- `ACL GENPASS`

Menghasilkan pseudorandom, kata sandi aman yang dapat digunakan untuk mengidentifikasi pengguna ACL.

Pelajari selengkapnya

- `ACL GETUSER`

  Menampilkan daftar aturan ACL pengguna.

  Pelajari selengkapnya

- `ACL LIST`

  Membuang aturan efektif dalam format file ACL.

  Pelajari selengkapnya

- `ACL USERS`

  Menampilkan daftar semua pengguna ACL.

  Pelajari selengkapnya

- `ACL WHOAMI`

  Mengembalikan nama pengguna yang diautentikasi koneksi saat ini.

  Pelajari selengkapnya

- `DBSIZE`

  Mengembalikan jumlah kunci dalam basis data yang dipilih saat ini. Operasi ini tidak dijamin atom di semua slot.

  Pelajari selengkapnya

- `COMMAND`

  Mengembalikan informasi mendetail tentang semua perintah.

  Pelajari selengkapnya

- `COMMAND COUNT`

  Mengembalikan hitungan perintah.

Pelajari selengkapnya

- `COMMAND DOCS`

  Mengembalikan informasi dokumenter tentang satu, beberapa, atau semua perintah.

  Pelajari selengkapnya

- `COMMAND GETKEYS`

  Mengekstrak nama kunci dari perintah arbitrer.

  Pelajari selengkapnya

- `COMMAND GETKEYSANDFLAGS`

  Mengekstrak nama kunci dan bendera akses untuk perintah arbitrer.

  Pelajari selengkapnya

- `COMMAND INFO`

  Mengembalikan informasi tentang satu, beberapa, atau semua perintah.

  Pelajari selengkapnya

- `COMMAND LIST`

  Mengembalikan daftar nama perintah.

  Pelajari selengkapnya

- `FLUSHALL`

  Menghapus semua kunci dari semua basis data. Operasi ini tidak dijamin atom di semua slot.

  Pelajari selengkapnya

- `FLUSHDB`

  Menghapus semua kunci dari basis data saat ini. Operasi ini tidak dijamin atom di semua slot.

  Pelajari selengkapnya

- `INFO`

  Mengembalikan informasi dan statistik tentang server.

Pelajari selengkapnya

- `LOLWUT`

  Menampilkan seni komputer dan versi Redis.

  Pelajari selengkapnya

- `ROLE`

  Mengembalikan peran replikasi.

  Pelajari selengkapnya

- `TIME`

  Mengembalikan waktu server.

  Pelajari selengkapnya

Perintah Set

- `SADD`

  Menambahkan satu atau beberapa anggota ke set. Membuat kunci jika tidak ada.

  Pelajari selengkapnya

- `SCARDT`

  Mengembalikan jumlah anggota dalam satu set.

  Pelajari selengkapnya

- `SDIFF`

  Mengembalikan perbedaan beberapa set.

  Pelajari selengkapnya

- `SDIFFSTORE`

  Menyimpan perbedaan beberapa set dalam kunci.

  Pelajari selengkapnya

- `SINTER`

  Mengembalikan potongan dari beberapa set.

  Pelajari selengkapnya

- `SINTERCARD`

  Mengembalikan jumlah anggota potongan dari beberapa set.

  Pelajari selengkapnya

- `SINTERSTORE`

  Menyimpan potongan beberapa set dalam kunci.

  Pelajari selengkapnya

- `SISMEMBER`

  Menentukan apakah anggota termasuk dalam set.

  Pelajari selengkapnya

- `SMEMBERS`

  Mengembalikan semua anggota dari satu set.

  Pelajari selengkapnya

- `SMISMEMBER`

  Menentukan apakah beberapa anggota termasuk dalam set.

  Pelajari selengkapnya

- `SMOVE`

  Memindahkan anggota dari satu set ke set lainnya.

  Pelajari selengkapnya

- `SPOP`

  Mengembalikan satu atau beberapa anggota acak dari satu set setelah menghapusnya. Menghapus set jika anggota terakhir muncul.

Pelajari selengkapnya

- SRANDMEMBER

  Mendapatkan satu atau beberapa anggota acak dari satu set

  Pelajari selengkapnya

- SREM

  Menghapus satu atau beberapa anggota dari satu set. Menghapus set jika anggota terakhir telah dihapus.

  Pelajari selengkapnya

- SSCAN

  Melakukan iterasi pada anggota set.

  Pelajari selengkapnya

- SUNION

  Mengembalikan gabungan dari beberapa set.

  Pelajari selengkapnya

- SUNIONSTORE

  Menyimpan gabungan beberapa set dalam kunci.

  Pelajari selengkapnya

Perintah Sorted Set

- BZMPOP

  Menghapus dan mengembalikan anggota berdasarkan skor dari satu atau lebih urutan set. Memblokir sampai anggota tersedia sebaliknya. Menghapus urutan daftar jika elemen terakhir muncul.

  Pelajari selengkapnya

- BZPOPMAX

Menghapus dan mengembalikan anggota dengan skor tertinggi dari satu atau beberapa urutan set. Memblokir sampai anggota tersedia sebaliknya. Menghapus urutan daftar jika elemen terakhir muncul.

Pelajari selengkapnya

- `BZPOPMIN`

  Menghapus dan mengembalikan anggota dengan skor terendah dari satu atau beberapa urutan set. Memblokir sampai anggota tersedia sebaliknya. Menghapus urutan daftar jika elemen terakhir muncul.

  Pelajari selengkapnya

- `ZADD`

  Menambahkan satu atau lebih anggota ke urutan set, atau memperbarui skornya. Membuat kunci jika tidak ada.

  Pelajari selengkapnya

- `ZCARD`

  Mengembalikan jumlah anggota dalam satu urutan set.

  Pelajari selengkapnya

- `ZCOUNT`

  Mengembalikan jumlah anggota dalam satu urutan set yang memiliki skor dalam rentang.

  Pelajari selengkapnya

- `ZDIFF`

  Mengembalikan perbedaan antara beberapa urutan set.

  Pelajari selengkapnya

- `ZDIFFSTORE`

  Menyimpan perbedaan beberapa urutan set dalam kunci.

  Pelajari selengkapnya

- `ZINCRBY`

Meningkatkan skor anggota dalam urutan set.

Pelajari selengkapnya

- `ZINTER`

  Mengembalikan potongan dari beberapa urutan set.

  Pelajari selengkapnya

- `ZINTERCARD`

  Mengembalikan jumlah anggota potongan dari beberapa urutan set.

  Pelajari selengkapnya

- `ZINTERSTORE`

  Menyimpan potongan beberapa urutan set dalam kunci.

  Pelajari selengkapnya

- `ZLEXCOUNT`

  Mengembalikan jumlah anggota dalam urutan set dalam rentang leksikografis.

  Pelajari selengkapnya

- `ZMPOP`

  Mengembalikan anggota dengan skor tertinggi atau terendah dari satu atau lebih urutan set setelah menghapusnya. Menghapus urutan set jika anggota terakhir muncul.

  Pelajari selengkapnya

- `ZMSCORE`

  Mengembalikan skor dari satu atau lebih anggota dalam satu urutan set.

  Pelajari selengkapnya

- `ZPOPMAX`

  Mengembalikan anggota dengan skor tertinggi dari satu urutan set setelah menghapusnya. Menghapus urutan set jika anggota terakhir muncul.

Pelajari selengkapnya

- ZPOPMIN

  Mengembalikan anggota dengan skor terendah dari satu urutan set setelah menghapusnya. Menghapus urutan set jika anggota terakhir muncul.

  Pelajari selengkapnya

- ZRANDMEMBER

  Mengembalikan satu atau lebih anggota acak dari urutan set.

  Pelajari selengkapnya

- ZRANGE

  Mengembalikan anggota dalam satu urutan set dalam rentang indeks.

  Pelajari selengkapnya

- ZRANGEBYLEX

  Mengembalikan anggota dalam urutan set dalam rentang leksikografis.

  Pelajari selengkapnya

- ZRANGEBYSCORE

  Mengembalikan anggota dalam satu urutan set dalam rentang skor.

  Pelajari selengkapnya

- ZRANGESTORE

  Menyimpan rentang anggota dari urutan set dalam kunci.

  Pelajari selengkapnya

- ZRANK

  Mengembalikan indeks anggota dalam urutan set yang diurutkan berdasarkan skor naik.

  Pelajari selengkapnya

- ZREM

Menghapus satu atau beberapa anggota dari urutan set. Menghapus urutan set jika semua anggota telah dihapus.

Pelajari selengkapnya

- `ZREMRANGEBYLEX`

  Mengembalikan anggota dalam urutan set dalam rentang leksikografis. Menghapus urutan set jika semua anggota telah dihapus.

  Pelajari selengkapnya

- `ZREMRANGEBYRANK`

  Mengembalikan anggota dalam satu urutan set dalam rentang indeks. Menghapus urutan set jika semua anggota telah dihapus.

  Pelajari selengkapnya

- `ZREMRANGEBYSCORE`

  Mengembalikan anggota dalam satu urutan set dalam rentang skor. Menghapus urutan set jika semua anggota telah dihapus.

  Pelajari selengkapnya

- `ZREVRANGE`

  Mengembalikan anggota dalam satu urutan set dalam rentang indeks dalam urutan mundur.

  Pelajari selengkapnya

- `ZREVRANGEBYLEX`

  Mengembalikan anggota dalam urutan set rentang leksikografis dalam urutan mundur.

  Pelajari selengkapnya

- `ZREVRANGEBYSCORE`

  Mengembalikan anggota dalam satu urutan set dalam rentang skor dalam urutan mundur.

  Pelajari selengkapnya

- `ZREVRANK`

Mengembalikan indeks anggota dalam urutan set yang diurutkan berdasarkan skor menurun.

Pelajari selengkapnya

- ZSCAN

  Melakukan iterasi atas anggota dan skor dari urutan set.

  Pelajari selengkapnya

- ZSCORE

  Mengembalikan skor anggota dalam urutan set.

  Pelajari selengkapnya

- ZUNION

  Mengembalikan gabungan dari beberapa urutan set.

  Pelajari selengkapnya

- ZUNIONSTORE

  Menyimpan gabungan beberapa urutan set dalam kunci.

  Pelajari selengkapnya

Perintah Stream

- XACK

  Mengembalikan jumlah pesan yang berhasil diakui oleh anggota grup konsumen aliran.

  Pelajari selengkapnya

- XADD

  Menambahkan pesan baru ke aliran. Membuat kunci jika tidak ada.

  Pelajari selengkapnya

- XAUTOCLAIM

Mengubah, atau memperoleh, kepemilikan pesan dalam grup konsumen, seolah-olah pesan yang telah dikirimkan berasal dari anggota grup konsumen.

Pelajari selengkapnya

- `XCLAIM`

  Mengubah, atau memperoleh, kepemilikan pesan dalam grup konsumen, seolah-olah pesan yang telah terkirim anggota grup konsumen.

  Pelajari selengkapnya

- `XDEL`

  Mengembalikan jumlah pesan setelah menghapusnya dari aliran.

  Pelajari selengkapnya

- `XGROUP CREATE`

  Membuat grup konsumen.

  Pelajari selengkapnya

- `XGROUP CREATECONSUMER`

  Membuat konsumen dalam grup konsumen.

  Pelajari selengkapnya

- `XGROUP DELCONSUMER`

  Menghapus konsumen dari grup konsumen.

  Pelajari selengkapnya

- `XGROUP DESTROY`

  Menghancurkan grup konsumen.

  Pelajari selengkapnya

- `XGROUP SETID`

  Menetapkan ID terakhir yang dikirimkan dari grup konsumen.

Pelajari selengkapnya

- `XINFO CONSUMERS`

  Mengembalikan daftar konsumen dalam grup konsumen.

  Pelajari selengkapnya

- `XINFO GROUPS`

  Mengembalikan daftar grup konsumen dari aliran.

  Pelajari selengkapnya

- `XINFO STREAM`

  Mengembalikan informasi tentang aliran.

  Pelajari selengkapnya

- `XLEN`

  Mengembalikan jumlah pesan dalam aliran.

  Pelajari selengkapnya

- `XPENDING`

  Mengembalikan informasi dan entri dari daftar entri tertunda grup konsumen aliran.

  Pelajari selengkapnya

- `XRANGE`

  Mengembalikan pesan dari aliran dalam rentang ID.

  Pelajari selengkapnya

- `XREAD`

  Mengembalikan pesan dari beberapa aliran dengan ID yang lebih besar dari yang diminta. Blok sampai pesan tersedia sebaliknya.

  Pelajari selengkapnya

- `XREADGROUP`

Mengembalikan pesan baru atau historis dari aliran untuk konsumen dalam grup. Blok sampai pesan tersedia sebaliknya.

Pelajari selengkapnya

- XREVRANGE

  Mengembalikan pesan dari aliran dalam rentang ID dengan urutan mundur.

  Pelajari selengkapnya

- XTRIM

  Menghapus pesan dari awal aliran.

  Pelajari selengkapnya

Perintah String

- APPEND

  Menambahkan string ke nilai kunci. Membuat kunci jika tidak ada.

  Pelajari selengkapnya

- DECR

  Mengurangi nilai bilangan bulat dari kunci satu nilai. Menggunakan 0 sebagai nilai awal jika kunci tidak ada.

  Pelajari selengkapnya

- DECRBY

  Penurunan angka dari nilai bilangan bulat kunci. Menggunakan 0 sebagai nilai awal jika kunci tidak ada.

  Pelajari selengkapnya

- GET

  Mengembalikan nilai string dari kunci.

  Pelajari selengkapnya

- `GETDEL`

  Mengembalikan nilai string kunci setelah menghapus kunci.

  Pelajari selengkapnya

- `GETEX`

  Mengembalikan nilai string kunci setelah mengatur waktu kedaluwarsa.

  Pelajari selengkapnya

- `GETRANGE`

  Mengembalikan substring dari string yang disimpan pada kunci.

  Pelajari selengkapnya

- `GETSET`

  Mengembalikan nilai string sebelumnya dari kunci setelah mengaturnya ke nilai baru.

  Pelajari selengkapnya

- `INCR`

  Menambah nilai bilangan bulat dari kunci satu nilai. Menggunakan 0 sebagai nilai awal jika kunci tidak ada.

  Pelajari selengkapnya

- `INCRBY`

  Menambah nilai bilangan bulat dari kunci dengan angka. Menggunakan 0 sebagai nilai awal jika kunci tidak ada.

  Pelajari selengkapnya

- `INCRBYFLOAT`

  Menambah nilai titik ambang dari bidang dengan angka. Menggunakan 0 sebagai nilai awal jika kunci tidak ada.

  Pelajari selengkapnya

- `LCS`

Menemukan substring umum terpanjang.

Pelajari selengkapnya

- `MGET`

  Secara atom mengembalikan nilai string dari satu atau lebih kunci.

  Pelajari selengkapnya

- `MSET`

  Secara atom membuat atau mengubah nilai string dari satu atau lebih kunci.

  Pelajari selengkapnya

- `MSETNX`

  Secara atom mengubah nilai string dari satu atau lebih kunci hanya ketika semua kunci tidak ada.

  Pelajari selengkapnya

- `PSETEX`

  Menetapkan nilai string dan waktu kedaluwarsa kunci dalam milidetik. Kunci dibuat jika tidak ada.

  Pelajari selengkapnya

- `SET`

  Menetapkan nilai string kunci, mengabaikan tipenya. Kunci dibuat jika tidak ada.

  Pelajari selengkapnya

- `SETEX`

  Menetapkan nilai string dan waktu kedaluwarsa kunci. Membuat kunci jika tidak ada.

  Pelajari selengkapnya

- `SETNX`

  Menetapkan nilai string kunci hanya jika kunci tidak ada.

  Pelajari selengkapnya

- `SETRANGE`

Menimpa bagian dari nilai string dengan yang lain dengan offset. Membuat kunci jika tidak ada.

Pelajari selengkapnya

- `STRLEN`

  Mengembalikan panjang nilai string.

  Pelajari selengkapnya

- `SUBSTR`

  Mengembalikan substring dari nilai string.

  Pelajari selengkapnya

### Perintah Transaction

- `DISCARD`

  Membuang transaksi.

  Pelajari selengkapnya

- `EXEC`

  Mengeksekusi semua perintah dalam transaksi.

  Pelajari selengkapnya

- `MULTI`

  Memulai transaksi.

  Pelajari selengkapnya

### Perintah Redis Terbatas

Untuk memberikan pengalaman layanan terkelola, ElastiCache membatasi akses ke perintah khusus mesin cache tertentu yang memerlukan hak istimewa lanjutan. Untuk cache yang menjalankan Redis, perintah berikut tidak tersedia:

- `acl setuser`

- acl load
- acl save
- acl deluser
- bgrewriteaof
- bgsave
- cluster addslot
- cluster addslotsrange
- cluster bumpepoch
- cluster delslot
- cluster delslotsrange
- cluster failover
- cluster flushslots
- cluster forget
- cluster links
- cluster meet
- cluster setslot
- config
- debug
- migrate
- psync
- replicaof
- save
- slaveof
- shutdown
- sync

Selain itu, perintah berikut tidak tersedia untuk cache nirserver:

- acl log
- client caching
- client getredir

- client id
- client info
- client kill
- client list
- client no-evict
- client pause
- client tracking
- client trackinginfo
- client unblock
- client unpause
- cluster count-failure-reports
- fcall
- fcall_ro
- function
- function delete
- function dump
- function flush
- function help
- function kill
- function list
- function load
- function restore
- function stats
- keys
- lastsave
- latency
- latency doctor
- latency graph
- latency help
- latency histogram

- latency history
- latency latest
- latency reset
- memory
- memory doctor
- memory help
- memory malloc-stats
- memory purge
- memory stats
- memory usage
- monitor
- move
- object
- object encoding
- object freq
- object help
- object idletime
- object refcount
- pfdebug
- pfselftest
- psubscribe
- pubsub numpat
- punsubscribe
- script kill
- slowlog
- slowlog get
- slowlog help
- slowlog len
- slowlog reset
- swapdb

- `unwatch`
- `wait`
- `watch`

## Konfigurasi dan batas Redis

Mesin Redis menyediakan sejumlah parameter konfigurasi, beberapa di antaranya dapat dimodifikasi di ElastiCache for Redis dan beberapa di antaranya tidak dapat dimodifikasi untuk memberikan kinerja dan keandalan yang stabil.

Cache nirserver

Untuk cache nirserver, grup parameter tidak digunakan dan semua konfigurasi Redis tidak dapat dimodifikasi. Parameter Redis berikut sudah ada:

| Nama | Detail | Deskripsi |
|---|---|---|
| `acl-pubsub-default` | `allchannels` | Izin saluran pubsub default untuk pengguna ACL di cache. |
| `client-output-buffer-limit` | `normal 0 0 0`<br>`pubsub 32mb 8mb 60` | Klien normal tidak memiliki batas buffer. Klien PUB/SUB akan terputus jika mereka melanggar backlog 32MiB, atau melanggar backlog 8MiB selama 60 detik. |
| `client-query-buffer-limit` | 1 GiB | Ukuran maksimum buffer kueri klien tunggal. Selain itu, klien tidak dapat mengeluarkan permintaan dengan lebih dari 4.000 argumen. |
| `cluster-allow-pubsubshard-when-down` | `yes` | Hal ini memungkinkan cache untuk melayani lalu lintas pubsub saat sebagian cache tidak aktif. |
| `cluster-allow-read` | `yes` | Ini memungkinkan cache untuk melayani lalu lintas baca saat sebagian cache tidak aktif. |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `s-when-down` | | |
| `cluster-enabled` | `yes` | Semua cache nirserver memiliki mode klaster diaktifkan, yang memungkinkan mereka untuk secara transparan mempartisi data di beberapa serpihan backend. Semua slot dimunculkan ke klien karena dimiliki oleh satu simpul virtual. |
| `cluster-require-full-coverage` | `no` | Ketika sebagian keyspace tidak aktif (setidakn ya satu slot hash tidak dapat diakses), cache akan terus menerima kueri untuk bagian ruang kunci yang masih tercakup. Seluruh ruang kunci akan selalu “dicakup” oleh satu simpul virtual di`cluster slots`. |
| `lua-time-limit` | 5000 | Waktu eksekusi maksimum untuk skrip Lua, dalam milidetik, sebelum ElastiCache mengambil tindakan untuk menghentikan skrip.<br><br>Jika `lua-time-limit` terlampaui, semua perintah Redis akan mengembalikan kesalahan dalam bentuk ____-BUSY. Karena keadaan ini dapat menyebabkan gangguan terhadap banyak operasi penting Redis, ElastiCache akan terlebih dahulu mengeluarkan perintah SCRIPT KILL. Jika tindakan ini tidak berhasil, ElastiCache akan memulai ulang Redis secara paksa. |
| `maxclients` | 65000 | Jumlah maksimum klien yang dapat dihubungk an pada satu waktu. Koneksi yang dibuat setelahnya dapat berhasil atau tidak berhasil. |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `maxmemory-policy` | `volatile-lru` | Item dengan set TTL dikosongkan mengikuti estimasi yang paling lama tidak digunakan (LRU) saat batas memori cache tercapai. |
| `notify-keyspace-events` | (sebuah string kosong) | Peristiwa keyspace saat ini tidak didukung pada cache nirserver. |
| `port` | Port primer: 6379<br>Port baca: 6380 | Cache nirserver menyatakan dua port dengan nama host yang sama. Port primer memungkinkan operasi baca dan tulis, sedangkan port baca memungkinkan bacaan akhir konsisten berlatensi rendah menggunakan perintah `READONLY`. |
| `proto-max-bulk-len` | 512 MiB | Ukuran maksimum dari permintaan elemen tunggal. |
| `timeout` | 0 | Klien tidak terputus secara paksa pada waktu idle tertentu, tetapi mungkin terputus selama stabil untuk demi tujuan penyeimbangan beban. |

Selain itu, berlaku batasan berikut:

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| Panjang nama kunci | 4 KiB | Ukuran maksimum untuk satu kunci atau nama saluran Redis. Kunci yang mereferensikan kunci lebih besar dari ini akan menerima kesalahan. |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| Ukuran skrip Lua | 4MiB | Ukuran maksimum satu skrip Redis Lua. Upaya untuk memuat skrip Lua yang lebih besar dari ini akan menerima kesalahan. |
| Ukuran slot | 32 GiB | Ukuran maksimum satu slot hash Redis. Klien yang mencoba mengatur data lebih banyak dari ini pada satu slot Redis akan memicu kebijakan pengosongan pada slot, dan jika tidak ada kunci yang dapat dikosongkan, akan menerima kesalahan memori habis (`OOM`). |

Klaster yang dirancang sendiri

Untuk cluster yang dirancang sendiri, lihat Parameter spesifik Redis untuk nilai default parameter konfigurasi dan mana yang dapat dikonfigurasi. Nilai default umumnya direkomendasikan kecuali Anda memiliki kasus penggunaan khusus yang mengharuskan penimpaan.

## Praktik terbaik dengan klien Redis

Pelajari praktik terbaik untuk skenario umum dan contoh kode dari beberapa pustaka klien Redis sumber terbuka paling populer (redis-py, PhPredis, dan Lettuce).

Topik

- Koneksi dalam jumlah besar
- Penemuan klien klaster Redis dan mundur eksponensial
- Mengonfigurasi batas waktu sisi klien
- Mengonfigurasi batas waktu idle sisi server
- Skrip Redis
- Menyimpan item komposit besar
- Konfigurasi klien Lettuce
- Contoh klien IPv6

## Koneksi dalam jumlah besar

Cache nirserver dan simpul ElastiCache for Redis individual mendukung hingga 65.000 koneksi klien bersamaan. Namun, untuk mengoptimalkan performa, kami menyarankan agar aplikasi klien tidak terus-menerus beroperasi pada tingkat koneksi tersebut. Redis adalah proses single-threaded berdasarkan loop peristiwa tempat permintaan klien yang masuk ditangani secara berurutan. Artinya, waktu respons klien tertentu menjadi lebih lama karena jumlah klien yang terhubung meningkat.

Anda dapat melakukan serangkaian tindakan berikut untuk menghindari hambatan koneksi di server Redis:

- Lakukan operasi baca dari replika baca. Hal ini dapat dilakukan dengan menggunakan titik akhir pembaca ElastiCache dalam mode klaster yang dinonaktifkan atau dengan menggunakan replika untuk pembacaan dalam mode klaster yang diaktifkan, termasuk cache nirserver.
- Mendistribusikan lalu lintas tulis di beberapa simpul primer. Anda dapat melakukannya dengan dua cara: Anda dapat menggunakan klaster Redis multiserpihan dengan klien dengan dukungan mode klaster Redis. Anda juga dapat menulis ke beberapa simpul primer dalam mode klaster dinonaktifkan dengan serpihan sisi klien. Hal ini dilakukan secara otomatis dalam cache nirserver.
- Gunakan kumpulan koneksi bila tersedia di pustaka klien Anda.

Secara umum, pembuatan koneksi TCP adalah operasi yang mahal secara komputasi dibandingkan dengan perintah Redis secara umum. Misalnya, menangani permintaan SET/GET jauh lebih cepat saat menggunakan kembali koneksi yang ada. Menggunakan kumpulan koneksi klien dengan ukuran terbatas akan mengurangi overhead manajemen koneksi. Hal ini juga membatasi jumlah koneksi masuk bersamaan dari aplikasi klien.

Contoh kode berikut dari PhPredis menunjukkan bahwa koneksi baru dibuat untuk setiap permintaan pengguna baru:

```
$redis = new Redis();
if ($redis->connect($HOST, $PORT) != TRUE) {
 //ERROR: connection failed
 return;
}
$redis->set($key, $value);
unset($redis);
$redis = NULL;
```

Kami membandingkan kode ini di loop pada instans Amazon Elastic Compute Cloud (Amazon EC2) yang terhubung ke simpul Graviton2 (m6g.2xlarge). Kami menempatkan klien dan server di Zona Ketersediaan yang sama. Latensi rata-rata seluruh operasi adalah 2,82 milidetik.

Saat kami memperbarui kode serta menggunakan koneksi persisten dan kumpulan koneksi, latensi rata-rata seluruh operasi adalah 0,21 milidetik:

```
$redis = new Redis();
if ($redis->pconnect($HOST, $PORT) != TRUE) {
 // ERROR: connection failed
 return;
}
$redis->set($key, $value);
unset($redis);
$redis = NULL;
```

Konfigurasi redis.ini yang diperlukan:

- `redis.pconnect.pooling_enabled=1`
- `redis.pconnect.connection_limit=10`

Kode berikut adalah contoh kumpulan koneksi Redis-Py:

```
conn = Redis(connection_pool=redis.BlockingConnectionPool(host=HOST,
 max_connections=10))
conn.set(key, value)
```

Kode berikut adalah contoh kumpulan koneksi Lettuce:

```
RedisClient client = RedisClient.create(RedisURI.create(HOST, PORT));
GenericObjectPool<StatefulRedisConnection> pool =
 ConnectionPoolSupport.createGenericObjectPool(() -> client.connect(), new
 GenericObjectPoolConfig());
pool.setMaxTotal(10); // Configure max connections to 10
try (StatefulRedisConnection connection = pool.borrowObject()) {
 RedisCommands syncCommands = connection.sync();
 syncCommands.set(key, value);
}
```

## Penemuan klien klaster Redis dan mundur eksponensial

Saat menghubungkan ke klaster ElastiCache for Redis dalam mode klaster diaktifkan, pustaka klien Redis yang sesuai harus peka terhadap klaster. Klien harus mendapatkan peta slot hash ke simpul yang sesuai di klaster untuk mengirim permintaan ke simpul yang tepat dan menghindari overhead performa penanganan pengalihan klaster. Akibatnya, klien harus menemukan daftar lengkap slot dan simpul yang dipetakan dalam dua situasi berbeda:

- Klien diinisialisasi dan harus mengisi konfigurasi slot awal
- Pengalihan MOVED diterima dari server, seperti dalam situasi failover ketika semua slot yang dilayani oleh simpul primer sebelumnya diambil alih oleh replika, atau melakukan re-sharding ketika slot dipindahkan dari sumber primer ke simpul primer target

Penemuan klien biasanya dilakukan dengan menjalankan perintah CLUSTER SLOT atau CLUSTER NODE ke server Redis. Sebaiknya gunakan metode SLOT CLUSTER karena metode ini mengembalikan set rentang slot dan simpul primer dan replika terkait ke klien. Metode ini tidak memerlukan parsing tambahan dari klien dan lebih efisien.

Tergantung topologi klaster, ukuran respons untuk perintah SLOT CLUSTER dapat bervariasi berdasarkan ukuran klaster. Klaster yang lebih besar dengan lebih banyak simpul menghasilkan respons yang lebih besar. Sebagai hasilnya, penting untuk memastikan bahwa jumlah klien yang melakukan penemuan topologi klaster tidak bertambah tanpa batas. Misalnya, ketika aplikasi klien memulai atau kehilangan koneksi dari server dan harus melakukan penemuan klaster, satu kesalahan umumnya adalah bahwa aplikasi klien mengaktifkan beberapa permintaan koneksi ulang dan penemuan tanpa menambahkan mundur eksponensial saat mencoba lagi. Hal ini dapat membuat server Redis tidak responsif untuk jangka waktu yang lama, dengan penggunaan CPU 100%. Pemadaman diperpanjang jika setiap perintah CLUSTER SLOT harus memproses sejumlah besar simpul di bus klaster. Kami telah mengamati beberapa pemadaman klien sebelumnya karena perilaku ini di sejumlah bahasa yang berbeda termasuk Python (redis-py-cluster) dan Java (Lettuce dan Redisson).

Dalam cache nirserver, banyak masalah secara otomatis dikurangi karena topologi klaster yang diiklankan bersifat statis dan terdiri dari dua entri: titik akhir tulis dan titik akhir baca. Penemuan klaster juga secara otomatis tersebar di beberapa simpul saat menggunakan titik akhir cache. Namun, rekomendasi berikut masih berguna.

Untuk mengurangi dampak yang disebabkan oleh masuknya permintaan koneksi dan penemuan secara tiba-tiba, kami merekomendasikan hal berikut:

- Menerapkan kumpulan koneksi klien dengan ukuran terbatas untuk membatasi jumlah koneksi masuk bersamaan dari aplikasi klien.
- Ketika klien terputus dari server karena batas waktu, coba lagi dengan mundur eksponensial dengan jitter. Hal ini membantu menghindari banyak klien di server pada waktu yang sama.
- Gunakan panduan di Menemukan titik akhir koneksi guna menemukan titik akhir klaster untuk melakukan penemuan klaster. Dengan begitu, Anda menyebarkan beban penemuan di semua simpul di klaster (hingga 90) alih-alih mengenai beberapa simpul benih hardcode di klaster.

Berikut ini adalah beberapa contoh kode untuk logika percobaan ulang mundur eksponensial di redis-py, PHPRedis, dan Lettuce.

Contoh logika backoff 1: redis-py

redis-py memiliki mekanisme coba ulang bawaan yang mencoba ulang satu kali segera setelah kegagalan. Mekanisme ini dapat diaktifkan melalui argumen `retry_on_timeout` yang diberikan saat membuat objek Redis. Di sini kami mendemonstrasikan mekanisme coba ulang khusus dengan mundur eksponensial dan jitter. Kami telah mengirimkan permintaan tarik untuk mengimplementasikan mundur eksponensial secara native di redis-py (#1494). Di masa depan, hal tersebut mungkin tidak perlu diimplementasikan secara manual.

```
def run_with_backoff(function, retries=5):
base_backoff = 0.1 # base 100ms backoff
max_backoff = 10 # sleep for maximum 10 seconds
tries = 0
while True:
try:
  return function()
except (ConnectionError, TimeoutError):
  if tries >= retries:
 raise
  backoff = min(max_backoff, base_backoff * (pow(2, tries) + random.random()))
  print(f"sleeping for {backoff:.2f}s")
  sleep(backoff)
  tries += 1
```

Anda kemudian dapat menggunakan kode berikut untuk menetapkan nilai:

```
client = redis.Redis(connection_pool=redis.BlockingConnectionPool(host=HOST,
 max_connections=10))
```

```
res = run_with_backoff(lambda: client.set("key", "value"))
print(res)
```

Bergantung pada beban kerja Anda, Anda mungkin ingin mengubah nilai backoff dasar dari 1 detik menjadi puluhan atau ratusan milidetik untuk beban kerja yang sensitif terhadap latensi.

Contoh logika backoff 2: PHPRedis

PHPRedis memiliki mekanisme coba ulang bawaan yang mencoba ulang maksimum (tidak dapat dikonfigurasi) sebanyak 10 kali. Ada penundaan yang dapat dikonfigurasi antara percobaan (dengan jitter dari percobaan ulang kedua dan seterusnya). Untuk informasi selengkapnya, lihat kode sampel berikut. Kami telah mengirimkan permintaan tarik untuk mengimplementasikan backoff eksponensial secara native di PHPredis (#1986) yang telah digabungkan dan didokumentasikan. Bagi yang menggunakan rilis terbaru PhPredis, Anda tidak perlu mengimplementasikan secara manual tetapi kami telah menyertakan referensi di sini untuk Anda menggunakan versi yang lebih lama. Untuk saat ini, berikut ini adalah contoh kode yang mengonfigurasi penundaan mekanisme coba lagi:

```
$timeout = 0.1; // 100 millisecond connection timeout
$retry_interval = 100; // 100 millisecond retry interval
$client = new Redis();
if($client->pconnect($HOST, $PORT, $timeout, NULL, $retry_interval) != TRUE) {
 return; // ERROR: connection failed
}
$client->set($key, $value);
```

Contoh logika backoff 3: Lettuce

Lettuce memiliki mekanisme coba ulang bawaan berdasarkan strategi mundur eksponensial yang dijelaskan dalam postingan Exponential Backoff dan Jitter. Berikut ini adalah kutipan kode yang menunjukkan pendekatan jitter penuh:

```
public static void main(String[] args)
{
 ClientResources resources = null;
 RedisClient client = null;

 try {
  resources = DefaultClientResources.builder()
    .reconnectDelay(Delay.fullJitter(
   Duration.ofMillis(100),     // minimum 100 millisecond delay
```

```
    Duration.ofSeconds(5),      // maximum 5 second delay
    100, TimeUnit.MILLISECONDS) // 100 millisecond base
   ).build();

   client = RedisClient.create(resources, RedisURI.create(HOST, PORT));
   client.setOptions(ClientOptions.builder()
  .socketOptions(SocketOptions.builder().connectTimeout(Duration.ofMillis(100)).build()) //
  100 millisecond connection timeout
  .timeoutOptions(TimeoutOptions.builder().fixedTimeout(Duration.ofSeconds(5)).build()) //
  5 second command timeout
  .build());

      // use the connection pool from above example
  } finally {
   if (connection != null) {
    connection.close();
   }

   if (client != null){
    client.shutdown();
   }

   if (resources != null){
    resources.shutdown();
   }

  }
}
```

## Mengonfigurasi batas waktu sisi klien

Konfigurasikan batas waktu sisi klien dengan tepat agar server memiliki cukup waktu untuk memproses permintaan dan menghasilkan respons. Hal juga memungkinkannya melakukan gagal cepat (fail fast) jika koneksi ke server tidak dapat dibuat. Perintah Redis tertentu bisa lebih mahal secara komputasi dibandingkan yang lain. Misalnya, skrip Lua atau transaksi MULTI/EXEC yang berisi beberapa perintah yang harus dijalankan secara atomik. Secara umum, batas waktu sisi klien yang lebih tinggi disarankan untuk menghindari batas waktu klien sebelum respons diterima dari server, termasuk yang berikut:

- Menjalankan perintah di beberapa kunci
- Menjalankan transaksi MULTI/EXEC atau skrip Lua yang terdiri dari beberapa perintah Redis individual

- Membaca nilai besar
- Melakukan operasi pemblokiran seperti BLPOP

Dalam kasus operasi pemblokiran seperti BLPOP, praktik terbaiknya adalah dengan mengatur batas waktu perintah ke jumlah yang lebih rendah dari batas waktu soket.

Berikut ini adalah contoh kode untuk menerapkan batas waktu sisi klien di redis-py, PhPredis, dan Lettuce.

Contoh konfigurasi batas waktu 1: redis-py

Berikut ini adalah contoh kode dengan redis-py:

```
# connect to Redis server with a 100 millisecond timeout
# give every Redis command a 2 second timeout
client = redis.Redis(connection_pool=redis.BlockingConnectionPool(host=HOST,
 max_connections=10,socket_connect_timeout=0.1,socket_timeout=2))

res = client.set("key", "value") # will timeout after 2 seconds
print(res)                       # if there is a connection error

res = client.blpop("list", timeout=1) # will timeout after 1 second
                                      # less than the 2 second socket timeout
print(res)
```

Contoh konfigurasi batas waktu 2: PHPRedis

Berikut ini adalah contoh kode dengan PHPRedis:

```
// connect to Redis server with a 100ms timeout
// give every Redis command a 2s timeout
$client = new Redis();
$timeout = 0.1; // 100 millisecond connection timeout
$retry_interval = 100; // 100 millisecond retry interval
$client = new Redis();
if($client->pconnect($HOST, $PORT, 0.1, NULL, 100, $read_timeout=2) != TRUE){
 return; // ERROR: connection failed
}
$client->set($key, $value);

$res = $client->set("key", "value"); // will timeout after 2 seconds
print "$res\n";                      // if there is a connection error
```

```
$res = $client->blpop("list", 1); // will timeout after 1 second
print "$res\n";                   // less than the 2 second socket timeout
```

Contoh konfigurasi batas waktu 3: Lettuce

Berikut ini adalah contoh kode dengan Lettuce:

```
// connect to Redis server and give every command a 2 second timeout
public static void main(String[] args)
{
 RedisClient client = null;
 StatefulRedisConnection<String, String> connection = null;
 try {
  client = RedisClient.create(RedisURI.create(HOST, PORT));
  client.setOptions(ClientOptions.builder()
 .socketOptions(SocketOptions.builder().connectTimeout(Duration.ofMillis(100)).build()) //
 100 millisecond connection timeout
 .timeoutOptions(TimeoutOptions.builder().fixedTimeout(Duration.ofSeconds(2)).build()) //
 2 second command timeout
 .build());

  // use the connection pool from above example

  commands.set("key", "value"); // will timeout after 2 seconds
  commands.blpop(1, "list"); // BLPOP with 1 second timeout
 } finally {
  if (connection != null) {
   connection.close();
  }

  if (client != null){
   client.shutdown();
  }
 }
}
```

## Mengonfigurasi batas waktu idle sisi server

Kami telah mengamati kasus ketika aplikasi pelanggan memiliki klien idle terhubung dalam jumlah yang tinggi, tetapi tidak secara aktif mengirim perintah. Dalam skenario tersebut, Anda dapat menghabiskan semua 65.000 koneksi dengan jumlah klien idle yang tinggi. Untuk menghindari

skenario tersebut, konfigurasikan pengaturan batas waktu dengan sesuai di server melalui Parameter spesifik Redis. Hal ini memastikan bahwa server secara aktif memutus klien idle untuk menghindari peningkatan jumlah koneksi. Pengaturan ini tidak tersedia di cache nirserver.

## Skrip Redis

Redis mendukung lebih dari 200 perintah, termasuk untuk menjalankan skrip Lua. Namun, ketika menangani skrip Lua, ada beberapa kesalahan yang dapat mempengaruhi memori dan ketersediaan Redis.

### Skrip Lua yang tidak diparameterisasi

Setiap skrip Lua di-cache di server Redis sebelum berjalan. Skrip Lua yang tidak diparameterisasi bersifat unik, yang dapat menyebabkan server Redis menyimpan skrip Lua dalam jumlah besar dan menghabiskan lebih banyak memori. Untuk memitigasi hal ini, pastikan bahwa semua skrip Lua diparameterisasi dan secara teratur melakukan SCRIPT FLUSH untuk membersihkan skrip Lua yang di-cache jika diperlukan.

Contoh berikut menunjukkan cara menggunakan skrip parameter. Pertama, kami memiliki contoh pendekatan tanpa parameter yang menghasilkan tiga skrip Lua yang di-cache yang berbeda dan tidak disarankan:

```
eval "return redis.call('set','key1','1')" 0
eval "return redis.call('set','key2','2')" 0
eval "return redis.call('set','key3','3')" 0
```

Sebagai gantinya, gunakan pola berikut untuk membuat skrip tunggal yang dapat menerima parameter yang diteruskan:

```
eval "return redis.call('set',KEYS[1],ARGV[1])" 1 key1 1
eval "return redis.call('set',KEYS[1],ARGV[1])" 1 key2 2
eval "return redis.call('set',KEYS[1],ARGV[1])" 1 key3 3
```

### Skrip Lua yang berjalan lama

Skrip Lua dapat menjalankan beberapa perintah secara atomik, sehingga bisa memakan waktu lebih lama untuk diselesaikan daripada perintah Redis pada umumnya. Jika skrip Lua hanya menjalankan operasi hanya-baca, Anda dapat menghentikannya di tengah proses. Namun, segera setelah skrip Lua melakukan operasi penulisan, prosesnya tidak dapat dihentikan dan harus dijalankan hingga

selesai. Skrip Lua yang berjalan lama yang bermutasi dapat menyebabkan server Redis tidak responsif untuk waktu yang lama. Untuk memitigasi masalah ini, hindari skrip Lua yang berjalan lama dan uji skrip di lingkungan praproduksi.

Skrip Lua dengan tulisan stealth

Ada beberapa cara skrip Lua dapat terus menulis data baru ke Redis bahkan ketika Redis selesai `maxmemory`:

- Skrip dimulai ketika server Redis berada di bawah `maxmemory`, dan berisi beberapa operasi tulis di dalamnya
- Perintah tulis pertama skrip tidak menghabiskan memori (seperti DEL), yang diikuti oleh lebih banyak operasi tulis yang menghabiskan memori
- Anda dapat memitigasi masalah ini dengan mengonfigurasi kebijakan pengosongan yang tepat di server Redis selain `noeviction`. Hal ini memungkinkan Redis mengosongkan item dan mengosongkan memori di antara skrip Lua.

## Menyimpan item komposit besar

Dalam beberapa skenario, aplikasi dapat menyimpan item komposit besar di Redis (seperti set data hash multi-GB). Hal ini bukanlah praktik yang direkomendasikan karena sering menyebabkan masalah performa di Redis. Misalnya, klien dapat melakukan perintah HGETALL untuk mengambil seluruh koleksi hash multi GB. Hal ini dapat menghasilkan tekanan memori yang signifikan ke server Redis yang melakukan buffering item besar di buffer output klien. Selain itu, untuk migrasi slot dalam mode klaster, ElastiCache tidak memigrasikan slot yang berisi item dengan ukuran serial yang lebih besar dari 256 MB.

Untuk mengatasi masalah item besar, kami memiliki rekomendasi berikut:

- Pecah item komposit besar menjadi beberapa item yang lebih kecil. Misalnya, pisahkan koleksi hash besar menjadi bidang nilai kunci individual dengan skema nama kunci yang mencerminkan koleksi dengan tepat, seperti menggunakan awalan umum dalam nama kunci untuk mengidentifikasi kumpulan item. Jika Anda harus mengakses beberapa bidang dalam koleksi yang sama secara atomik, Anda dapat menggunakan perintah MGET untuk mengambil beberapa nilai kunci dalam perintah yang sama.
- Jika Anda mengevaluasi semua opsi dan masih tidak dapat memecah kumpulan data koleksi besar, coba gunakan perintah yang beroperasi di subset data dalam koleksi alih-alih seluruh koleksi. Hindari kasus penggunaan yang mengharuskan Anda mengambil seluruh koleksi multi-GB

secara atomik dalam perintah yang sama. Salah satu contohnya adalah menggunakan perintah HGET atau HMGET alih-alih HGETALL di koleksi hash.

## Konfigurasi klien Lettuce

Bagian ini menjelaskan opsi konfigurasi Java dan Lettuce yang direkomendasikan, dan penerapannya di klaster ElastiCache.

Rekomendasi di bagian ini diuji dengan Lettuce versi 6.2.2.

Topik

- Contoh: Konfigurasi Lettuce untuk mode klaster dan TLS diaktifkan
- Contoh: Konfigurasi Lettuce untuk mode klaster dinonaktifkan dan TLS diaktifkan.

### TTL cache DNS Java

Mesin virtual Java (JVM) menyimpan cache pencarian nama DNS. Ketika JVM menyelesaikan nama host ke alamat IP, JVM menyimpan cache alamat IP untuk jangka waktu tertentu, yang disebut time-to-live (TTL).

Pilihan nilai TTL merupakan trade-off antara latensi dan responsif terhadap perubahan. Dengan TTL yang lebih pendek, penyelesai DNS melihat pembaruan di DNS klaster dengan lebih cepat. Hal ini dapat membuat aplikasi Anda merespons penggantian atau alur kerja lain yang dialami klaster Anda lebih cepat. Namun, jika TTL terlalu rendah, hal ini meningkatkan volume kueri, yang dapat meningkatkan latensi aplikasi Anda. Meskipun tidak ada nilai TTL yang benar, ada baiknya untuk mempertimbangkan lamanya waktu yang dapat Anda tunggu agar perubahan berlaku saat menetapkan nilai TTL Anda.

Karena simpul ElastiCache menggunakan entri nama DNS yang mungkin berubah, sebaiknya konfigurasi JVM Anda dengan TTL rendah 5 hingga 10 detik. Hal ini dapat memastikan bahwa ketika alamat IP simpul berubah, aplikasi Anda dapat menerima dan menggunakan alamat IP baru sumber daya dengan mengkueri ulang DNS.

Pada beberapa konfigurasi Java, TTL default JVM diatur untuk tidak pernah menyegarkan entri DNS hingga JVM dimulai ulang.

Untuk detail tentang cara mengatur TTL JVM Anda, lihat Cara mengatur TTL JVM.

### Versi Lettuce

Sebaiknya gunakan Lettuce versi 6.2.2 atau versi yang lebih baru.

Titik akhir

Saat Anda menggunakan klaster dengan mode klastert diaktifkan, atur `redisUri` ke titik akhir konfigurasi klaster. Pencarian DNS untuk URI ini mengembalikan daftar semua simpul yang tersedia di klaster, dan diselesaikan secara acak ke salah satu simpul selama inisialisasi klaster. Untuk detail selengkapnya tentang cara kerja penyegaran topologi, lihat dynamicRefreshResources nanti dalam topik ini.

SocketOption

Aktifkan KeepAlive. Mengaktifkan opsi ini akan mengurangi kebutuhan untuk menangani koneksi yang gagal selama runtime perintah.

Pastikan Anda menetapkan Batas waktu koneksi berdasarkan persyaratan aplikasi dan beban kerja Anda. Untuk informasi selengkapnya, lihat bagian Batas waktu dalam topik ini.

ClusterClientOption: Mode klaster Opsi klien yang diaktifkan

Aktifkan AutoRconnect saat koneksi hilang.

Setel CommandTimeout. Untuk detail selengkapnya, lihat bagian Batas waktu dalam topik ini.

Setel NodeFilter untuk menyaring simpul yang gagal dari topologi. Lettuce menyimpan semua simpul yang ditemukan di output 'simpul klaster' (termasuk simpul dengan status PFAIL/FAIL) di 'partisi' klien (juga dikenal sebagai serpihan). Selama proses pembuatan topologi klaster, Lettuce mencoba untuk terhubung ke semua simpul partisi. Perilaku Lettuce menambahkan simpul yang gagal ini dapat menyebabkan kesalahan koneksi (atau peringatan) ketika simpul diganti karena alasan apa pun.

Misalnya, setelah failover selesai dan klaster memulai proses pemulihan, sementara ClusterTopology sedang diperbarui, peta simpul bus klaster memiliki periode waktu singkat ketika simpul bawah terdaftar sebagai simpul FAIL, sebelum benar-benar dihapus dari topologi. Selama periode ini, klien Lettuce Redis menganggapnya sebagai simpul yang sehat dan terus terhubung dengannya. Hal ini menyebabkan kegagalan setelah percobaan kembali habis.

Contoh:

```
final ClusterClientOptions clusterClientOptions =
    ClusterClientOptions.builder()
    ... // other options
    .nodeFilter(it ->
```

```
        ! (it.is(RedisClusterNode.NodeFlag.FAIL)
        || it.is(RedisClusterNode.NodeFlag.EVENTUAL_FAIL)
        || it.is(RedisClusterNode.NodeFlag.HANDSHAKE)
        || it.is(RedisClusterNode.NodeFlag.NOADDR)))
    .validateClusterNodeMembership(false)
    .build();
redisClusterClient.setOptions(clusterClientOptions);
```

> **Note**
>
> Pemfilteran simpul paling baik digunakan dengan DynamicRefreshSources disetel ke true. Jika tidak, jika tampilan topologi diambil dari satu simpul seed bermasalah, yang melihat simpul primer dari beberapa serpihan gagal, itu akan menyaring simpul primer ini, yang akan mengakibatkan slot tidak tercakup. Memiliki beberapa simpul seed (ketika DynamicRefreshSources benar) akan mengurangi kemungkinan masalah ini, karena setidaknya beberapa simpul seed harus memiliki tampilan topologi yang diperbarui setelah failover dengan primer yang baru dipromosikan.

ClusterTopologyRefreshOptions: Opsi untuk mengontrol penyegaran topologi klaster dari klien Mode Klaster Diaktifkan

> **Note**
>
> Klaster dengan mode klaster dinonaktifkan tidak mendukung perintah penemuan klaster dan tidak kompatibel dengan semua fungsionalitas penemuan topologi dinamis klien.
> Mode klaster dinonaktifkan dengan ElastiCache tidak kompatibel dengan `MasterSlaveTopologyRefresh` Lettuce. Sebagai gantinya, untuk mode klaster dinonaktifkan, Anda dapat mengonfigurasi `StaticMasterReplicaTopologyProvider` dan menyediakan titik akhir baca dan tulis klaster.
> Untuk informasi selengkapnya tentang menghubungkan ke klaster dengan mode klaster dinonaktifkan, lihat Menemukan Titik Akhir Klaster Redis (Mode Klaster Dinonaktifkan) (Konsol).
> Jika Anda ingin menggunakan fungsionalitas penemuan topologi dinamis Lettuce, Anda dapat membuat klaster dengan mode klaster diaktifkan dengan konfigurasi serpihan yang sama dengan klaster yang ada. Namun, untuk klaster dengan mode klaster diaktifkan, sebaiknya konfigurasi setidaknya 3 serpihan dengan minimal 1 replika untuk mendukung failover cepat.

Aktifkan EnablePeriodIcreFresh. Hal ini memungkinkan pembaruan topologi klaster periodik sehingga klien memperbarui topologi klaster dalam interval RefreshPeriod (default: 60 detik). Ketika dinonaktifkan, klien memperbarui topologi klaster hanya ketika kesalahan terjadi ketika mencoba menjalankan perintah terhadap klaster.

Dengan mengaktifkan opsi ini, Anda dapat mengurangi latensi yang terkait dengan penyegaran topologi klaster dengan menambahkan pekerjaan ini ke tugas latar belakang. Saat penyegaran topologi dilakukan dalam pekerjaan latar belakang, prosesnya bisa agak lambat untuk klaster yang memiliki banyak simpul. Hal ini karena semua simpul sedang dikueri untuk mendapatkan tampilan klaster yang paling terbaru. Jika Anda menjalankan klaster dalam jumlah besar, Anda mungkin ingin menambah periode.

Aktifkan enableAllAdaptiveRefreshTriggers. Hal ini memungkinkan penyegaran topologi adaptif yang menggunakan semua pemicu: MOVED_REDIRECT, ASK_REDIRECT, PERSISTENT_RECONNECTS, UNCOVERED_SLOT, UNKNOWN_NODE. Pemicu penyegaran adaptif memulai pembaruan tampilan topologi berdasarkan peristiwa yang terjadi selama operasi klaster Redis. Mengaktifkan opsi ini akan membuat penyegaran topologi langsung dilakukan ketika salah satu pemicu sebelumnya terjadi. Penyegaran yang dipicu secara adaptif dibatasi kecepatannya menggunakan batas waktu karena peristiwa dapat terjadi dalam skala besar (batas waktu default antara pembaruan: 30).

Aktifkan closeStaleConnections. Hal ini memungkinkan penutupan koneksi yang sudah tidak berlaku saat menyegarkan topologi klaster. Hal ini hanya berlaku jika ClusterTopologyRefreshOptions.isPeriodicRefreshEnabled() benar. Ketika diaktifkan, klien dapat menutup koneksi yang tidak berlaku dan membuat yang baru di latar belakang. Hal ini mengurangi kebutuhan untuk menangani koneksi yang gagal selama runtime perintah.

Aktifkan dynamicRefreshResources. Sebaiknya aktifkan dynamicRefreshResources untuk klaster kecil, dan menonaktifkannya untuk klaster besar. dynamicRefreshResources memungkinkan penemuan simpul klaster dari simpul seed yang disediakan (misalnya, titik akhir konfigurasi klaster). Ia menggunakan semua simpul yang ditemukan sebagai sumber untuk menyegarkan topologi klaster.

Menggunakan dynamic refresh akan mengkueri semua simpul yang ditemukan untuk topologi klaster dan mencoba memilih tampilan klaster yang paling akurat. Jika disetel ke false, hanya simpul seed awal yang digunakan sebagai sumber untuk penemuan topologi, dan jumlah klien diperoleh hanya untuk simpul seed awal. Saat dinonaktifkan, jika titik akhir konfigurasi klaster diselesaikan ke simpul yang gagal, percobaan untuk menyegarkan tampilan klaster akan gagal dan menyebabkan pengecualian. Skenario ini dapat terjadi karena dibutuhkan beberapa waktu hingga entri simpul yang

gagal dihapus dari titik akhir konfigurasi klaster. Oleh karena itu, titik akhir konfigurasi masih dapat diselesaikan secara acak ke simpul yang gagal untuk waktu yang singkat.

Namun, ketika diaktifkan, kita menggunakan semua simpul klaster yang diterima dari tampilan klaster untuk mengkueri tampilan mereka saat ini. Karena kita memfilter simpul yang gagal dari tampilan tersebut, penyegaran topologi akan berhasil. Namun, ketika dynamicRefreshSources benar, Lettuce mengkueri semua simpul untuk mendapatkan tampilan klaster, lalu membandingkan hasilnya. Jadi, klaster dengan banyak simpul bisa jadi mahal. Sebaiknya nonaktifkan fitur ini untuk klaster dengan banyak simpul.

```
final ClusterTopologyRefreshOptions topologyOptions =
    ClusterTopologyRefreshOptions.builder()
    .enableAllAdaptiveRefreshTriggers()
    .enablePeriodicRefresh()
    .dynamicRefreshSources(true)
    .build();
```

ClientResources

Konfigurasikan DnsResolver dengan DirContextDnsResolver. Penyelesai DNS didasarkan pada com.sun.jndi.dns.DnsContextFactory Java.

Konfigurasikan reconnectDelay dengan mundur eksponensial dan jitter penuh. Lettuce memiliki mekanisme coba ulang bawaan berdasarkan strategi mundur eksponensial. Untuk detailnya, lihat Mundur Eksponensial dan Jitter di Blog Arsitektur AWS. Untuk informasi lebih lanjut tentang pentingnya memiliki strategi backoff percobaan ulang, lihat bagian logika backoff postingan blog Praktik Terbaik di Blog Basis Data AWS.

```
ClientResources clientResources = DefaultClientResources.builder()
   .dnsResolver(new DirContextDnsResolver())
    .reconnectDelay(
        Delay.fullJitter(
            Duration.ofMillis(100),     // minimum 100 millisecond delay
            Duration.ofSeconds(10),      // maximum 10 second delay
            100, TimeUnit.MILLISECONDS)) // 100 millisecond base
    .build();
```

Batas Waktu

Gunakan nilai batas waktu koneksi yang lebih rendah daripada batas waktu perintah Anda. Lettuce menggunakan pembentukan koneksi lambat. Jadi, jika batas waktu koneksi lebih tinggi dari batas

waktu perintah, Anda dapat mengalami periode kegagalan persisten setelah penyegaran topologi jika Lettuce mencoba terhubung ke simpul yang tidak sehat dan batas waktu perintah selalu terlampaui.

Gunakan batas waktu perintah dinamis untuk perintah yang berbeda. Sebaiknya tetapkan batas waktu perintah berdasarkan durasi perintah yang diharapkan. Misalnya, gunakan batas waktu yang lebih lama untuk perintah yang mengulangi beberapa kunci, seperti skrip FLUSHDB, FLUSHALL, KEYS, SMEMBERS, atau Lua. Gunakan batas waktu yang lebih pendek untuk perintah kunci tunggal, seperti SET, GET, dan HSET.

> **Note**
>
> Batas waktu yang dikonfigurasi dalam contoh berikut adalah untuk pengujian yang menjalankan perintah SET/GET dengan kunci dan nilai hingga 20 byte. Waktu pemrosesan bisa lebih lama ketika perintah kompleks atau kunci dan nilai lebih besar. Anda harus mengatur batas waktu berdasarkan kasus penggunaan aplikasi Anda.

```
private static final Duration META_COMMAND_TIMEOUT = Duration.ofMillis(1000);
private static final Duration DEFAULT_COMMAND_TIMEOUT = Duration.ofMillis(250);
// Socket connect timeout should be lower than command timeout for Lettuce
private static final Duration CONNECT_TIMEOUT = Duration.ofMillis(100);

SocketOptions socketOptions = SocketOptions.builder()
    .connectTimeout(CONNECT_TIMEOUT)
    .build();


class DynamicClusterTimeout extends TimeoutSource {
     private static final Set<ProtocolKeyword> META_COMMAND_TYPES =
 ImmutableSet.<ProtocolKeyword>builder()
          .add(CommandType.FLUSHDB)
          .add(CommandType.FLUSHALL)
          .add(CommandType.CLUSTER)
          .add(CommandType.INFO)
          .add(CommandType.KEYS)
          .build();

    private final Duration defaultCommandTimeout;
    private final Duration metaCommandTimeout;

    DynamicClusterTimeout(Duration defaultTimeout, Duration metaTimeout)
```

```
    {
        defaultCommandTimeout = defaultTimeout;
        metaCommandTimeout = metaTimeout;
    }

    @Override
    public long getTimeout(RedisCommand<?, ?, ?> command) {
        if (META_COMMAND_TYPES.contains(command.getType())) {
            return metaCommandTimeout.toMillis();
        }
        return defaultCommandTimeout.toMillis();
    }
}

// Use a dynamic timeout for commands, to avoid timeouts during
// cluster management and slow operations.
TimeoutOptions timeoutOptions = TimeoutOptions.builder()
.timeoutSource(
    new DynamicClusterTimeout(DEFAULT_COMMAND_TIMEOUT, META_COMMAND_TIMEOUT))
.build();
```

Contoh: Konfigurasi Lettuce untuk mode klaster dan TLS diaktifkan

> **Note**
>
> Batas waktu dalam contoh berikut adalah untuk pengujian yang menjalankan perintah SET/GET dengan kunci dan nilai hingga 20 byte. Waktu pemrosesan bisa lebih lama ketika perintah kompleks atau kunci dan nilai lebih besar. Anda harus mengatur batas waktu berdasarkan kasus penggunaan aplikasi Anda.

```
// Set DNS cache TTL
public void setJVMProperties() {
    java.security.Security.setProperty("networkaddress.cache.ttl", "10");
}

private static final Duration META_COMMAND_TIMEOUT = Duration.ofMillis(1000);
private static final Duration DEFAULT_COMMAND_TIMEOUT = Duration.ofMillis(250);
// Socket connect timeout should be lower than command timeout for Lettuce
private static final Duration CONNECT_TIMEOUT = Duration.ofMillis(100);

// Create RedisURI from the cluster configuration endpoint
```

```
clusterConfigurationEndpoint = <cluster-configuration-endpoint> // TODO: add your
 cluster configuration endpoint
final RedisURI redisUriCluster =
    RedisURI.Builder.redis(clusterConfigurationEndpoint)
        .withPort(6379)
        .withSsl(true)
        .build();

// Configure the client's resources
ClientResources clientResources = DefaultClientResources.builder()
    .reconnectDelay(
        Delay.fullJitter(
            Duration.ofMillis(100),     // minimum 100 millisecond delay
            Duration.ofSeconds(10),      // maximum 10 second delay
            100, TimeUnit.MILLISECONDS)) // 100 millisecond base
    .dnsResolver(new DirContextDnsResolver())
    .build();

// Create a cluster client instance with the URI and resources
RedisClusterClient redisClusterClient =
    RedisClusterClient.create(clientResources, redisUriCluster);

// Use a dynamic timeout for commands, to avoid timeouts during
// cluster management and slow operations.
class DynamicClusterTimeout extends TimeoutSource {
     private static final Set<ProtocolKeyword> META_COMMAND_TYPES =
 ImmutableSet.<ProtocolKeyword>builder()
          .add(CommandType.FLUSHDB)
          .add(CommandType.FLUSHALL)
          .add(CommandType.CLUSTER)
          .add(CommandType.INFO)
          .add(CommandType.KEYS)
          .build();

    private final Duration metaCommandTimeout;
    private final Duration defaultCommandTimeout;

    DynamicClusterTimeout(Duration defaultTimeout, Duration metaTimeout)
    {
        defaultCommandTimeout = defaultTimeout;
        metaCommandTimeout = metaTimeout;
    }

    @Override
```

```
    public long getTimeout(RedisCommand<?, ?, ?> command) {
        if (META_COMMAND_TYPES.contains(command.getType())) {
            return metaCommandTimeout.toMillis();
        }
        return defaultCommandTimeout.toMillis();
    }
}

TimeoutOptions timeoutOptions = TimeoutOptions.builder()
    .timeoutSource(new DynamicClusterTimeout(DEFAULT_COMMAND_TIMEOUT,
 META_COMMAND_TIMEOUT))
     .build();

// Configure the topology refreshment optionts
final ClusterTopologyRefreshOptions topologyOptions =
    ClusterTopologyRefreshOptions.builder()
    .enableAllAdaptiveRefreshTriggers()
    .enablePeriodicRefresh()
    .dynamicRefreshSources(true)
    .build();

// Configure the socket options
final SocketOptions socketOptions =
    SocketOptions.builder()
    .connectTimeout(CONNECT_TIMEOUT)
    .keepAlive(true)
    .build();

// Configure the client's options
final ClusterClientOptions clusterClientOptions =
    ClusterClientOptions.builder()
    .topologyRefreshOptions(topologyOptions)
    .socketOptions(socketOptions)
    .autoReconnect(true)
    .timeoutOptions(timeoutOptions)
    .nodeFilter(it ->
        ! (it.is(RedisClusterNode.NodeFlag.FAIL)
        || it.is(RedisClusterNode.NodeFlag.EVENTUAL_FAIL)
        || it.is(RedisClusterNode.NodeFlag.NOADDR)))
    .validateClusterNodeMembership(false)
    .build();

redisClusterClient.setOptions(clusterClientOptions);
```

```
// Get a connection
final StatefulRedisClusterConnection<String, String> connection =
    redisClusterClient.connect();

// Get cluster sync/async commands
RedisAdvancedClusterCommands<String, String> sync = connection.sync();
RedisAdvancedClusterAsyncCommands<String, String> async = connection.async();
```

Contoh: Konfigurasi Lettuce untuk mode klaster dinonaktifkan dan TLS diaktifkan.

> **Note**
>
> Batas waktu dalam contoh berikut adalah untuk pengujian yang menjalankan perintah SET/GET dengan kunci dan nilai hingga 20 byte. Waktu pemrosesan bisa lebih lama ketika perintah kompleks atau kunci dan nilai lebih besar. Anda harus mengatur batas waktu berdasarkan kasus penggunaan aplikasi Anda.

```
// Set DNS cache TTL
public void setJVMProperties() {
    java.security.Security.setProperty("networkaddress.cache.ttl", "10");
}

private static final Duration META_COMMAND_TIMEOUT = Duration.ofMillis(1000);
private static final Duration DEFAULT_COMMAND_TIMEOUT = Duration.ofMillis(250);
// Socket connect timeout should be lower than command timeout for Lettuce
private static final Duration CONNECT_TIMEOUT = Duration.ofMillis(100);

// Create RedisURI from the primary/reader endpoint
clusterEndpoint = <primary/reader-endpoint> // TODO: add your node endpoint
RedisURI redisUriStandalone =

 RedisURI.Builder.redis(clusterEndpoint).withPort(6379).withSsl(true).withDatabase(0).build();

ClientResources clientResources =
    DefaultClientResources.builder()
        .dnsResolver(new DirContextDnsResolver())
        .reconnectDelay(
            Delay.fullJitter(
                Duration.ofMillis(100), // minimum 100 millisecond delay
                Duration.ofSeconds(10), // maximum 10 second delay
                100,
```

```
                TimeUnit.MILLISECONDS)) // 100 millisecond base
        .build();

// Use a dynamic timeout for commands, to avoid timeouts during
// slow operations.
class DynamicTimeout extends TimeoutSource {
     private static final Set<ProtocolKeyword> META_COMMAND_TYPES =
 ImmutableSet.<ProtocolKeyword>builder()
          .add(CommandType.FLUSHDB)
          .add(CommandType.FLUSHALL)
          .add(CommandType.INFO)
          .add(CommandType.KEYS)
          .build();

    private final Duration metaCommandTimeout;
    private final Duration defaultCommandTimeout;

    DynamicTimeout(Duration defaultTimeout, Duration metaTimeout)
    {
        defaultCommandTimeout = defaultTimeout;
        metaCommandTimeout = metaTimeout;
    }

    @Override
    public long getTimeout(RedisCommand<?, ?, ?> command) {
        if (META_COMMAND_TYPES.contains(command.getType())) {
            return metaCommandTimeout.toMillis();
        }
        return defaultCommandTimeout.toMillis();
    }
}

TimeoutOptions timeoutOptions = TimeoutOptions.builder()
    .timeoutSource(new DynamicTimeout(DEFAULT_COMMAND_TIMEOUT, META_COMMAND_TIMEOUT))
     .build();

final SocketOptions socketOptions =
    SocketOptions.builder().connectTimeout(CONNECT_TIMEOUT).keepAlive(true).build();

ClientOptions clientOptions =

 ClientOptions.builder().timeoutOptions(timeoutOptions).socketOptions(socketOptions).build();

RedisClient redisClient = RedisClient.create(clientResources, redisUriStandalone);
```

```
redisClient.setOptions(clientOptions);
```

## Contoh klien IPv6

Berikut ini adalah praktik terbaik untuk berinteraksi dengan sumber daya berkemampuan IPv6 dengan pustaka ElastiCache klien sumber terbuka yang umum digunakan. Anda dapat melihat praktik terbaik yang ada untuk berinteraksi dengan ElastiCache rekomendasi tentang mengonfigurasi klien untuk ElastiCache sumber daya. Namun, ada beberapa peringatan yang perlu diperhatikan saat berinteraksi dengan sumber daya dengan IPv6 aktif.

Klien yang divalidasi

ElastiCache kompatibel dengan Redis open-source. Ini berarti bahwa klien Redis open source yang mendukung koneksi IPv6 harus dapat terhubung ke IPv6 yang diaktifkan untuk kluster Redis. ElastiCache Selain itu, beberapa klien Python dan Java paling populer telah diuji dan divalidasi secara khusus untuk berfungsi dengan semua konfigurasi jenis jaringan yang didukung (hanya IPv4, hanya IPv6, dan Tumpukan Ganda)

Klien yang Divalidasi:

- Redis Py () – 4.1.2
- Lettuce – Versi: 6.1.6.RELEASE
- Jedis - Versi: 3.6.0

Mengonfigurasi protokol pilihan untuk klaster tumpukan ganda

Untuk klaster Redis dengan mode klaster diaktifkan, Anda dapat mengontrol protokol yang akan digunakan klien untuk terhubung ke simpul di klaster dengan parameter Penemuan IP. Parameter Penemuan IP dapat diatur ke IPv4 atau IPv6.

Untuk klaster Redis, parameter Penemuan IP menetapkan protokol IP yang digunakan dalam output slot klaster (), serpihan klaster (), dan simpul klaster (). Perintah tersebut digunakan oleh klien untuk menemukan topologi klaster. Klien menggunakan IP dalam perintah tersebut untuk terhubung ke simpul lain dalam klaster.

Perubahan pada Penemuan IP tidak akan mengakibatkan waktu henti untuk klien yang terhubung. Namun, perubahan ini akan memakan waktu untuk disebarkan. Untuk menentukan kapan perubahan selesai disebarkan untuk Klaster Redis, pantau output dari `cluster slots`. Setelah semua simpul

yang ditampilkan oleh perintah slot klaster melaporkan IP dengan protokol baru, berarti perubahan telah selesai disebarkan.

Contoh dengan Redis-Py:

```
cluster = RedisCluster(host="xxxx", port=6379)
target_type = IPv6Address # Or IPv4Address if changing to IPv4

nodes = set()
while len(nodes) == 0 or not all((type(ip_address(host)) is target_type) for host in
 nodes):
    nodes = set()

   # This refreshes the cluster topology and will discovery any node updates.
   # Under the hood it calls cluster slots
    cluster.nodes_manager.initialize()
    for node in cluster.get_nodes():
        nodes.add(node.host)
    self.logger.info(nodes)

    time.sleep(1)
```

Contoh dengan Lettuce:

```
RedisClusterClient clusterClient = RedisClusterClient.create(RedisURI.create("xxxx",
 6379));

Class targetProtocolType = Inet6Address.class; // Or Inet4Address.class if you're
 switching to IPv4

Set<String> nodes;

do {
   // Check for any changes in the cluster topology.
   // Under the hood this calls cluster slots
    clusterClient.refreshPartitions();
    Set<String> nodes = new HashSet<>();

    for (RedisClusterNode node : clusterClient.getPartitions().getPartitions()) {
        nodes.add(node.getUri().getHost());
    }

    Thread.sleep(1000);
```

```
} while (!nodes.stream().allMatch(node -> {
            try {
                return finalTargetProtocolType.isInstance(InetAddress.getByName(node));
            } catch (UnknownHostException ignored) {}
            return false;
}));
```

TLS mengaktifkan cluster tumpukan ElastiCache ganda

Saat TLS diaktifkan untuk ElastiCache cluster, fungsi penemuan klaster (`cluster slots`,`cluster shards`, dan`cluster nodes`) mengembalikan nama host alih-alih IP. Nama host kemudian digunakan sebagai pengganti IP untuk terhubung ke ElastiCache cluster dan melakukan jabat tangan TLS. Hal ini berarti bahwa klien tidak akan terpengaruh oleh parameter Penemuan IP. Untuk klaster dengan TLS diaktifkan, parameter Penemuan IP tidak berpengaruh pada protokol IP pilihan. Sebagai gantinya, protokol IP yang digunakan akan ditentukan berdasarkan protokol IP mana yang lebih dipilih klien saat meresolusi nama host DNS.

Klien Java

Saat menghubungkan dari lingkungan Java yang mendukung IPv4 dan IPv6, Java secara default akan lebih memilih IPv4 daripada IPv6 untuk kompatibilitas mundur. Namun, preferensi protokol IP dapat dikonfigurasi melalui argumen JVM. Untuk memilih IPv4, JVM menerima `-Djava.net.preferIPv4Stack=true` dan untuk memilih IPv6, JVM mengatur `-Djava.net.preferIPv6Stack=true`. Pengaturan `-Djava.net.preferIPv4Stack=true` berarti bahwa JVM tidak akan lagi membuat koneksi IPv6. Ini termasuk koneksi ke aplikasi non-Redis lainnya.

Preferensi Tingkat Host

Secara umum, jika klien atau runtime klien tidak menyediakan opsi konfigurasi untuk mengatur preferensi protokol IP, saat melakukan resolusi DNS, protokol IP akan bergantung pada konfigurasi host. Secara default, sebagian besar host lebih memilih IPv6 daripada IPv4, tetapi preferensi ini dapat dikonfigurasi di tingkat host. Ini akan memengaruhi semua permintaan DNS dari host itu, bukan hanya permintaan ke ElastiCache cluster.

Host Linux

Untuk Linux, preferensi protokol IP dapat dikonfigurasi dengan memodifikasi file `gai.conf`. File `gai.conf` dapat ditemukan dalam `/etc/gai.conf`. Jika tidak ada `gai.conf` yang ditentukan, maka contohnya akan tersedia di `/usr/share/doc/glibc-common-x.xx/gai.conf` yang dapat

disalin ke `/etc/gai.conf` lalu konfigurasi default-nya harus di-uncommenting. Untuk memperbarui konfigurasi agar lebih memilih IPv4 saat menghubungkan ke ElastiCache kluster, perbarui prioritas untuk rentang CIDR yang mencakup IP cluster berada di atas prioritas untuk koneksi IPv6 default. Secara default, koneksi IPv6 memiliki prioritas 40. Misalnya, anggaplah klaster berada di subnet dengan CIDR 172.31.0.0:0/16, konfigurasi di bawah ini akan menyebabkan klien lebih memilih koneksi IPv4 ke klaster tersebut.

```
label ::1/128       0
label ::/0          1
label 2002::/16     2
label ::/96         3
label ::ffff:0:0/96 4
label fec0::/10     5
label fc00::/7      6
label 2001:0::/32   7
label ::ffff:172.31.0.0/112 8
#
#    This default differs from the tables given in RFC 3484 by handling
#    (now obsolete) site-local IPv6 addresses and Unique Local Addresses.
#    The reason for this difference is that these addresses are never
#    NATed while IPv4 site-local addresses most probably are.  Given
#    the precedence of IPv6 over IPv4 (see below) on machines having only
#    site-local IPv4 and IPv6 addresses a lookup for a global address would
#    see the IPv6 be preferred.  The result is a long delay because the
#    site-local IPv6 addresses cannot be used while the IPv4 address is
#    (at least for the foreseeable future) NATed.  We also treat Teredo
#    tunnels special.
#
# precedence  <mask>   <value>
#    Add another rule to the RFC 3484 precedence table.  See section 2.1
#    and 10.3 in RFC 3484.  The default is:
#
precedence  ::1/128       50
precedence  ::/0          40
precedence  2002::/16     30
precedence ::/96          20
precedence ::ffff:0:0/96  10
precedence ::ffff:172.31.0.0/112 100
```

Detail selengkapnya tentang `gai.conf` tersedia di Halaman utama Linux.

Host Windows

Proses untuk host Windows juga serupa. Untuk host Windows, Anda dapat menjalankan `netsh interface ipv6 set prefix CIDR_CONTAINING_CLUSTER_IPS PRECEDENCE LABEL`. Hal ini memiliki efek yang sama seperti memodifikasi file `gai.conf` pada host Linux.

Hal ini akan memperbarui kebijakan preferensi untuk memilih koneksi IPv4 daripada koneksi IPv6 untuk rentang CIDR yang ditentukan. Misalnya, dengan asumsi bahwa klaster berada dalam subnet dengan CIDR 172.31.0.0:0/16, eksekusi `netsh interface ipv6 set prefix ::ffff:172.31.0.0:0/112 100 15` akan menghasilkan tabel prioritas berikut yang akan menyebabkan klien lebih memilih IPv4 saat terhubung ke klaster.

```
C:\Users\Administrator>netsh interface ipv6 show prefixpolicies
Querying active state...

Precedence Label Prefix
---------- ----- --------------------------------
100 15 ::ffff:172.31.0.0:0/112
20 4 ::ffff:0:0/96
50 0 ::1/128
40 1 ::/0
30 2 2002::/16
5 5 2001::/32
3 13 fc00::/7
1 11 fec0::/10
1 12 3ffe::/16
1 3 ::/96
```

# Praktik terbaik saat bekerja dengan klaster yang dirancang sendiri

Bagian ini hanya berlaku ketika Anda memilih untuk mendesain klaster Redis Anda sendiri. Sebaiknya tinjau dan ikuti praktik terbaik ini.

Topik



## Meminimalkan waktu henti dengan Multi-AZ

Lihat Meminimalkan waktu henti di ElastiCache for Redis dengan Multi-AZ, untuk mempelajari Multi-AZ selengkapnya dan meminimalkan waktu henti.

## Memastikan bahwa Anda memiliki cukup memori untuk membuat snapshot Redis

Snapshot dan sinkronisasi Redis pada versi 2.8.22 dan yang lebih baru

Redis 2.8.22 memperkenalkan proses simpan forkless yang memungkinkan Anda mengalokasikan lebih banyak memori Anda untuk penggunaan aplikasi Anda tanpa menimbulkan peningkatan penggunaan swap selama sinkronisasi dan penyimpanan. Untuk informasi selengkapnya, lihat Cara penerapan sinkronisasi dan pencadangan.

Snapshot dan sinkronisasi Redis sebelum versi 2.8.22

Jika Anda bekerja dengan Redis ElastiCache, Redis memanggil perintah tulis di latar belakang dalam sejumlah kasus:

- Saat membuat snapshot untuk cadangan.
- Saat menyinkronkan replika dengan primer dalam grup replikasi.
- Saat mengaktifkan fitur file tambah-saja (AOF) untuk Redis.
- Saat mempromosikan replika menjadi primer (yang menyebabkan sinkronisasi primer/replika).

Setiap kali Redis mengeksekusi proses tulis di latar belakang, Anda harus memiliki memori tersedia yang cukup untuk mengakomodasi proses overhead. Jika memori yang tersedia tidak cukup, proses akan gagal. Oleh karena itu, penting untuk memilih jenis instans simpul yang memiliki cukup memori saat membuat klaster Redis Anda.

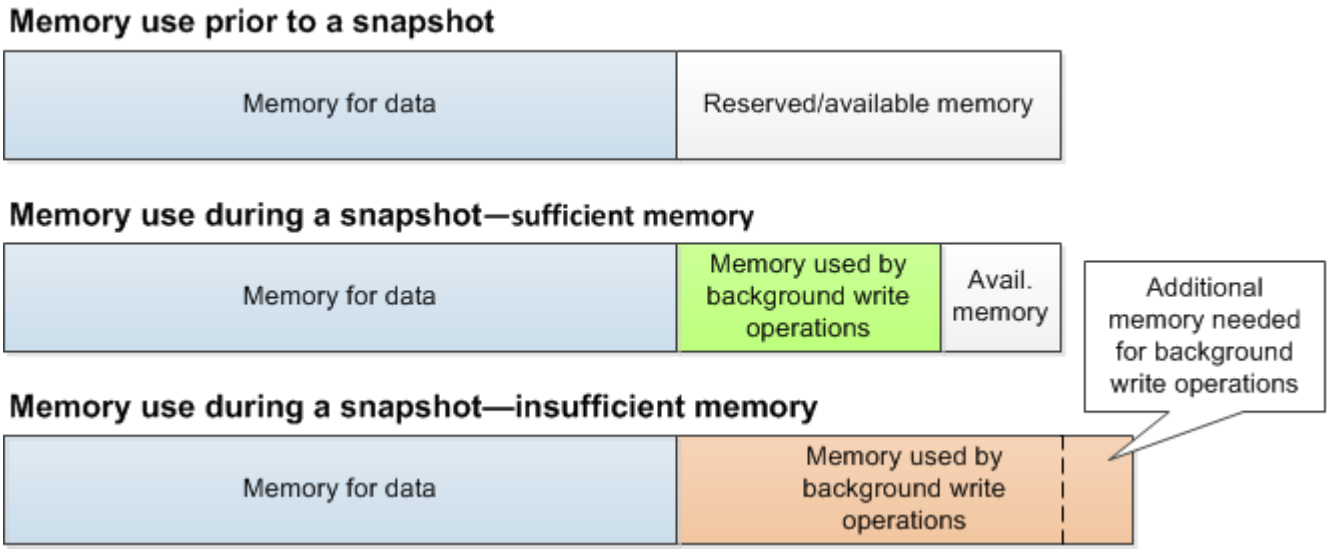
Proses Tulis di Latar Belakang dan Penggunaan Memori

Setiap kali proses tulis di latar belakang dipanggil, Redis melakukan fork pada prosesnya (ingat, Redis adalah thread tunggal). Satu fork menyimpan data Anda ke disk dalam file snapshot .rdb Redis. Layanan fork lain semuanya melakukan operasi baca dan tulis. Untuk memastikan bahwa snapshot Anda adalah snapshot di suatu titik waktu, semua pembaruan dan penambahan data ditulis ke area dengan memori tersedia yang terpisah dari area data.

Selama Anda memiliki cukup memori tersedia untuk merekam semua operasi tulis saat data sedang disimpan ke disk, Anda tidak akan mengalami masalah kekurangan memori. Anda mungkin mengalami masalah kekurangan memori jika mengalami salah satu di bawah ini:

- Aplikasi Anda melakukan banyak operasi tulis, sehingga membutuhkan sejumlah besar memori tersedia untuk menerima data baru atau pembaruan data.
- Anda memiliki sedikit memori yang tersedia untuk data yang ditulis dan diperbarui.
- Anda memiliki set data besar yang membutuhkan waktu lama untuk disimpan ke disk, sehingga memerlukan sejumlah besar operasi tulis.

Diagram berikut menggambarkan penggunaan memori saat menjalankan proses tulis di latar belakang.



Untuk informasi tentang dampak melakukan pencadangan pada performa, lihat Dampak performa pencadangan klaster yang dirancang sendiri.

Untuk informasi selengkapnya tentang cara Redis melakukan snapshot, lihat http://redis.io.

Untuk informasi selengkapnya tentang wilayah dan Zona Ketersediaan, lihat Memilih wilayah dan zona ketersediaan.

Menghindari kehabisan memori saat menjalankan proses tulis di latar belakang

Setiap kali proses tulis di latar belakang seperti BGSAVE atau BGREWRITEAOF dipanggil, untuk mencegah proses gagal, Anda harus memiliki lebih banyak memori tersedia dibandingkan yang akan dikonsumsi oleh operasi tulis selama proses. Skenario terburuknya adalah bahwa selama operasi tulis di latar belakang, setiap catatan Redis diperbarui dan beberapa catatan baru ditambahkan ke cache. Oleh karena itu, sebaiknya atur `reserved-memory-percent` ke nilai 50 (50 persen) untuk versi Redis sebelum 2.8.22 atau ke nilai 25 (25 persen) untuk Redis versi 2.8.22 dan yang lebih baru.

Nilai `maxmemory` menunjukkan memori yang tersedia untuk overhead data dan operasional. Karena Anda tidak dapat mengubah parameter `reserved-memory` dalam grup parameter default, Anda harus membuat grup parameter khusus untuk klaster. Nilai default untuk `reserved-memory` adalah 0, yang memungkinkan Redis untuk mengonsumsi semua dari maxmemory dengan data, sehingga berpotensi meninggalkan terlalu sedikit memori untuk kegunaan lain, seperti proses tulis di latar belakang. Untuk nilai `maxmemory` berdasarkan jenis instans simpul, lihat Parameter khusus jenis simpul Redis.

Anda juga dapat menggunakan parameter `reserved-memory` untuk mengurangi jumlah memori yang digunakan oleh Redis pada sistem.

Untuk informasi selengkapnya tentang parameter khusus Redis di ElastiCache, lihat Parameter spesifik Redis.

Untuk informasi tentang cara membuat dan mengubah grup parameter DB, lihat Membuat grup parameter dan Mengubah grup parameter.

# Mengelola Memori Terpesan

Memori terpesan adalah memori yang disisihkan untuk penggunaan nondata. Saat melakukan cadangan atau failover, Redis menggunakan memori yang tersedia untuk merekam operasi tulis ke klaster Anda sementara data klaster sedang ditulis ke file .rdb. Jika Anda tidak memiliki memori yang mencukup untuk semua proses tulis, proses akan gagal. Selanjutnya, Anda dapat menemukan informasi tentang opsi untuk mengelola memori terpesan untuk ElastiCache for Redis dan cara menerapkan opsi tersebut.

Topik



## Berapa Banyak Memori Terpesan yang Dibutuhkan?

Jika Anda menjalankan versi Redis sebelum 2.8.22, sisihkan lebih banyak memori untuk cadangan dan failover daripada jika Anda menjalankan Redis 2.8.22 atau yang lebih baru. Persyaratan ini adalah karena perbedaan cara implementasi proses pencadangan ElastiCache for Redis. Rumus praktisnya adalah mencadangkan setengah dari nilai `maxmemory` dari jenis simpul untuk overhead Redis kepada versi sebelum 2.8.22, dan seperempat untuk versi Redis 2.8.22 dan yang lebih baru.

Ketika menggunakan klaster dengan tingkat data, kami sarankan meningkatkan maxmemory hingga setengah dari memori simpul yang tersedia Anda jika beban kerja Anda adalah tulis berat.

Untuk informasi selengkapnya, lihat berikut ini:

- Memastikan bahwa Anda memiliki cukup memori untuk membuat snapshot Redis
- Cara penerapan sinkronisasi dan pencadangan
- Tingkatan data

## Parameter untuk Mengelola Memori Terpesan

Mulai tanggal 16 Maret 2017, Amazon ElastiCache for Redis menyediakan dua parameter yang saling eksklusif untuk mengelola memori Redis Anda, `reserved-memory` dan `reserved-memory-percent`. Kedua parameter ini bukan merupakan bagian dari distribusi Redis.

Tergantung pada kapan Anda menjadi pelanggan ElastiCache, salah satu dari parameter ini adalah parameter pengelolaan memori default. Parameter ini berlaku saat Anda membuat klaster atau grup replikasi Redis yang baru dan menggunakan grup parameter default.

- Untuk pelanggan yang mulai berlangganan sebelum 16 Maret 2017 – Saat Anda membuat klaster atau grup replikasi Redis menggunakan grup parameter default, parameter pengelolaan memori Anda adalah `reserved-memory`. Dalam hal ini, memori yang terpesan adalah nol (0) byte.
- Untuk pelanggan yang mulai berlangganan pada atau setelah 16 Maret 2017 – Saat Anda membuat klaster atau grup replikasi Redis menggunakan grup parameter default, parameter pengelolaan memori Anda adalah `reserved-memory-percent`. Dalam hal ini, 25 persen dari nilai `maxmemory` simpul Anda terpesan untuk tujuan nondata.

Setelah membaca tentang dua parameter manajemen memori Redis, Anda mungkin lebih memilih untuk menggunakan salah satu yang bukan default Anda atau dengan nilai non-default. Jika demikian, Anda dapat mengubah ke parameter manajemen memori cadangan yang lain.

Untuk mengubah nilai parameter tersebut, Anda dapat membuat grup parameter khusus dan mengubahnya agar menggunakan parameter dan nilai manajemen memori pilihan Anda. Anda kemudian dapat menggunakan grup parameter khusus setiap kali Anda membuat klaster atau grup replikasi Redis yang baru. Untuk klaster atau grup replikasi yang telah ada, Anda dapat mengubahnya untuk menggunakan grup parameter khusus Anda.

Untuk informasi selengkapnya, lihat berikut ini:

- Menentukan Parameter Manajemen Memori Terpesan Anda
- Membuat grup parameter
- Mengubah grup parameter
- Memodifikasi sebuah cluster ElastiCache
- Mengubah grup replikasi

### Parameter reserved-memory

Sebelum 16 Maret 2017, semua pengelolaan memori terpesan ElastiCache for Redis dilakukan menggunakan parameter `reserved-memory`. Nilai default `reserved-memory` adalah 0. Default ini tidak menyisihkan memori untuk overhead Redis dan memungkinkan Redis untuk menggunakan semua memori simpul dengan data.

Mengubah `reserved-memory` agar Anda memiliki cukup memori tersedia untuk backup dan failover mengharuskan Anda untuk membuat grup parameter khusus. Pada grup parameter khusus ini, Anda menetapkan `reserved-memory` ke nilai yang sesuai untuk versi Redis yang berjalan pada klaster dan jenis simpul dari klaster Anda. Untuk informasi selengkapnya, lihat Berapa Banyak Memori Terpesan yang Dibutuhkan?

Parameter `reserved-memory` ElastiCache for Redis bersifat khusus untuk ElastiCache for Redis dan bukan bagian dari distribusi Redis.

Prosedur berikut menunjukkan cara menggunakan `reserved-memory` untuk mengelola memori di klaster Redis Anda.

Untuk menyisihkan memori menggunakan reserved-memory

1. Buat grup parameter khusus yang menentukan keluarga grup parameter yang sesuai dengan versi mesin yang Anda jalankan—misalnya, menentukan keluarga grup parameter `redis2.8`. Untuk informasi selengkapnya, lihat Membuat grup parameter.

   ```
   aws elasticache create-cache-parameter-group \
      --cache-parameter-group-name redis6x-m3xl \
      --description "Redis 2.8.x for m3.xlarge node type" \
      --cache-parameter-group-family redis6.x
   ```

2. Hitung jumlah byte memori yang akan disisihkan untuk overhead Redis. Anda dapat menemukan nilai `maxmemory` untuk jenis simpul Anda di Parameter khusus jenis simpul Redis.
3. Ubah grup parameter khusus sehingga parameter `reserved-memory` adalah jumlah byte yang Anda hitung pada langkah sebelumnya. Contoh AWS CLI berikut mengasumsikan bahwa Anda menjalankan versi Redis sebelum 2.8.22 dan perlu menyisihkan setengah dari `maxmemory` simpul. Untuk informasi selengkapnya, lihat Mengubah grup parameter.

   ```
   aws elasticache modify-cache-parameter-group \
      --cache-parameter-group-name redis28-m3xl \
      --parameter-name-values "ParameterName=reserved-memory,
    ParameterValue=7130316800"
   ```

   Anda memerlukan grup parameter khusus terpisah untuk setiap jenis simpul yang Anda gunakan, karena setiap jenis simpul memiliki nilai `maxmemory` yang berbeda. Dengan demikian, setiap jenis simpul membutuhkan nilai yang berbeda untuk `reserved-memory`.
4. Ubah klaster atau grup replikasi Redis Anda untuk menggunakan grup parameter khiusus Anda.

Contoh CLI berikut mengubah klaster `my-redis-cluster` untuk menggunakan grup parameter khusus `redis28-m3xl` yang dimulai segera. Untuk informasi selengkapnya, lihat Memodifikasi sebuah cluster ElastiCache .

```
aws elasticache modify-cache-cluster \
   --cache-cluster-id my-redis-cluster \
   --cache-parameter-group-name redis28-m3xl \
   --apply-immediately
```

Contoh CLI berikut mengubah grup relikasi `my-redis-repl-grp` untuk menggunakan grup parameter khusus `redis28-m3xl` yang dimulai segera. Untuk informasi selengkapnya, Mengubah grup replikasi.

```
aws elasticache modify-replication-group \
   --replication-group-id my-redis-repl-grp \
   --cache-parameter-group-name redis28-m3xl \
   --apply-immediately
```

Parameter reserved–memory-percent

Pada tanggal 16 Maret 2017, Amazon ElastiCache memperkenalkan parameter `reserved-memory-percent` dan membuatnya tersedia di semua versi ElastiCache for Redis. Tujuan dari `reserved-memory-percent` adalah untuk menyederhanakan manajemen memori terpesan di semua klaster Anda. Hal ini dilakukan dengan memungkinkan Anda untuk memiliki satu grup parameter untuk setiap keluarga grup parameter (seperti `redis2.8`) untuk mengelola memori terpesan dari klaster Anda, terlepas dari jenis simpulnya. Nilai default untuk `reserved-memory-percent` adalah 25 (25 persen).

Parameter `reserved-memory-percent` ElastiCache for Redis bersifat khusus untuk ElastiCache for Redis dan bukan bagian dari distribusi Redis.

Jika cluster Anda menggunakan jenis simpul dari keluarga r6gd dan penggunaan memori Anda mencapai 75 persen, tingkat data akan secara otomatis dipicu. Untuk informasi selengkapnya, lihat Tingkatan data.

Untuk menyisihkan memori menggunakan reserved-memory-percent

Untuk menggunakan `reserved-memory-percent` guna mengelola memori pada klaster ElastiCache for Redis Anda, lakukan salah satu hal berikut:

- Jika Anda menjalankan Redis 2.8.22 atau yang lebih baru, tetapkan grup parameter default untuk klaster Anda. Default 25 persen seharusnya sudah memadai. Jika tidak, lakukan langkah yang dijelaskan berikut untuk mengubah nilai.
- Jika Anda menjalankan versi Redis sebelum 2.8.22, Anda mungkin perlu untuk menyimpan lebih banyak memori dari default 25 persen `reserved-memory-percent`. Untuk melakukannya, gunakan prosedur berikut.

Untuk mengubah nilai persen dari reserved-memory-percent

1. Buat grup parameter khusus yang menentukan keluarga grup parameter yang cocok dengan versi mesin yang Anda jalankan—misalnya, menentukan keluarga grup parameter `redis2.8`. Grup parameter khusus diperlukan karena Anda tidak dapat memodifikasi grup parameter default. Untuk informasi selengkapnya, lihat Membuat grup parameter.

   ```
   aws elasticache create-cache-parameter-group \
      --cache-parameter-group-name redis28-50 \
      --description "Redis 2.8.x 50% reserved" \
      --cache-parameter-group-family redis2.8
   ```

   Karena `reserved-memory-percent` menyisihkan memori sebagai persentase dari `maxmemory` simpul, Anda tidak memerlukan grup parameter khusus untuk setiap jenis simpul.

2. Ubah grup parameter khusus agar `reserved-memory-percent` menjadi 50 (50 persen). Untuk informasi selengkapnya, lihat Mengubah grup parameter.

   ```
   aws elasticache modify-cache-parameter-group \
      --cache-parameter-group-name redis28-50 \
      --parameter-name-values "ParameterName=reserved-memory-percent,
    ParameterValue=50"
   ```

3. Gunakan grup parameter khusus ini untuk setiap klaster atau grup replikasi Redis yang menjalankan versi Redis lebih lama dari 2.8.22.

   Contoh CLI berikut mengubah klaster `my-redis-cluster` Redis untuk menggunakan grup parameter khusus `redis28-50` yang dimulai segera. Untuk informasi selengkapnya, lihat Memodifikasi sebuah cluster ElastiCache .

```
aws elasticache modify-cache-cluster \
   --cache-cluster-id my-redis-cluster \
   --cache-parameter-group-name redis28-50 \
   --apply-immediately
```

Contoh CLI berikut mengubah grup relikasi `my-redis-repl-grp` agar menggunakan grup parameter khusus `redis28-50` yang dimulai segera. Untuk informasi selengkapnya, lihat Mengubah grup replikasi.

```
aws elasticache modify-replication-group \
   --replication-group-id my-redis-repl-grp \
   --cache-parameter-group-name redis28-50 \
   --apply-immediately
```

Menentukan Parameter Manajemen Memori Terpesan Anda

Jika Anda sudah menjadi pelanggan ElastiCache pada 16 Maret 2017, maka parameter manajemen memori cadangan default Anda adalah `reserved-memory` dengan memori terpesan nol (0) byte. Jika Anda menjadi pelanggan ElastiCache setelah 16 Maret 2017, maka parameter manajemen memori cadangan default Anda adalah `reserved-memory-percent` dengan 25 persen dari memori terpesan simpul. Hal ini berlaku terlepas dari waktu ketika Anda membuat klaster ElastiCache for Redis atau grup replikasi Anda. Akan tetapi, Anda dapat mengubah parameter manajemen memori cadangan Anda menggunakan AWS CLI atau API ElastiCache.

Parameter `reserved-memory` dan `reserved-memory-percent` keduanya bersifat saling eksklusif. Grup parameter selalu memiliki salah satu dari keduanya tetapi tidak keduanya sekaligus. Anda dapat mengubah parameter mana yang digunakan grup parameter untuk manajemen memori terpesan dengan memodifikasi grup parameter. Grup parameter harus berupa grup parameter khusus, karena Anda tidak dapat mengubah grup parameter default. Untuk informasi selengkapnya, lihat Membuat grup parameter.

Untuk menentukan reserved-memory-percent

Untuk menggunakan `reserved-memory-percent` sebagai parameter manajemen memori terpesan Anda, ubah grup parameter khusus menggunakan perintah `modify-cache-parameter-group`. Gunakan parameter `parameter-name-values` untuk menentukan `reserved-memory-percent` dan nilainya.

Contoh CLI berikut mengubah grup parameter khusus `redis32-cluster-on` sehingga menggunakan `reserved-memory-percent` untuk mengelola memori terpesan. Nilai `ParameterValue` harus ditetapkan agar grup parameter dapat menggunakan parameter `ParameterName` untuk manajemen memori terpesan. Untuk informasi selengkapnya, lihat Mengubah grup parameter.

```
aws elasticache modify-cache-parameter-group \
   --cache-parameter-group-name redis32-cluster-on \
   --parameter-name-values "ParameterName=reserved-memory-percent, ParameterValue=25"
```

Untuk menentukan reserved-memory

Untuk menggunakan `reserved-memory` sebagai parameter manajemen memori terpesan Anda, ubah grup parameter khusus menggunakan perintah `modify-cache-parameter-group`. Gunakan parameter `parameter-name-values` untuk menentukan `reserved-memory` dan nilainya.

Contoh CLI berikut mengubah grup parameter khusus `redis32-m3xl` sehingga menggunakan `reserved-memory` untuk mengelola memori terpesan. Nilai `ParameterValue` harus ditetapkan agar grup parameter dapat menggunakan parameter `ParameterName` untuk manajemen memori terpesan. Karena versi mesin lebih baru dari 2.8.22, kami menetapkan nilainya ke `3565158400`, yaitu 25 persen dari `maxmemory` `cache.m3.xlarge`. Untuk informasi selengkapnya, lihat Mengubah grup parameter.

```
aws elasticache modify-cache-parameter-group \
   --cache-parameter-group-name redis32-m3xl \
   --parameter-name-values "ParameterName=reserved-memory, ParameterValue=3565158400"
```

## Perubahan ukuran klaster online

Resharding melibatkan proses menambahkan dan menghapus serpihan atau simpul ke klaster Anda dan mendistribusikan ulang keyspace. Sebagai hasilnya, beberapa hal memiliki dampak pada operasi resharding, seperti beban pada klaster, pemanfaatan memori, dan ukuran keseluruhan data. Untuk pengalaman terbaik, sebaiknya ikuti praktik terbaik klaster secara keseluruhan untuk distribusi pola beban kerja seragam. Selain itu, sebaiknya lakukan beberapa langkah berikut.

Sebelum memulai resharding, sebaiknya lakukan yang berikut:

- Uji aplikasi Anda – Uji perilaku aplikasi Anda selama resharding di lingkungan pengujian jika memungkinkan.

- Dapatkan notifikasi awal untuk masalah penskalaan – Resharding adalah operasi komputasi yang intensif. Karena ini, sebaiknya jaga pemanfaatan CPU di bawah 80 persen pada instans multicore dan kurang dari 50 persen pada instans inti tunggal selama resharding. Pantau metrik ElastiCache untuk Redis dan mulai resharding sebelum aplikasi Anda mulai mengalami masalah penskalaan. Metrik yang berguna untuk melacak adalah `CPUUtilization`, `NetworkBytesIn`, `NetworkBytesOut`, `CurrConnections`, `NewConnections`, `FreeableMemory`, `SwapUsage`, dan `BytesUsedForCacheItems`.
- Pastikan untuk menyediakan memori yang cukup sebelum penskalaan ke dalam – Jika Anda melakukan penskalaan ke dalam, pastikan memori pada serpihan yang akan dipertahankan tersedia setidaknya 1,5 kali dari memori yang akan digunakan pada serpihan yang akan dihapus.
- Mulai resharding selama jam bukan puncak – Praktik ini membantu mengurangi dampak latensi dan throughput pada klien selama operasi resharding. Hal ini juga membantu untuk menyelesaikan resharding lebih cepat karena lebih banyak sumber daya dapat digunakan untuk distribusi ulang slot.
- Tinjau perilaku timeout klien – Beberapa klien mungkin mengalami latensi yang lebih tinggi selama perubahan ukuran klaster secara online. Anda dapat membuat konfigurasi pustaka klien dengan batas waktu yang lebih tinggi agar waktu sistem memiliki waktu untuk terhubung bahkan dalam kondisi beban yang lebih tinggi pada server. Dalam beberapa kasus, mungkin ada baiknya untuk membuka sejumlah besar koneksi ke server. Dalam kasus ini, pertimbangkan untuk menambahkan mundur eksponensial untuk menyambung logika kembali. Melakukan hal ini dapat membantu mencegah lonjakan koneksi baru di server pada waktu yang sama.
- Muat Fungsi Anda di setiap shard — Saat menskalakan klaster, ElastiCache akan secara otomatis mereplikasi Fungsi yang dimuat di salah satu simpul yang ada (dipilih secara acak) ke simpul baru. Jika klaster Anda memiliki Redis 7.0 atau yang lebih baru dan aplikasi Anda menggunakan Fungsi Redis, sebaiknya muat semua fungsi Anda ke semua serpihan sebelum melakukan penskalaan ke luar agar klaster Anda tidak berakhir dengan fungsi yang berbeda pada serpihan yang berbeda.

Setelah resharding, perhatikan hal berikut:

- Penskalaan ke dalam mungkin berhasil sebagian jika memori tidak cukup pada serpihan target. Jika hal tersebut terjadi, tinjau memori yang tersedia dan coba lagi operasi, jika perlu. Data pada serpihan target tidak akan dihapus.
- Slot dengan item besar tidak dimigrasikan. Secara khusus, slot dengan item yang lebih besar dari 256 MB pasca-serialisasi tidak dimigrasikan.

- Perintah FLUSHALL dan FLUSHDB tidak didukung di dalam skrip Lua selama operasi resharding. Sebelum Redis 6, perintah BRPOPLPUSH tidak didukung jika beroperasi pada slot yang dimigrasi.

## Meminimalkan waktu henti selama pemeliharaan

Konfigurasi mode klaster memiliki ketersediaan terbaik selama operasi terkelola atau tidak terkelola. Sebaiknya gunakan klien yang mendukung mode klaster yang menghubungkan ke titik akhir penemuan klaster. Untuk mode klaster dinonaktifkan, sebaiknya gunakan titik akhir primer untuk semua operasi tulis.

Untuk aktivitas baca, aplikasi juga dapat tersambung ke simpul apa pun di klaster. Tidak seperti titik akhir primer, titik akhir simpul diresolusi ke titik akhir tertentu. Jika Anda membuat perubahan di dalam klaster Anda, seperti menambahkan atau menghapus replika, Anda harus memperbarui titik akhir simpul di aplikasi Anda. Inilah sebabnya mengapa untuk mode klaster dinonaktifkan. Sebaiknya gunakan titik akhir pembaca untuk aktivitas baca.

Jika autofailover diaktifkan di klaster, simpul primer mungkin berubah. Oleh karena itu, aplikasi harus mengonfirmasi peran simpul dan memperbarui semua titik akhir baca. Melakukan hal ini akan membantu memastikan Anda tidak menyebabkan beban besar pada primer. Dengan autofailover dinonaktifkan, peran simpul tidak berubah. Namun, waktu henti dalam operasi terkelola atau tidak terkelola menjadi lebih tinggi dibandingkan klaster yang mengaktifkan auto failover.

Hindari mengarahkan permintaan baca ke satu simpul replika baca, karena ketidaktersediaannya dapat menyebabkan pemadaman baca. Anda dapat melakukan fallback untuk membaca dari primer, atau pastikan bahwa Anda memiliki setidaknya dua replika baca untuk menghindari gangguan proses baca selama pemeliharaan.

# Praktik terbaik Redis

Berikut ini adalah praktik terbaik saat menggunakan Redis untuk meningkatkan kinerja dan keandalan:

- Gunakan konfigurasi berkemampuan mode klaster – Mode klaster diaktifkan memungkinkan cache menskalakan secara horizontal untuk mencapai penyimpanan dan throughput yang lebih tinggi daripada konfigurasi dengan mode klaster dinonaktifkan. ElastiCache nirserver hanya tersedia dalam konfigurasi dengan mode klaster diaktifkan.
- Gunakan koneksi berumur panjang – Membuat koneksi baru itu mahal, serta membutuhkan waktu dan sumber daya CPU dari cache. Gunakan kembali koneksi jika memungkinkan (misalnya dengan penyatuan koneksi) untuk mengamortisasi biaya ini atas banyak perintah.
- Baca dari replika – Jika Anda menggunakan ElastiCache nirserver atau telah menyediakan replika baca (klaster yang dirancang sendiri), arahkan pembacaan ke replika untuk mencapai skalabilitas yang lebih baik dan/atau latensi yang lebih rendah. Pembacaan dari replika pada akhirnya akan konsisten dengan yang utama.

  Dalam klaster yang dirancang sendiri, hindari mengarahkan permintaan baca ke satu replika baca karena pembacaan mungkin tidak tersedia sementara jika simpul gagal. Konfigurasi klien Anda untuk mengarahkan permintaan baca ke setidaknya dua replika baca, atau langsung membaca ke replika tunggal dan replika utama.

  Di ElastiCache nirserver, pembacaan dari port replika (6380) akan mengarahkan pembacaan ke zona ketersediaan lokal klien jika memungkinkan, sehingga mengurangi latensi pengambilan. Hal ini akan secara otomatis melakukan fallback ke simpul lainnya selama kegagalan.
- Hindari perintah yang mahal – Hindari menjalankan operasi yang intensif dalam hal komputasi dan I/O, seperti perintah `KEYS` dan `SMEMBERS`. Pendekatan ini disarankan karena operasi ini meningkatkan beban pada klaster dan memiliki dampak pada performa klaster. Sebagai gantinya, gunakan perintah `SCAN` dan `SSCAN`.
- Ikuti praktik terbaik Lua – Hindari menjalankan script Lua terlalu lama, dan selalu nyatakan kunci yang digunakan dalam script Lua di depan. Pendekatan ini disarankan untuk menentukan bahwa script Lua tidak menggunakan perintah cross slot. Pastikan bahwa kunci yang digunakan dalam skrip Lua adalah milik slot yang sama.
- Gunakan pub/sub sharded - Saat menggunakan Redis untuk mendukung beban kerja pub/sub dengan throughput tinggi, sebaiknya gunakan pub/sub sharded (tersedia dengan Redis 7 atau yang lebih baru). Pub/sub tradisional dalam klaster dengan mode klaster yang diaktifkan akan menyiarkan pesan ke semua simpul di klaster, yang dapat menghasilkan

`EngineCPUUtilization` yang tinggi. Perhatikan bahwa di ElastiCache nirserver, perintah pub/sub tradisional secara internal menggunakan perintah pub/sub sharded.

# Strategi Pembuatan Cache

Pada topik berikut ini, Anda dapat menemukan strategi untuk mengisi dan memelihara cache Anda.

Strategi yang akan diterapkan untuk mengisi dan memelihara cache Anda akan tergantung pada data apa yang dibuat cache dan pola akses ke data itu. Misalnya, Anda kemungkinan tidak ingin menggunakan strategi yang sama untuk papan peringkat 10 teratas di situs game dan berita yang sedang tren. Di bagian yang tersisa dari bagian ini, kita akan membahas strategi pemeliharaan cache yang umum dengan kelebihan dan kekurangannya.

Topik



## Pemuatan malas

Seperti namanya, pemuatan malas adalah strategi pembuatan cache yang memuat data ke dalam cache hanya jika diperlukan. Cara kerjanya adalah seperti dijelaskan berikut.

Amazon ElastiCache adalah penyimpanan nilai-kunci dalam memori yang berada di antara aplikasi Anda dan penyimpanan data (basis data) yang diaksesnya. Setiap kali aplikasi Anda meminta data, aplikasi pertama-tama akan membuat permintaan ke cache ElastiCache. Jika data terdapat di dalam cache dan data itu masih baru, maka ElastiCache mengembalikan data itu ke aplikasi Anda. Jika data tidak ada di dalam cache atau telah kedaluwarsa, maka aplikasi Anda akan meminta data dari penyimpanan data Anda. Penyimpanan data Anda kemudian mengembalikan data ke aplikasi Anda. Aplikasi Anda selanjutnya menulis data yang diterima dari penyimpanan ke cache. Dengan cara ini, data dapat lebih cepat diambil jika diminta lagi pada waktu berikutnya.

Cache kena terjadi ketika data ada di dalam cache dan tidak kedaluwarsa:

1. Aplikasi Anda meminta data dari cache.
2. Cache mengembalikan data ke aplikasi.

Cache meleset terjadi ketika data tidak berada di dalam cache dan kedaluwarsa:

1. Aplikasi Anda meminta data dari cache.
2. Cache tidak memiliki data yang diminta, sehingga mengembalikan `null`.
3. Aplikasi Anda meminta dan menerima data dari basis data.
4. Aplikasi Anda memperbarui cache dengan data baru.

### Kelebihan dan kekurangan dari pemuatan malas

Kelebihan dari pemuatan malas adalah seperti berikut:

- Hanya data yang diminta yang dibuat cache.

  Karena sebagian besar data tidak pernah diminta, pemuatan malas menghindari membuat cache terisi penuh oleh data yang tidak diminta.
- Kegagalan simpul tidak menjadi fatal untuk aplikasi Anda.

  Ketika simpul gagal dan digantikan oleh simpul baru yang kosong, aplikasi Anda terus berfungsi, meskipun dengan peningkatan latensi. Karena permintaan dibuat ke simpul baru, setiap terjadi cache meleset mengakibatkan kueri ke basis data. Pada saat yang sama, salinan data ditambahkan ke cache sehingga permintaan berikutnya diambil dari cache.

Kelebihan dari pemuatan malas adalah seperti berikut:

- Terjadi kerugian akibat cache meleset. Setiap cache meleset menyebabkan tiga perjalanan:
  1. Permintaan awal untuk data dari cache
  2. Kueri basis data untuk data
  3. Menulis data ke cache

  Kejadian meleset ini dapat menyebabkan penundaan yang terasa pada data yang masuk ke aplikasi.
- Data yang usang.

  Jika data ditulis ke cache hanya bila ada cache meleset, maka data di dalam cache dapat menjadi usang. Hal ini terjadi karena ada tidak ada pembaruan pada cache ketika data berubah di dalam basis data. Untuk mengatasi masalah ini, Anda dapat menggunakan strategi Write-through dan Menambahkan TTL.

### Contoh kode semu pemuatan malas

Berikut adalah contoh kode semu logika pemuatan malas.

```
// ****************************************
// function that returns a customer's record.
// Attempts to retrieve the record from the cache.
// If it is retrieved, the record is returned to the application.
// If the record is not retrieved from the cache, it is
//    retrieved from the database,
//    added to the cache, and
//    returned to the application
// ****************************************
get_customer(customer_id)

    customer_record = cache.get(customer_id)
    if (customer_record == null)

        customer_record = db.query("SELECT * FROM Customers WHERE id = {0}",
 customer_id)
        cache.set(customer_id, customer_record)

    return customer_record
```

Untuk contoh ini, kode aplikasi yang mengambil data adalah sebagai berikut.

```
customer_record = get_customer(12345)
```

## Write-through

Strategi write-through menambahkan data atau pembaruan data ke dalam cache setiap kali data ditulis ke basis data.

### Kelebihan dan kekurangan dari write-through

Kelebihan dari write-through adalah sebagai berikut:

- Data dalam cache tidak pernah usang.

  Karena data dalam cache diperbarui setiap kali data itu ditulis ke basis data, maka data dalam cache selalu yang terbaru.

- Kerugian tulis vs kerugian baca.

  Setiap proses tulis menyangkut dua perjalanan:

  1. Tulis ke cache
  2. Tulis ke basis data

  Yang menambahkan latensi pada proses. Namun, pengguna akhir umumnya lebih toleran terhadap latensi saat memperbarui data daripada saat mengambil data. Ada perasaan yang melekat bahwa memperbarui membutuhkan upaya lebih banyak dan karena itu memakan waktu lebih lama.

Kekurangan dari write-through adalah sebagai berikut:

- Data yang hilang.

  Jika Anda membuat simpul baru, dengan alasan kegagalan simpul atau untuk penskalaan ke luar, maka akan ada data yang hilang. Data ini tetap hilang hingga ditambahkan atau diperbarui pada basis data. Anda dapat meminimalkan ini dengan menerapkan pemuatan malas dengan write-through.
- Churn Cache.

  Sebagian besar data tidak pernah dibaca, yang merupakan pemborosan sumber daya. Dengan menambahkan nilai waktu untuk tayang (TTL), Anda dapat meminimalkan ruang penyimpanan yang terbuang.

### Contoh kode semu Write-through

Berikut adalah contoh kode semu dari logika write-through.

```
// ****************************************
// function that saves a customer's record.
// ****************************************
save_customer(customer_id, values)

    customer_record = db.query("UPDATE Customers WHERE id = {0}", customer_id, values)
    cache.set(customer_id, customer_record)
    return success
```

Untuk contoh ini, kode aplikasi yang mengambil data adalah sebagai berikut.

```
save_customer(12345,{"address":"123 Main"})
```

## Menambahkan TTL

Pemuatan malas memungkinkan data menjadi usang tetapi tidak gagal dengan simpul kosong. Write-through memastikan data selalu segar, tetapi dapat gagal dengan simpul kosong dan dapat mengisi cache dengan data berlebihan yang tidak berguna. Dengan menambahkan nilai waktu untuk tayang (TTL) untuk setiap proses tulis, Anda dapat memiliki kelebihan dari setiap strategi. Pada saat yang sama, Anda dapat dan secara garis besar berhasil menghindari cache menjadi penuh dengan data tambahan.

Waktu untuk tayang (TTL) adalah nilai integer yang menentukan jumlah detik hingga kunci kedaluwarsa. Redis dapat menentukan detik atau milidetik untuk nilai ini. Ketika sebuah aplikasi mencoba untuk membaca kunci yang kedaluwarsa, maka perlakuannya adalah seolah-olah kunci itu tidak ditemukan. Terjadi kueri ke basis data untuk meminta kunci dan cache yang diperbarui. Pendekatan ini tidak menjamin sebuah nilai tidak menjadi usang. Namun, itu menjaga data agar tidak terlalu usang dan mensyaratkan nilai di dalam cache agar kadang-kadang disegarkan kembali dari basis data.

Untuk informasi selengkapnya, lihat perintah `set` Redis .

Contoh kode semu TTL

Berikut adalah contoh kode semu dari logika write-through dengan TTL.

```
// ****************************************
// function that saves a customer's record.
// The TTL value of 300 means that the record expires
//    300 seconds (5 minutes) after the set command
//    and future reads will have to query the database.
// ****************************************
save_customer(customer_id, values)

    customer_record = db.query("UPDATE Customers WHERE id = {0}", customer_id, values)
    cache.set(customer_id, customer_record, 300)

    return success
```

Berikut adalah contoh kode semu logika pemuatan malas dengan TTL.

```
// ****************************************
// function that returns a customer's record.
// Attempts to retrieve the record from the cache.
// If it is retrieved, the record is returned to the application.
// If the record is not retrieved from the cache, it is
//    retrieved from the database,
//    added to the cache, and
//    returned to the application.
// The TTL value of 300 means that the record expires
//    300 seconds (5 minutes) after the set command
//    and subsequent reads will have to query the database.
// ****************************************
get_customer(customer_id)

    customer_record = cache.get(customer_id)

    if (customer_record != null)
        if (customer_record.TTL < 300)
            return customer_record        // return the record and exit function

    // do this only if the record did not exist in the cache OR
    //    the TTL was >= 300, i.e., the record in the cache had expired.
    customer_record = db.query("SELECT * FROM Customers WHERE id = {0}", customer_id)
    cache.set(customer_id, customer_record, 300)  // update the cache
    return customer_record                // return the newly retrieved record and exit
 function
```

Untuk contoh ini, kode aplikasi yang mengambil data adalah sebagai berikut.

```
save_customer(12345,{"address":"123 Main"})
```

```
customer_record = get_customer(12345)
```

## Topik terkait

- Penyimpanan Data Dalam Memori
- Memilih mesin dan versi
- Penskalaan ElastiCache untuk Redis

# Mengelola klaster yang dirancang sendiri

Bagian-bagian ini berisi topik-topik yang membantu Anda mengelola klaster yang dirancang sendiri.

> **Note**
>
> Topik-topik ini tidak berlaku untuk ElastiCache Nirserver.

Topik



## Auto Scaling ElastiCache untuk kluster Redis

### Prasyarat

ElastiCache untuk Redis Auto Scaling terbatas pada hal-hal berikut:

- Klaster Redis (mode klaster diaktifkan) yang menjalankan mesin Redis versi 6.0 dan seterusnya
- Tingkatan data (mode klaster diaktifkan) yang menjalankan mesin Redis versi 7.0.7 dan yang lebih baru
- Ukuran instans - Large, XLarge, 2XLarge
- Kelompok tipe instans - R7g, R6g, R6gd, R5, M7g, M6g, M5, C7gn
- Auto Scaling in ElastiCache untuk Redis tidak didukung untuk cluster yang berjalan di Datastores Global, Outposts atau Local Zones.

### Mengelola Kapasitas Secara Otomatis dengan ElastiCache Redis Auto Scaling

ElastiCache untuk Redis auto scaling adalah kemampuan untuk menambah atau mengurangi pecahan atau replika yang diinginkan dalam layanan ElastiCache untuk Redis Anda secara otomatis.

ElastiCache untuk Redis memanfaatkan layanan Application Auto Scaling untuk menyediakan fungsionalitas ini. Untuk informasi lebih lanjut, lihat Application Auto Scaling. Untuk menggunakan penskalaan otomatis, Anda menentukan dan menerapkan kebijakan penskalaan yang menggunakan CloudWatch metrik dan nilai target yang Anda tetapkan. ElastiCache untuk Redis auto scaling menggunakan kebijakan untuk menambah atau mengurangi jumlah instance sebagai respons terhadap beban kerja aktual.

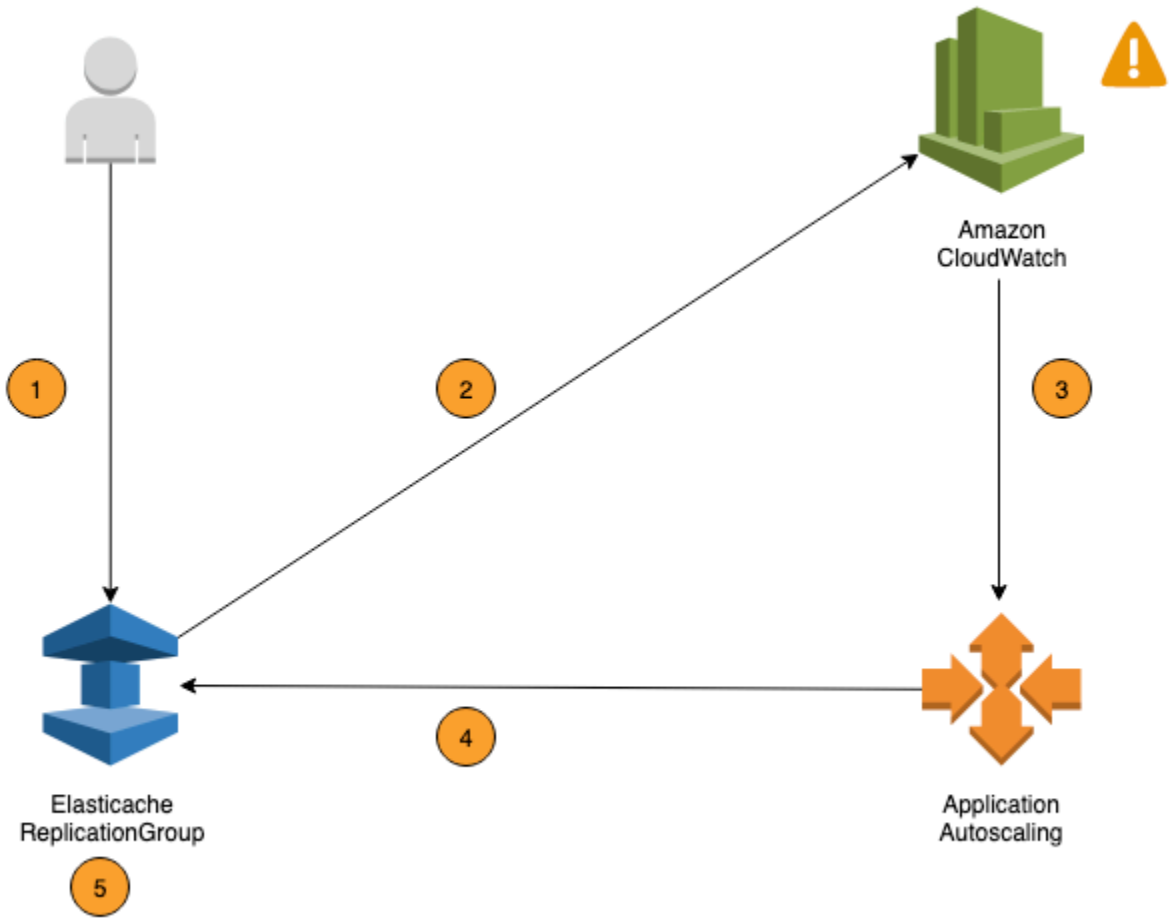
Anda dapat menggunakan AWS Management Console untuk menerapkan kebijakan penskalaan berdasarkan metrik yang telah ditentukan sebelumnya. Sebuah `predefined metric` didefinisikan dalam penghitungan sehingga Anda dapat menentukannya dengan nama dalam kode atau menggunakannya dalam AWS Management Console. Metrik kustom tidak tersedia untuk pilihan menggunakan AWS Management Console. Atau, Anda dapat menggunakan Application Auto Scaling API AWS CLI atau Application Auto Scaling untuk menerapkan kebijakan penskalaan berdasarkan metrik yang telah ditentukan atau kustom.

ElastiCache untuk Redis mendukung penskalaan untuk dimensi berikut:

- Serpihan – Secara otomatis menambahkan/menghapus serpihan di klaster yang mirip dengan resharding online secara manual. Dalam hal ini, ElastiCache untuk Redis auto scaling memicu penskalaan atas nama Anda.
- Replika – Secara otomatis menambahkan/menghapus replika di klaster yang mirip dengan operasi Meningkatkan/Menurunkan replika secara manual. ElastiCache untuk penskalaan otomatis Redis menambah/menghapus replika secara seragam di semua pecahan di cluster.

ElastiCache untuk Redis mendukung jenis kebijakan penskalaan otomatis berikut:

- Kebijakan penskalaan pelacakan target – Tingkatkan atau turunkan jumlah serpihan/replika yang dijalankan layanan Anda berdasarkan nilai target untuk metrik tertentu. Hal ini serupa dengan cara termostat mempertahankan suhu rumah Anda. Anda memilih suhu dan termostat melakukan sisanya.
- Penskalaan terjadwal ElastiCache untuk Aplikasi untuk penskalaan otomatis Redis - Menambah atau mengurangi jumlah shard/replika yang dijalankan layanan Anda berdasarkan tanggal dan waktu.




Langkah-langkah berikut merangkum proses penskalaan otomatis Redis seperti yang ditunjukkan pada diagram sebelumnya: ElastiCache

1. Anda membuat kebijakan ElastiCache penskalaan otomatis Redis untuk Grup Replikasi Redis Anda ElastiCache .
2. ElastiCache untuk Redis auto scaling membuat sepasang CloudWatch alarm atas nama Anda. Setiap pasangan mewakili batas atas dan bawah Anda untuk metrik. CloudWatch Alarm ini dipicu ketika penggunaan aktual cluster menyimpang dari penggunaan target Anda untuk jangka waktu yang berkelanjutan. Anda dapat melihat alarm di konsol.
3. Jika nilai metrik yang dikonfigurasi melebihi pemanfaatan target Anda (atau berada di bawah target) untuk jangka waktu tertentu, CloudWatch memicu alarm yang memanggil penskalaan otomatis ElastiCache Redis untuk mengevaluasi kebijakan penskalaan Anda.
4. ElastiCache untuk masalah penskalaan otomatis Redis, permintaan Ubah untuk menyesuaikan kapasitas cluster Anda.

5. ElastiCache untuk Redis memproses permintaan Modify, secara dinamis meningkatkan (atau mengurangi) kapasitas Shards/Replicas cluster sehingga mendekati pemanfaatan target Anda.

Untuk memahami cara ElastiCache kerja Redis Auto Scaling, misalkan Anda memiliki cluster bernama. `UsersCluster` Dengan memantau CloudWatch metrik`UsersCluster`, Anda menentukan pecahan Max yang dibutuhkan cluster saat lalu lintas berada di puncaknya dan Min Shards saat lalu lintas berada pada titik terendah. Anda juga memutuskan nilai target untuk penggunaan CPU untuk klaster `UsersCluster`. ElastiCache untuk Redis auto scaling menggunakan algoritma pelacakan targetnya untuk memastikan bahwa pecahan yang `UsersCluster` disediakan disesuaikan sesuai kebutuhan sehingga pemanfaatan tetap pada atau mendekati nilai target.

> **Note**
>
> Penskalaan mungkin membutuhkan waktu yang nyata dan akan membutuhkan sumber daya klaster tambahan agar pecahan dapat diseimbangkan kembali. ElastiCache untuk Redis Auto Scaling memodifikasi pengaturan sumber daya hanya jika beban kerja sebenarnya tetap meningkat (atau tertekan) selama beberapa menit. Algoritma pelacakan target penskalaan otomatis Redis berupaya menjaga pemanfaatan target pada atau mendekati nilai yang Anda pilih dalam jangka panjang. ElastiCache

## Kebijakan Auto Scaling

Kebijakan penskalaan memiliki komponen berikut:

- Metrik target – Metrik CloudWatch yang digunakan Auto Scaling ElastiCache for Redis untuk menentukan waktu dan jumlah yang harus diskalakan.
- Kapasitas minimum dan maksimum – Jumlah serpihan atau replika minimum dan maksimum yang digunakan untuk penskalaan.

> **Important**
>
> Saat membuat kebijakan penskalaan otomatis, jika kapasitas saat ini lebih tinggi dari kapasitas maksimum yang dikonfigurasi, kita menskalakan ke dalam (scaleIn) ke MaxCapacity selama pembuatan kebijakan. Demikian pula jika kapasitas saat ini lebih

rendah dari kapasitas minimum yang dikonfigurasi, kita menskalakan ke luar (scaleOut) ke MinCapacity.

- Periode pendinginan – Jumlah waktu, dalam detik, setelah aktivitas penskalaan ke dalam atau penskalaan ke luar selesai sebelum aktivitas penskalaan ke luar lainnya dapat dimulai.
- Peran tertaut layanan – Sebuah peran Identity and Access Management (IAM) AWS yang ditautkan ke layanan AWS tertentu. Peran tertaut layanan mencakup semua izin yang diperlukan layanan untuk memanggil layanan AWS lain atas nama Anda. Auto Scaling ElastiCache for Redis secara otomatis menghasilkan peran ini, `AWSServiceRoleForApplicationAutoScaling_ElastiCacheRG`, untuk Anda.
- Aktifkan atau nonaktifkan aktivitas penskalaan ke dalam - Kemampuan untuk mengaktifkan atau menonaktifkan aktivitas penskalaan ke dalam untuk suatu kebijakan.

Topik



## Metrik target untuk Auto Scaling

Dalam jenis kebijakan ini, metrik standar atau kustom dan nilai target untuk metrik tersebut ditentukan dalam konfigurasi kebijakan penskalaan pelacakan target. Auto Scaling ElastiCache for Redis membuat dan mengelola alarm CloudWatch yang memicu kebijakan penskalaan dan menghitung penyesuaian penskalaan berdasarkan metrik dan nilai target. Kebijakan penskalaan menambahkan atau menghapus serpihan/replika yang diperlukan untuk menjaga metrik pada, atau mendekati, nilai target yang ditentukan. Selain menjaga metrik tetap dekat dengan nilai target, kebijakan penskalaan pelacakan target juga disesuaikan menurut fluktuasi metrik karena pola beban kerja yang berubah. Kebijakan tersebut juga meminimalkan fluktuasi cepat dalam jumlah serpihan/replika yang tersedia untuk klaster Anda.

Misalnya, pertimbangkan kebijakan penskalaan yang menggunakan metrik `ElastiCachePrimaryEngineCPUUtilization` rata-rata yang telah ditentukan sebelumnya. Kebijakan tersebut dapat menjaga pemanfaatan CPU pada, atau mendekati, persentase penggunaan tertentu, seperti 70 persen.

> **Note**
>
> Untuk setiap klaster, Anda hanya dapat membuat satu kebijakan Auto Scaling untuk setiap metrik target.

Kapasitas minimum dan maksimum

Serpihan

Anda dapat menentukan jumlah maksimum serpihan yang dapat diskalakan dengan penskalaan otomatis ElastiCache for Redis. Nilai ini harus kurang dari atau sama dengan 250 dengan minimum 1. Anda juga dapat menentukan jumlah minimum serpihan yang akan dikelola oleh penskalaan otomatis ElastiCache for Redis. Nilai ini harus minimal 1, dan sama dengan atau kurang dari nilai yang ditentukan untuk serpihan maksimum yaitu 250.

Replika

Anda dapat menentukan jumlah maksimum replika yang akan dikelola oleh penskalaan otomatis ElastiCache for Redis. Nilai ini harus kurang dari atau sama dengan 5. Anda juga dapat menentukan jumlah minimum replika yang akan dikelola oleh penskalaan otomatis ElastiCache for Redis. Nilai ini harus minimal 1, dan sama dengan atau kurang dari nilai yang ditentukan untuk replika maksimum yaitu 5.

Untuk menentukan jumlah minimum dan maksimum serpihan/replika yang Anda perlukan untuk lalu lintas biasa, uji konfigurasi Auto Scaling Anda dengan tingkat lalu lintas yang diharapkan untuk model Anda.

> **Note**
>
> Kebijakan penskalaan otomatis ElastiCache for Redis meningkatkan kapasitas klaster hingga mencapai ukuran maksimum yang Anda tetapkan atau hingga kuota layanan berlaku. Untuk meminta penambahan batas, lihat Batas Layanan AWS dan pilih jenis batas Simpul per klaster per jenis instans.

> **Important**
>
> Penskalaan ke dalam terjadi ketika tidak ada lalu lintas. Jika lalu lintas varian menjadi nol, ElastiCache for Redis secara otomatis menskalakan ke jumlah minimum instans yang ditentukan.

### Periode pendinginan

Anda dapat menyesuaikan daya respons kebijakan penskalaan pelacakan target dengan menambahkan periode pendinginan yang memengaruhi penskalaan klaster Anda. Periode pendinginan memblokir permintaan penskalaan ke dalam atau ke luar berikutnya hingga periode ini berakhir. Hal ini memperlambat penghapusan serpihan/replika di klaster ElastiCache for Redis Anda untuk permintaan penskalaan ke dalam, dan pembuatan serpihan/replika untuk permintaan penskalaan ke luar. Anda dapat menentukan periode pendinginan berikut:

- Aktivitas penskalaan ke dalam mengurangi jumlah serpihan/replika di klaster ElastiCache for Redis Anda. Periode pendinginan penskalaan ke dalam menentukan jumlah waktu, dalam detik, setelah aktivitas penskalaan ke dalam selesai sebelum aktivitas penskalaan ke dalam lainnya dapat dimulai.
- Aktivitas penskalaan ke luar meningkatkan jumlah serpihan/replika di klaster ElastiCache for Redis Anda. Periode pendinginan penskalaan ke luar menentukan jumlah waktu, dalam detik, setelah aktivitas penskalaan ke luar selesai sebelum aktivitas penskalaan ke luar lainnya dapat dimulai.

Ketika periode pendinginan penskalaan ke dalam atau penskalaan ke luar tidak ditentukan, nilai default untuk penskalaan ke luar adalah 600 detik dan untuk penskalaan ke dalam adalah 900 detik.

### Mengaktifkan atau menonaktifkan aktivitas penskalaan ke dalam

Anda dapat mengaktifkan atau menonaktifkan aktivitas penskalaan ke dalam untuk sebuah kebijakan. Mengaktifkan aktivitas penskalaan ke dalam memungkinkan kebijakan penskalaan untuk menghapus serpihan/replika. Saat aktivitas penskalaan ke dalam diaktifkan, periode pendinginan penskalaan ke dalam di kebijakan penskalaan berlaku untuk aktivitas penskalaan ke dalam. Menonaktifkan aktivitas penskalaan ke luar mencegah kebijakan penskalaan menghapus serpihan/replika.

> **Note**
>
> Aktivitas penskalaan ke luar selalu diaktifkan agar kebijakan penskalaan dapat membuat serpihan/replika ElastiCache for Redis sesuai kebutuhan.

## Izin IAM Diperlukan ElastiCache untuk Redis Auto Scaling

ElastiCache untuk Redis Auto Scaling dimungkinkan oleh kombinasi untuk Redis, CloudWatch, dan ElastiCache Application Auto Scaling API. Cluster dibuat dan diperbarui dengan ElastiCache Redis, alarm dibuat dengan CloudWatch, dan kebijakan penskalaan dibuat dengan Application Auto Scaling. Selain izin IAM standar untuk membuat dan memperbarui cluster, pengguna IAM yang mengakses pengaturan Redis ElastiCache Auto Scaling harus memiliki izin yang sesuai untuk layanan yang mendukung penskalaan dinamis. Pengguna IAM harus memiliki izin untuk menggunakan tindakan yang ditunjukkan dalam contoh kebijakan berikut:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "application-autoscaling:*",
                "elasticache:DescribeReplicationGroups",
                "elasticache:ModifyReplicationGroupShardConfiguration",
                "elasticache:IncreaseReplicaCount",
                "elasticache:DecreaseReplicaCount",
                "elasticache:DescribeCacheClusters",
                "elasticache:DescribeCacheParameters",
                "cloudwatch:DeleteAlarms",
                "cloudwatch:DescribeAlarmHistory",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:DescribeAlarmsForMetric",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics",
                "cloudwatch:PutMetricAlarm",
                "cloudwatch:DisableAlarmActions",
                "cloudwatch:EnableAlarmActions",
                "iam:CreateServiceLinkedRole",
                "sns:CreateTopic",
                "sns:Subscribe",
```

```
                "sns:Get*",
                "sns:List*"
            ],
            "Resource": "arn:aws:iam::123456789012:role/autoscaling-roles-for-cluster"
        }
    ]
}
```

## Peran terkait layanan

Layanan penskalaan otomatis ElastiCache untuk Redis juga memerlukan izin untuk mendeskripsikan cluster dan CloudWatch alarm Anda, dan izin untuk memodifikasi kapasitas target Redis Anda ElastiCache atas nama Anda. Jika Anda mengaktifkan Auto Scaling untuk klaster Redis Anda ElastiCache , itu akan membuat peran terkait layanan bernama. `AWSServiceRoleForApplicationAutoScaling_ElastiCacheRG` Peran terkait layanan ini memberikan izin penskalaan otomatis ElastiCache Redis untuk menjelaskan alarm kebijakan Anda, untuk memantau kapasitas armada saat ini, dan untuk memodifikasi kapasitas armada. Peran terkait layanan adalah peran default ElastiCache untuk penskalaan otomatis Redis. Untuk informasi selengkapnya, lihat Peran terkait layanan ElastiCache untuk penskalaan otomatis Redis di Panduan Pengguna Application Auto Scaling.

## Praktik Terbaik Auto Scaling

Sebelum mendaftar ke Auto Scaling, kami merekomendasikan hal berikut:

1. Gunakan hanya satu metrik pelacakan – Identifikasi apakah klaster Anda memiliki beban kerja intensif CPU atau data dan gunakan metrik standar yang sesuai untuk menentukan Kebijakan Penskalaan.
   - CPU mesin: `ElastiCachePrimaryEngineCPUUtilization` (dimensi serpihan) atau `ElastiCacheReplicaEngineCPUUtilization` (dimensi replika)
   - Penggunaan basis data: `ElastiCacheDatabaseCapacityUsageCountedForEvictPercentage` Kebijakan penskalaan ini bekerja paling baik dengan maxmemory-policy yang ditetapkan ke noeviction di klaster.

   Kami menyarankan Anda menghindari beberapa kebijakan per dimensi pada klaster. ElastiCache untuk Redis Auto scaling akan menskalakan target yang dapat diskalakan jika ada kebijakan pelacakan target yang siap untuk diskalakan, tetapi akan menskalakan hanya jika semua kebijakan pelacakan target (dengan bagian penskalaan diaktifkan) siap untuk diskalakan. Jika

beberapa kebijakan menginstruksikan target yang dapat diskalakan untuk menskalakan ke luar atau ke dalam pada saat yang sama, target akan diskalakan berdasarkan kebijakan yang memberikan kapasitas terbesar untuk penskalaan ke dalam dan penskalaan ke luar.

2. Metrik yang Dikustomisasi untuk Pelacakan Target – Berhati-hatilah saat menggunakan metrik yang dikustomisasi untuk Pelacakan Target karena Penskalaan otomatis paling sesuai untuk menskalakan ke luar/ke dalam sebanding dengan perubahan metrik yang dipilih untuk kebijakan. Jika metrik tersebut tidak berubah secara proporsional dengan tindakan penskalaan yang digunakan untuk pembuatan kebijakan, hal tersebut dapat menyebabkan tindakan penskalaan ke luar atau penskalaan ke dalam berkelanjutan yang dapat memengaruhi ketersediaan atau biaya.

   Untuk klaster tingkat data (tipe instans keluarga r6gd), hindari penggunaan metrik berbasis memori untuk penskalaan.

3. Penskalaan Terjadwal – Jika Anda mengidentifikasi bahwa beban kerja Anda bersifat deterministik (mencapai tinggi/rendah pada waktu tertentu), sebaiknya gunakan Penskalaan Terjadwal dan konfigurasikan kapasitas target Anda sesuai dengan kebutuhan. Pelacakan Target paling cocok untuk beban kerja non-deterministik dan agar klaster beroperasi pada metrik target yang diperlukan dengan penskalaan ke luar saat Anda membutuhkan lebih banyak sumber daya dan penskalaan ke dalam saat Anda membutuhkan lebih sedikit sumber daya.
4. Nonaktifkan Penskalaan ke Dalam – Penskalaan otomatis pada Pelacakan Target paling sesuai untuk klaster dengan peningkatan/penurunan beban kerja secara bertahap karena lonjakan/penurunan metrik dapat memicu osilasi penskalaan ke luar/ke dalam berturut-turut. Untuk menghindari osilasi tersebut, Anda dapat memulai dengan menonaktifkan penskalaan ke luar, dan Anda dapat melakukan penskalaan ke dalam secara manual sesuai kebutuhan Anda kapan saja.
5. Uji aplikasi Anda – Kami merekomendasikan Anda menguji aplikasi Anda dengan perkiraan beban kerja min & maks untuk menentukan serpihan/replika Min,Maks mutlak yang diperlukan untuk klaster sambil membuat kebijakan Penskalaan untuk menghindari masalah ketersediaan. Penskalaan otomatis dapat menskalakan ke luar ke Maks dan menskalakan ke dalam ke ambang Min yang dikonfigurasi untuk target.
6. Mendefinisikan Nilai Target — Anda dapat menganalisis CloudWatch metrik yang sesuai untuk pemanfaatan klaster selama periode empat minggu untuk menentukan ambang nilai target. Jika Anda masih tidak yakin nilai apa yang harus dipilih, sebaiknya mulai dengan nilai metrik standar minimum yang didukung.
7. AutoScaling pada Pelacakan Target paling cocok untuk cluster dengan distribusi beban kerja yang seragam di seluruh dimensi pecahan/replika. Memiliki distribusi yang tidak seragam dapat menyebabkan:

- Penskalaan saat tidak diperlukan karena lonjakan/penurunan beban kerja pada beberapa serpihan/replika panas.
- Tidak menskalakan saat diperlukan karena rata-rata keseluruhan mendekati target meskipun memiliki serpihan/replika panas.

> **Note**
>
> Saat menskalakan cluster Anda, secara otomatis ElastiCache akan mereplikasi Fungsi yang dimuat di salah satu node yang ada (dipilih secara acak) ke node baru. Jika klaster Anda memiliki Redis 7.0 atau yang lebih baru dan aplikasi Anda menggunakan Fungsi Redis, sebaiknya muat semua fungsi Anda ke semua serpihan sebelum melakukan penskalaan ke luar agar klaster Anda tidak berakhir dengan fungsi yang berbeda pada serpihan yang berbeda.

Setelah mendaftar AutoScaling, perhatikan hal berikut:

- Ada batasan pada penskalaan Otomatis Konfigurasi yang Didukung. Jadi, kami menyarankan Anda untuk tidak mengubah konfigurasi grup replikasi yang terdaftar untuk penskalaan Otomatis. Berikut ini adalah beberapa contohnya:
  - Memodifikasi tipe Instans secara manual ke tipe yang tidak didukung.
  - Mengaitkan grup replikasi ke penyimpanan data Global.
  - Mengubah parameter `ReservedMemoryPercent`.
  - Meningkatkan/mengurangi serpihan/replika secara manual di luar kapasitas Min/Maks yang dikonfigurasi selama pembuatan kebijakan.

## Menggunakan Penskalaan Otomatis dengan serpihan

Bagian berikut memberikan rincian tentang pelacakan target dan kebijakan terjadwal serta cara menerapkannya menggunakan AWS Management Console AWS CLI dan API.

Kebijakan penskalaan pelacakan target

Dengan kebijakan penskalaan pelacakan target, Anda memilih metrik dan menetapkan nilai target. Penskalaan Otomatis ElastiCache for Redis membuat dan mengelola alarm CloudWatch yang memicu kebijakan penskalaan dan menghitung penyesuaian penskalaan berdasarkan metrik dan

nilai target. Kebijakan penskalaan menambahkan atau menghapus serpihan yang diperlukan untuk menjaga metrik pada, atau mendekati, nilai target yang ditentukan. Selain menjaga metrik agar mendekati nilai target, kebijakan penskalaan pelacakan target juga menyesuaikan dengan fluktuasi metrik karena fluktuasi pola muatan dan meminimalkan fluktuasi cepat dalam kapasitas armada.

Misalnya, pertimbangkan kebijakan penskalaan yang menggunakan metrik `ElastiCachePrimaryEngineCPUUtilization` rata-rata yang telah ditetapkan dengan nilai target yang dikonfigurasi. Kebijakan tersebut dapat menjaga pemanfaatan CPU pada, atau mendekati, nilai target yang ditentukan.

Metrik yang telah ditetapkan

Metrik yang telah ditetapkan adalah struktur yang mengacu pada nama, dimensi, dan statistik tertentu (`average`) dari metrik CloudWatch tertentu. Kebijakan Penskalaan Otomatis menentukan metrik yang telah ditetapkan berikut untuk klaster Anda:

| Nama Metrik yang Telah Ditetapkan | Nama Metrik CloudWatch | Dimensi Metrik CloudWatch | Tipe Instans yang Tidak Memenuhi Syarat |
|---|---|---|---|
| `ElastiCachePrimaryEngineCPUUtilization` | `EngineCPUUtilization` | ReplicationGroupId, Peran = Primer | Tidak ada |
| `ElastiCacheDatabaseCapacityUsageCountedForEvictPercentage` | `DatabaseCapacityUsageCountedForEvictPercentage` | Metrik Grup Replikasi Redis | Tidak ada |
| `ElastiCacheDatabaseMemoryUsageCounte` | `DatabaseMemoryUsageCountedForEvictPercentage` | Metrik Grup Replikasi Redis | R6gd |

| Nama Metrik yang Telah Ditetapkan | Nama Metrik CloudWatch | Dimensi Metrik CloudWatch | Tipe Instans yang Tidak Memenuhi Syarat |
|---|---|---|---|
| `dForEvict Percentag e` | | | |

Tipe instans berjenjang menurut data tidak dapat menggunakan `ElastiCacheDatabaseMemoryUsageCountedForEvictPercentage`, karena tipe instans ini menyimpan data di memori dan SSD. Kasus penggunaan yang diharapkan untuk instans berjenjang menurut data adalah memiliki 100 persen penggunaan memori dan mengisi SSD sesuai kebutuhan.

Kriteria Penskalaan Otomatis untuk serpihan

Ketika layanan mendeteksi bahwa metrik standar Anda sama dengan atau lebih besar dari pengaturan Target, kapasitas serpihan akan ditingkatkan secara otomatis. ElastiCache for Redis menskalakan ke luar serpihan klaster dengan memilih angka berdasarkan nilai yang lebih besar di antara dua nilai berikut: Variasi persen dari Target dan 20 persen serpihan saat ini. Untuk menskalakan ke dalam, ElastiCache for Redis tidak akan melakukan penskalaan otomatis kecuali jika nilai metrik keseluruhan di bawah 75 persen dari Target yang Anda tentukan.

Untuk contoh penskalaan ke luar, jika Anda memiliki 50 serpihan dan

- jika Target Anda dilanggar sebesar 30 persen, ElastiCache for Redis akan menskalakan ke luar sebesar 30 persen, yang menghasilkan 65 serpihan per klaster.
- jika Target Anda dilanggar sebesar 10 persen, ElastiCache for Redis akan menskalakan ke luar secara default Minimum 20 persen, yang menghasilkan 60 serpihan per klaster.

Sebagai contoh penskalaan ke dalam, jika Anda telah memilih nilai Target 60 persen, ElastiCache for Redis tidak akan melakukan penskalaan otomatis sampai metrik kurang dari atau sama dengan 45 persen (25 persen di bawah Target 60 persen).

Pertimbangan untuk Penskalaan Otomatis

Perhatikan sejumlah pertimbangan berikut:

- Kebijakan penskalaan pelacakan target mengasumsikan bahwa penskalaan ke luar harus dilakukan saat metrik yang ditentukan berada di atas nilai target. Anda tidak dapat menggunakan

kebijakan penskalaan pelacakan target untuk menskalakan ke luar jika metrik yang ditentukan berada di bawah nilai target. ElastiCache for Redis menskalakan serpihan ke luar dengan minimal 20 persen deviasi terhadap target serpihan yang ada di klaster.

- Kebijakan penskalaan pelacakan target tidak melakukan penskalaan saat metrik yang telah ditetapkan tidak memiliki data yang mencukupi. Kebijakan penskalaan pelacakan target tidak melakukan penskalaan kedalam karena data yang tidak mencukupi tidak ditafsirkan sebagai pemanfaatan yang rendah.
- Anda mungkin melihat kesenjangan antara nilai target dan titik data metrik aktual. Hal ini karena Penskalaan Otomatis ElastiCache for Redis selalu bertindak konservatif dengan membulatkan ke atas atau ke bawah saat menentukan jumlah kapasitas yang dapat ditambahkan atau dihapus. Hal ini mencegah layanan menambahkan kapasitas yang tidak mencukupi atau menghapus kapasitas terlalu banyak.
- Untuk memastikan ketersediaan aplikasi, layanan menskalakan ke luar secara proporsional berdasarkan metrik secepat mungkin, tetapi menskalakan ke dalam secara lebih konservatif.
- Anda dapat memiliki beberapa kebijakan penskalaan pelacakan target untuk klaster ElastiCache for Redis, asalkan masing-masing kebijakan tersebut menggunakan metrik yang berbeda. Tujuan Penskalaan Otomatis ElastiCache for Redis adalah untuk selalu memprioritaskan ketersediaan, sehingga perilakunya akan berbeda-beda, tergantung apakah kebijakan pelacakan target siap untuk menskalakan ke luar atau menskalakan ke dalam. Fitur ini akan menskalakan ke luar layanan jika salah satu kebijakan pelacakan target siap untuk diskalakan ke luar. Namun, penskalaan ke dalam akan dilakukan hanya jika semua kebijakan pelacakan target (dengan porsi penskalaan ke dalam diaktifkan) siap untuk diskalakan ke dalam.
- Jangan mengedit atau menghapus alarm CloudWatch yang dikelola Penskalaan Otomatis ElastiCache for Redis untuk kebijakan penskalaan pelacakan target. Penskalaan Otomatis ElastiCache for Redis menghapus alarm secara otomatis saat Anda menghapus kebijakan penskalaan.
- Dengan Penskalaan Otomatis ElastiCache for Redis, Anda masih dapat mengubah serpihan klaster secara manual. Penyesuaian manual ini tidak memengaruhi alarm CloudWatch saat ini yang dilampirkan ke kebijakan penskalaan, tetapi dapat memengaruhi metrik yang dapat memicu alarm CloudWatch ini.
- Alarm CloudWatch yang dikelola oleh Penskalaan Otomatis ini ditentukan melalui metrik AVG di semua serpihan di klaster. Jadi, serpihan dengan lalu lintas tertinggi dapat menghasilkan salah satu skenario berikut:
  - melakukan penskalaan saat tidak diperlukan karena pemuatan pada beberapa serpihan berlalu lintas tinggi yang memicu alarm CloudWatch

  - tidak melakukan penskalaan saat diperlukan karena AVG agregat di semua serpihan yang memengaruhi alarm agar tidak melanggar ketentuan yang ditetapkan.
- Batas default ElastiCache for Redis pada Simpul per klaster masih berlaku. Jadi, saat memilih Penskalaan Otomatis dan jika Anda mengharapkan simpul maksimum lebih dari batas default, minta peningkatan batas di Batas Layanan AWS dan pilih tipe batas Simpul per klaster per tipe instans.
- Pastikan Anda memiliki cukup ENI (Antarmuka Jaringan Elastis) yang tersedia di VPC Anda, yang diperlukan selama penskalaan ke luar. Untuk informasi selengkapnya, lihat Antarmuka Jaringan Elastis.
- Jika kapasitas yang tersedia dari EC2 tidak memadai, Penskalaan Otomatis ElastiCache for Redis tidak akan melakukan penskalaan dan menundanya hingga kapasitas tersedia.
- Penskalaan Otomatis ElastiCache for Redis selama penskalaan ke dalam tidak akan menghapus serpihan dengan slot yang memiliki ukuran item lebih besar dari 256 MB pasca-serialisasi.
- Selama penskalaan ke dalam, serpihan tidak akan dihapus jika memori yang tersedia tidak mencukupi pada konfigurasi serpihan yang dihasilkan.

### Menambahkan kebijakan penskalaan

Anda dapat menambahkan kebijakan penskalaan menggunakan AWS Management Console.

Untuk menambahkan kebijakan Auto Scaling ke klaster ElastiCache for Redis

1. Masuk ke AWS Management Console dan buka konsol Amazon ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Di panel navigasi, pilih Redis.
3. Pilih klaster yang ingin Anda tambahi kebijakan (pilih nama klaster, bukan tombol di sebelah kirinya).
4. Pilih tab Kebijakan Auto Scaling.
5. Pilih tambahkan penskalaan dinamis.
6. Untuk nama Kebijakan, masukkan nama kebijakan.
7. Untuk Dimensi yang Dapat Diskalakan pilih serpihan.
8. Untuk metrik target, pilih salah satu dari berikut:

   - Penggunaan CPU Primer untuk membuat kebijakan berdasarkan penggunaan CPU rata-rata.
   - Memori untuk membuat kebijakan berdasarkan memori database rata-rata.

- Kapasitas untuk membuat kebijakan berdasarkan penggunaan kapasitas basis data rata-rata. Metrik Kapasitas mencakup penggunaan memori dan SSD untuk instans berjenjang data, dan pemanfaatan memori untuk semua jenis instans lainnya.

9. Untuk nilai target, pilih nilai yang lebih besar dari atau sama dengan 35 dan kurang dari atau sama dengan 70. Penskalaan otomatis akan mempertahankan nilai ini untuk metrik target yang dipilih di seluruh serpihan ElastiCache Anda:

   - Pemanfaatan CPU Primer: mempertahankan nilai target untuk metrik `EngineCPUUtilization` di simpul primer.
   - Memori: mempertahankan nilai target untuk metrik `DatabaseMemoryUsageCountedForEvictPercentage`
   - Kapasitas mempertahankan nilai target untuk metrik `DatabaseCapacityUsageCountedForEvictPercentage`,

   Serpihan klaster ditambahkan atau dihapus untuk menjaga metrik tetap mendekati nilai yang ditentukan.

10. (Opsional) Periode pendinginan penskalaan ke dalam atau penskalaan ke luar tidak didukung dari konsol. Gunakan AWS CLI untuk memodifikasi nilai pendinginan.
11. Untuk Kapasitas minimum, ketikkan jumlah minimum serpihan yang diperlukan oleh kebijakan ElastiCache for Redis Auto Scaling untuk dipertahankan.
12. Untuk Kapasitas maksimum, ketikkan jumlah maksimum serpihan yang diperlukan oleh kebijakan ElastiCache for Redis Auto Scaling untuk dipertahankan. Nilai ini harus kurang dari atau sama dengan 250.
13. Pilih Buat.

### Mendaftarkan Target yang Dapat Diskalakan

Sebelum Anda dapat menggunakan Penskalaan Otomatis dengan klaster ElastiCache for Redis, Anda mendaftarkan klaster dengan penskalaan otomatis ElastiCache for Redis. Anda melakukannya untuk menentukan dimensi dan batasan penskalaan yang akan diterapkan pada klaster tersebut. Penskalaan otomatis ElastiCache for Redis menskalakan klaster ElastiCache for Redis secara dinamis berdasarkan dimensi `elasticache:replication-group:NodeGroups` yang dapat diskalakan, yang merepresentasikan jumlah serpihan klaster.

#### Menggunakan AWS CLI

Untuk mendaftarkan klaster ElastiCache for Redis Anda, gunakan perintah register-scalable-target dengan parameter berikut:

- `--service-namespace` – Atur nilai ini ke `elasticache`.
- `--resource-id`— Pengidentifikasi sumber daya untuk klaster ElastiCache for Redis. Untuk parameter ini, jenis sumber daya adalah `ReplicationGroup` dan pengidentifikasi unik adalah nama klaster ElastiCache for Redis, misalnya `replication-group/myscalablecluster`.
- `--scalable-dimension` – Atur nilai ini ke `elasticache:replication-group:NodeGroups`.
- `--max-capacity` – Jumlah maksimum serpihan yang akan dikelola oleh penskalaan otomatis ElastiCache for Redis. Untuk informasi tentang hubungan antara `--min-capacity`, `--max-capacity`, dan jumlah serpihan dalam klaster Anda, lihat Kapasitas minimum dan maksimum.
- `--min-capacity` – Jumlah minimum serpihan yang akan dikelola oleh penskalaan otomatis ElastiCache for Redis. Untuk informasi tentang hubungan antara `--min-capacity`, `--max-capacity`, dan jumlah serpihan dalam klaster Anda, lihat Kapasitas minimum dan maksimum.

Example

Dalah contoh berikut, Anda mendaftarkan klaster ElastiCache for Redis bernama `myscalablecluster`. Pendaftaran ini menunjukkan bahwa klaster harus diskalakan secara dinamis agar memiliki satu hingga sepuluh serpihan.

Untuk Linux, macOS, atau Unix:

```
aws application-autoscaling register-scalable-target \
    --service-namespace elasticache \
    --resource-id replication-group/myscalablecluster \
    --scalable-dimension elasticache:replication-group:NodeGroups \
    --min-capacity 1 \
    --max-capacity 10 \
```

Untuk Windows:

```
aws application-autoscaling register-scalable-target ^
    --service-namespace elasticache ^
    --resource-id replication-group/myscalablecluster ^
    --scalable-dimension elasticache:replication-group:NodeGroups ^
    --min-capacity 1 ^
```

```
    --max-capacity 10 ^
```

Menggunakan API

Untuk mendaftarkan klaster ElastiCache Anda, gunakan perintah register-scalable-target dengan parameter berikut:

- ServiceNamespace — Atur nilai ini ke elasticache.
- ResourceID — Pengidentifikasi sumber daya untuk klaster ElastiCache. Untuk parameter ini, jenis sumber daya adalah ReplicationGroup dan pengidentifikasi unik adalah nama klaster ElastiCache for Redis, misalnya `replication-group/myscalablecluster`.
- ScalableDimension - Atur nilai ini ke `elasticache:replication-group:NodeGroups`.
- MinCapacity – Jumlah minimum serpihan yang akan dikelola oleh penskalaan otomatis ElastiCache for Redis. Untuk informasi tentang hubungan antara --min-capacity, --max-capacity, dan jumlah replika dalam klaster Anda, lihat Kapasitas minimum dan maksimum.
- MaxCapacity – Jumlah maksimum serpihan yang akan dikelola oleh penskalaan otomatis ElastiCache for Redis. Untuk informasi tentang hubungan antara --min-capacity, --max-capacity, dan jumlah replika dalam klaster Anda, lihat Kapasitas minimum dan maksimum.

Example

Dalah contoh berikut, Anda mendaftarkan klaster ElastiCache for Redis bernama `myscalablecluster` dengan API Penskalaan Otomatis Aplikasi. Pendaftaran ini menunjukkan bahwa klaster harus diskalakan secara dinamis agar memiliki satu hingga 5 replika.

```
POST / HTTP/1.1
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.RegisterScalableTarget
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS
{
    "ServiceNamespace": "elasticache",
    "ResourceId": "replication-group/myscalablecluster",
    "ScalableDimension": "elasticache:replication-group:NodeGroups",
    "MinCapacity": 1,
```

```
    "MaxCapacity": 5
}
```

Menetapkan kebijakan penskalaan

Konfigurasi kebijakan penskalaan pelacakan target direpresentasikan oleh blok JSON yang digunakan untuk mendefinisikan metrik dan nilai target. Anda dapat menyimpan konfigurasi kebijakan penskalaan sebagai blok JSON dalam file teks. Anda menggunakan file teks tersebut saat menjalankan AWS CLI atau Application Auto Scaling API. Untuk informasi selengkapnya tentang sintaks konfigurasi kebijakan, lihat TargetTrackingScalingPolicyConfigurationdi Referensi API Application Auto Scaling.

Opsi berikut tersedia untuk menetapkan konfigurasi kebijakan penskalaan pelacakan target:

Topik

- Menggunakan metrik yang sudah ditentukan sebelumnya
- Menggunakan metrik khusus
- Menggunakan periode pendinginan
- Menonaktifkan aktivitas penskalaan ke dalam
- Menerapkan kebijakan penskalaan

Menggunakan metrik yang sudah ditentukan sebelumnya

Dengan menggunakan metrik yang telah ditentukan sebelumnya, Anda dapat dengan cepat menentukan kebijakan penskalaan pelacakan target untuk klaster Redis yang berfungsi dengan pelacakan target ElastiCache untuk Redis Auto Scaling. ElastiCache

Saat ini, ElastiCache untuk Redis mendukung metrik standar berikut ElastiCache untuk Redis Auto NodeGroup Scaling:

- ElastiCachePrimaryEngineCPUUutilization — Nilai rata-rata `EngineCPUUtilization` metrik di CloudWatch seluruh node primer di cluster ElastiCache for Redis.
- ElastiCacheDatabaseMemoryUsageCountedForEvictPercentage— Nilai rata-rata `DatabaseMemoryUsageCountedForEvictPercentage` metrik di CloudWatch seluruh node primer di cluster ElastiCache for Redis.
- ElastiCacheDatabaseCapacityUsageCountedForEvictPercentage— Nilai rata-rata `ElastiCacheDatabaseCapacityUsageCountedForEvictPercentage` metrik di CloudWatch seluruh node primer di cluster ElastiCache for Redis.

Untuk informasi selengkapnya tentang metrik `EngineCPUUtilization`, `DatabaseMemoryUsageCountedForEvictPercentage` dan `DatabaseCapacityUsageCountedForEvictPercentage`, lihat Memantau penggunaan dengan Metrik CloudWatch. Untuk menggunakan metrik yang sudah ditentukan sebelumnya dalam kebijakan penskalaan, Anda membuat konfigurasi pelacakan target untuk kebijakan penskalaan Anda. Konfigurasi ini harus menyertakan a `PredefinedMetricSpecification` untuk metrik yang telah ditentukan dan a TargetValue untuk nilai target metrik tersebut.

Example

Contoh berikut menjelaskan konfigurasi kebijakan tipikal untuk penskalaan pelacakan target untuk klaster Redis ElastiCache . Dalam konfigurasi ini, metrik yang `ElastiCachePrimaryEngineCPUUtilization` telah ditentukan digunakan untuk menyesuaikan cluster ElastiCache for Redis berdasarkan pemanfaatan CPU rata-rata 40 persen di semua node primer di cluster.

```
{
    "TargetValue": 40.0,
    "PredefinedMetricSpecification":
    {
        "PredefinedMetricType": "ElastiCachePrimaryEngineCPUUtilization"
    }
}
```

Menggunakan metrik khusus

Dengan menggunakan metrik kustom, Anda dapat menentukan kebijakan penskalaan pelacakan target yang memenuhi persyaratan kustom Anda. Anda dapat menentukan metrik kustom berdasarkan ElastiCache metrik apa pun yang berubah sebanding dengan penskalaan. Tidak semua ElastiCache metrik berfungsi untuk pelacakan target. Metrik harus berupa metrik penggunaan yang valid dan menjelaskan seberapa sibuk suatu instans. Nilai metrik harus meningkat atau menurun proporsinya terhadap jumlah Serpihan dalam klaster. Peningkatan atau penurunan proporsional ini diperlukan untuk menggunakan data metrik untuk menskalakan jumlah serpihan ke luar atau masuk secara proporsional.

Example

Contoh berikut menjelaskan konfigurasi pelacakan target untuk kebijakan penskalaan. Dalam konfigurasi ini, metrik khusus menyesuaikan cluster ElastiCache for Redis berdasarkan pemanfaatan CPU rata-rata 50 persen di semua pecahan dalam cluster bernama. `my-db-cluster`

```
{
    "TargetValue": 50,
    "CustomizedMetricSpecification":
    {
        "MetricName": "EngineCPUUtilization",
        "Namespace": "AWS/ElastiCache",
        "Dimensions": [
            {
                "Name": "RelicationGroup","Value": "my-db-cluster"
            },
            {
                "Name": "Role","Value": "PRIMARY"
            }
        ],
        "Statistic": "Average",
        "Unit": "Percent"
    }
}
```

Menggunakan periode pendinginan

Anda dapat menentukan nilai, dalam detik, untuk `ScaleOutCooldown` guna menambahkan periode pendinginan untuk menskalakan klaster Anda ke luar. Demikian pula, Anda dapat menentukan nilai, dalam detik, untuk `ScaleInCooldown` guna menambahkan periode pendinginan untuk menskalakan klaster Anda ke dalam. Untuk informasi selengkapnya, lihat TargetTrackingScalingPolicyConfigurationdi Referensi API Application Auto Scaling.

Contoh berikut menjelaskan konfigurasi pelacakan target untuk kebijakan penskalaan. Dalam konfigurasi ini, metrik yang `ElastiCachePrimaryEngineCPUUtilization` telah ditentukan digunakan untuk menyesuaikan cluster ElastiCache for Redis berdasarkan pemanfaatan CPU rata-rata 40 persen di semua node utama di cluster itu. Konfigurasi ini menyediakan periode pendinginan penskalaan ke dalam selama 10 menit dan periode pendinginan penskalaan ke luar selama 5 menit.

```
{
    "TargetValue": 40.0,
    "PredefinedMetricSpecification":
    {
        "PredefinedMetricType": "ElastiCachePrimaryEngineCPUUtilization"
    },
    "ScaleInCooldown": 600,
    "ScaleOutCooldown": 300
```

```
}
```

Menonaktifkan aktivitas penskalaan ke dalam

Anda dapat mencegah konfigurasi kebijakan penskalaan pelacakan target dari penskalaan di klaster Redis Anda dengan menonaktifkan ElastiCache aktivitas penskalaan. Menonaktifkan aktivitas penskalaan ke dalam mencegah kebijakan penskalaan menghapus serpihan, namun masih memungkinkan kebijakan penskalaan untuk membuatnya sesuai kebutuhan.

Anda dapat menentukan nilai Boolean untuk `DisableScaleIn` guna mengaktifkan atau menonaktifkan aktivitas penskalaan ke dalam untuk klaster Anda. Untuk informasi selengkapnya, lihat TargetTrackingScalingPolicyConfigurationdi Referensi API Application Auto Scaling.

Contoh berikut menjelaskan konfigurasi pelacakan target untuk kebijakan penskalaan. Dalam konfigurasi ini, metrik yang `ElastiCachePrimaryEngineCPUUtilization` telah ditentukan menyesuaikan cluster ElastiCache for Redis berdasarkan pemanfaatan CPU rata-rata 40 persen di semua node utama di cluster itu. Konfigurasi ini menonaktifkan aktivitas penskalaan ke dalam untuk kebijakan penskalaan.

```
{
    "TargetValue": 40.0,
    "PredefinedMetricSpecification":
    {
        "PredefinedMetricType": "ElastiCachePrimaryEngineCPUUtilization"
    },
    "DisableScaleIn": true
}
```

Menerapkan kebijakan penskalaan

Setelah mendaftarkan klaster Anda dengan ElastiCache penskalaan otomatis Redis dan menentukan kebijakan penskalaan, Anda menerapkan kebijakan penskalaan ke klaster terdaftar. Untuk menerapkan kebijakan penskalaan ke klaster Redis, Anda dapat menggunakan Application Auto Scaling API AWS CLI atau Application Auto Scaling. ElastiCache

Menerapkan kebijakan penskalaan menggunakan AWS CLI

Untuk menerapkan kebijakan penskalaan ke klaster Redis Anda ElastiCache , gunakan put-scaling-policyperintah dengan parameter berikut:

- --policy-name – Nama kebijakan penskalaan.

- --policy-name – Atur nilai ini ke `TargetTrackingScaling`.
- --resource-id — Pengidentifikasi sumber daya untuk Redis. ElastiCache Untuk parameter ini, tipe sumber daya adalah `ReplicationGroup` dan pengidentifikasi unik adalah nama ElastiCache untuk cluster Redis, misalnya. `replication-group/myscalablecluster`
- --service-namespace – Atur nilai ini ke `elasticache`.
- --scalable-dimension – Atur nilai ini ke `elasticache:replication-group:NodeGroups`.
- -- target-tracking-scaling-policy -konfigurasi - Konfigurasi kebijakan penskalaan pelacakan target yang akan digunakan untuk klaster for Redis. ElastiCache

Dalam contoh berikut, Anda menerapkan kebijakan penskalaan pelacakan target yang diberi nama `myscalablepolicy` ke kluster Redis yang diberi nama ElastiCache untuk `myscalablecluster` penskalaan otomatis ElastiCache Redis. Untuk melakukannya, Anda menggunakan konfigurasi kebijakan yang disimpan dalam file bernama `config.json`.

Untuk Linux, macOS, atau Unix:

```
aws application-autoscaling put-scaling-policy \
    --policy-name myscalablepolicy \
    --policy-type TargetTrackingScaling \
    --resource-id replication-group/myscalablecluster \
    --service-namespace elasticache \
    --scalable-dimension elasticache:replication-group:NodeGroups \
    --target-tracking-scaling-policy-configuration file://config.json
```

Untuk Windows:

```
aws application-autoscaling put-scaling-policy ^
    --policy-name myscalablepolicy ^
    --policy-type TargetTrackingScaling ^
    --resource-id replication-group/myscalablecluster ^
    --service-namespace elasticache ^
    --scalable-dimension elasticache:replication-group:NodeGroups ^
    --target-tracking-scaling-policy-configuration file://config.json
```

### Menerapkan kebijakan penskalaan menggunakan API

Untuk menerapkan kebijakan penskalaan ke klaster Redis Anda ElastiCache , gunakan PutScalingPolicy AWS CLI perintah dengan parameter berikut:

- --policy-name – Nama kebijakan penskalaan.
- --resource-id — Pengidentifikasi sumber daya untuk Redis. ElastiCache Untuk parameter ini, tipe sumber daya adalah `ReplicationGroup` dan pengidentifikasi unik adalah nama ElastiCache untuk cluster Redis, misalnya. `replication-group/myscalablecluster`
- --service-namespace – Atur nilai ini ke `elasticache`.
- --scalable-dimension – Atur nilai ini ke `elasticache:replication-group:NodeGroups`.
- -- target-tracking-scaling-policy -konfigurasi - Konfigurasi kebijakan penskalaan pelacakan target yang akan digunakan untuk klaster for Redis. ElastiCache

Dalam contoh berikut, Anda menerapkan kebijakan penskalaan pelacakan target yang diberi nama `myscalablepolicy` ke kluster Redis yang diberi nama ElastiCache untuk `myscalablecluster` penskalaan otomatis ElastiCache Redis. Anda menggunakan konfigurasi kebijakan berdasarkan pada metrik `ElastiCachePrimaryEngineCPUUtilization` yang telah ditentukan sebelumnya.

```
POST / HTTP/1.1
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.PutScalingPolicy
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS
{
    "PolicyName": "myscalablepolicy",
    "ServiceNamespace": "elasticache",
    "ResourceId": "replication-group/myscalablecluster",
    "ScalableDimension": "elasticache:replication-group:NodeGroups",
    "PolicyType": "TargetTrackingScaling",
    "TargetTrackingScalingPolicyConfiguration": {
        "TargetValue": 40.0,
        "PredefinedMetricSpecification":
        {
            "PredefinedMetricType": "ElastiCachePrimaryEngineCPUUtilization"
```

```
        }
    }
}
```

### Mengedit kebijakan penskalaan

Anda dapat mengedit kebijakan penskalaan menggunakan AWS Management Console, AWS CLI, atau API Application Auto Scaling.

#### Mengedit kebijakan penskalaan menggunakan AWS Management Console

Untuk mengedit kebijakan Auto Scaling untuk klaster ElastiCache for Redis

1. Masuk ke AWS Management Console dan buka konsol Amazon ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Di panel navigasi, pilih Redis.
3. Pilih klaster yang ingin Anda tambahi kebijakan (pilih nama klaster, bukan tombol di sebelah kirinya).
4. Pilih tab Kebijakan Auto Scaling.
5. Di bagian Kebijakan penskalaan, pilih tombol di samping kebijakan Auto Scaling yang ingin Anda ubah, lalu pilih Ubah.
6. Buat perubahan yang diperlukan pada kebijakan.
7. Pilih Ubah.

#### Mengedit kebijakan penskalaan menggunakan AWS CLI dan API

Anda dapat menggunakan AWS CLI atau API Application Auto Scaling untuk mengedit kebijakan penskalaan dengan cara yang sama seperti Anda menerapkan kebijakan penskalaan:

- Saat menggunakan AWS CLI, tentukan nama kebijakan yang ingin Anda edit dalam parameter `--policy-name`. Tentukan nilai baru untuk parameter yang ingin Anda ubah.
- Saat menggunakan API Application Auto Scaling, tentukan nama kebijakan yang ingin Anda edit dalam parameter `PolicyName`. Tentukan nilai baru untuk parameter yang ingin Anda ubah.

Untuk informasi selengkapnya, lihat Menerapkan kebijakan penskalaan.

### Menghapus kebijakan penskalaan

Anda dapat menghapus kebijakan penskalaan menggunakan AWS Management Console, AWS CLI, atau API Application Auto Scaling.

#### Menghapus kebijakan penskalaan menggunakan AWS Management Console

Untuk menghapus kebijakan Auto Scaling untuk klaster ElastiCache for Redis

1. Masuk ke AWS Management Console dan buka konsol Amazon ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Di panel navigasi, pilih Redis.
3. Pilih klaster yang kebijakan Auto Scaling yang ingin Anda edit (pilih nama klaster, bukan tombol di sebelah kiri).
4. Pilih tab Kebijakan Auto Scaling.
5. Di bagian Kebijakan penskalaan, pilih kebijakan Auto Scaling, lalu pilih Hapus.

#### Menghapus kebijakan penskalaan menggunakan AWS CLI

Untuk menghapus kebijakan penskalaan pada klaster ElastiCache for Redis Anda, gunakan perintah AWS CLI delete-scaling-policy dengan parameter berikut:

- --policy-name – Nama kebijakan penskalaan.
- --resource-id – Pengidentifikasi sumber daya untuk ElastiCache for Redis. Untuk parameter ini, jenis sumber daya adalah `ReplicationGroup` dan pengidentifikasi unik adalah nama klaster ElastiCache for Redis, misalnya `replication-group/myscalablecluster`.
- --service-namespace – Atur nilai ini ke `elasticache`.
- --scalable-dimension – Atur nilai ini ke `elasticache:replication-group:NodeGroups`.

Dalam contoh berikut, Anda menghapus kebijakan penskalaan pelacakan target yang bernama `myscalablepolicy` dari klaster ElastiCache for Redis bernama `myscalablecluster`.

Untuk Linux, macOS, atau Unix:

```
aws application-autoscaling delete-scaling-policy \
    --policy-name myscalablepolicy \
    --resource-id replication-group/myscalablecluster \
```

```
    --service-namespace elasticache \
    --scalable-dimension elasticache:replication-group:NodeGroups
```

Untuk Windows:

```
aws application-autoscaling delete-scaling-policy ^
    --policy-name myscalablepolicy ^
    --resource-id replication-group/myscalablecluster ^
    --service-namespace elasticache ^
    --scalable-dimension elasticache:replication-group:NodeGroups
```

Menghapus kebijakan penskalaan menggunakan API

Untuk menghapus kebijakan penskalaan pada klaster ElastiCache for Redis Anda, gunakan perintah AWS CLI DeleteScalingPolicy dengan parameter berikut:

- --policy-name – Nama kebijakan penskalaan.
- --resource-id – Pengidentifikasi sumber daya untuk ElastiCache for Redis. Untuk parameter ini, jenis sumber daya adalah `ReplicationGroup` dan pengidentifikasi unik adalah nama klaster ElastiCache for Redis, misalnya `replication-group/myscalablecluster`.
- --service-namespace – Atur nilai ini ke `elasticache`.
- --scalable-dimension – Atur nilai ini ke `elasticache:replication-group:NodeGroups`.

Dalam contoh berikut, Anda menghapus kebijakan penskalaan pelacakan target yang bernama `myscalablepolicy` dari klaster ElastiCache for Redis bernama `myscalablecluster`.

```
POST / HTTP/1.1
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.DeleteScalingPolicy
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS
{
    "PolicyName": "myscalablepolicy",
    "ServiceNamespace": "elasticache",
    "ResourceId": "replication-group/myscalablecluster",
    "ScalableDimension": "elasticache:replication-group:NodeGroups"
```

```
}
```

Menggunakan AWS CloudFormation untuk kebijakan Penskalaan Otomatis

Cuplikan ini menunjukkan cara membuat kebijakan target pelacakan dan menerapkannya ke sumber daya AWS::ElastiCache::ReplicationGroup menggunakan sumber daya AWS::ApplicationAutoScaling::ScalableTarget. Tindakan tersebut menggunakan fungsi intrinsik Fn::Join dan Ref untuk membangun properti `ResourceId` dengan nama logis sumber daya `AWS::ElastiCache::ReplicationGroup` yang ditentukan dalam templat yang sama.

```
ScalingTarget:
   Type: 'AWS::ApplicationAutoScaling::ScalableTarget'
   Properties:
     MaxCapacity: 3
     MinCapacity: 1
     ResourceId: !Sub replication-group/${logicalName}
     ScalableDimension: 'elasticache:replication-group:NodeGroups'
     ServiceNamespace: elasticache
     RoleARN: !Sub "arn:aws:iam::${AWS::AccountId}:role/aws-
service-role/elasticache.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ElastiCacheRG"

  ScalingPolicy:
    Type: "AWS::ApplicationAutoScaling::ScalingPolicy"
    Properties:
      ScalingTargetId: !Ref ScalingTarget
      ServiceNamespace: elasticache
      PolicyName: testpolicy
      PolicyType: TargetTrackingScaling
      ScalableDimension: 'elasticache:replication-group:NodeGroups'
      TargetTrackingScalingPolicyConfiguration:
        PredefinedMetricSpecification:
          PredefinedMetricType: ElastiCachePrimaryEngineCPUUtilization
        TargetValue: 40
```

Penskalaan terjadwal

Penskalaan berdasarkan jadwal memungkinkan Anda menskalakan aplikasi sebagai respons terhadap perubahan permintaan yang dapat diprediksi. Untuk menggunakan penskalaan terjadwal, Anda membuat tindakan terjadwal, yang memberi tahu ElastiCache for Redis untuk melakukan aktivitas penskalaan pada waktu tertentu. Saat Anda membuat tindakan terjadwal, Anda menentukan klaster ElastiCache for Redis, kapan aktivitas penskalaan harus terjadi, kapasitas minimum, dan

kapasitas maksimum. Anda dapat membuat tindakan terjadwal yang menskalakan satu kali saja atau menskalakan berdasarkan jadwal berulang.

Anda hanya dapat membuat tindakan terjadwal untuk klaster ElastiCache for Redis yang sudah ada. Anda tidak dapat membuat tindakan terjadwal pada saat yang sama saat Anda membuat klaster.

Untuk informasi selengkapnya tentang terminologi untuk pembuatan, pengelolaan, dan penghapusan tindakan terjadwal, lihat Perintah yang umum digunakan untuk pembuatan, pengelolaan, dan penghapusan tindakan terjadwal

Untuk membuat jadwal berulang:

1. Masuk ke AWS Management Console dan buka konsol Amazon ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Di panel navigasi, pilih Redis.
3. Pilih klaster yang ingin Anda tambahkan kebijakannya.
4. Pilih Kelola kebijakan Auto Scaling dari drop-down Tindakan.
5. Pilih tab Kebijakan Auto Scaling.
6. Di bagian Kebijakan penskalaan otomatis, kotak dialog Tambahkan kebijakan penskalaan akan muncul. Pilih Penskalaan terjadwal.
7. Untuk Nama Kebijakan, masukkan nama kebijakan.
8. Untuk Dimensi yang Dapat Diskalakan, pilih Serpihan.
9. Untuk Serpihan Target, pilih nilai.
10. Untuk Perulangan, pilih Berulang.
11. Untuk Frekuensi, pilih nilai masing-masing.
12. Untuk Tanggal Mulai dan Waktu mulai, pilih waktu dari kapan kebijakan akan berlaku.
13. Pilih Tambahkan kebijakan.

Untuk membuat tindakan terjadwal satu kali:

1. Masuk ke AWS Management Console dan buka konsol Amazon ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Di panel navigasi, pilih Redis.
3. Pilih klaster yang ingin Anda tambahkan kebijakannya.
4. Pilih Kelola kebijakan Auto Scaling dari drop-down Tindakan.

5. Pilih tab Kebijakan Auto Scaling.
6. Di bagian Kebijakan penskalaan otomatis, kotak dialog Tambahkan kebijakan penskalaan akan muncul. Pilih Penskalaan terjadwal.
7. Untuk Nama Kebijakan, masukkan nama kebijakan.
8. Untuk Dimensi yang Dapat Diskalakan, pilih Serpihan.
9. Untuk Serpihan Target, pilih nilai.
10. Untuk Perulangan, pilih Satu Kali.
11. Untuk Tanggal Mulai dan Waktu mulai, pilih waktu dari kapan kebijakan akan berlaku.
12. Untuk Tanggal Berakhir pilih tanggal sampai kapan kebijakan akan berlaku.
13. Pilih Tambahkan kebijakan.

Untuk menghapus tindakan terjadwal

1. Masuk ke AWS Management Console dan buka konsol Amazon ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Di panel navigasi, pilih Redis.
3. Pilih klaster yang ingin Anda tambahkan kebijakannya.
4. Pilih Kelola kebijakan Auto Scaling dari drop-down Tindakan.
5. Pilih tab Kebijakan Auto Scaling.
6. Pada bagian Kebijakan penskalaan otomatis, pilih kebijakan penskalaan otomatis, lalu pilih Hapus dari dialog Tindakan.

Untuk mengelola penskalaan terjadwal menggunakan AWS CLI

Gunakan API penskalaan otomatis-aplikasi berikut:

- put-scheduled-action
- describe-scheduled-actions
- delete-scheduled-action

Gunakan AWS CloudFormation untuk membuat tindakan terjadwal

Cuplikan ini menunjukkan cara membuat kebijakan pelacakan target dan menerapkannya ke sumber daya AWS::ElastiCache::ReplicationGroup menggunakan sumber daya

AWS::ApplicationAutoScaling::ScalableTarget. Tindakan tersebut menggunakan fungsi intrinsik Fn::Join dan Ref untuk membangun properti `ResourceId` dengan nama logis sumber daya `AWS::ElastiCache::ReplicationGroup` yang ditentukan dalam templat yang sama.

```
ScalingTarget:
   Type: 'AWS::ApplicationAutoScaling::ScalableTarget'
   Properties:
     MaxCapacity: 3
     MinCapacity: 1
     ResourceId: !Sub replication-group/${logicalName}
     ScalableDimension: 'elasticache:replication-group:NodeGroups'
     ServiceNamespace: elasticache
     RoleARN: !Sub "arn:aws:iam::${AWS::AccountId}:role/aws-
service-role/elasticache.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ElastiCacheRG"
     ScheduledActions:
       - EndTime: '2020-12-31T12:00:00.000Z'
         ScalableTargetAction:
           MaxCapacity: '5'
           MinCapacity: '2'
         ScheduledActionName: First
         Schedule: 'cron(0 18 * * ? *)'
```

## Menggunakan Penskalaan Otomatis dengan replika

Bagian berikut memberikan rincian tentang pelacakan target dan kebijakan terjadwal serta cara menerapkannya menggunakan AWS Management Console AWS CLI dan API.

### Kebijakan penskalaan pelacakan target

Dengan kebijakan penskalaan pelacakan target, Anda memilih metrik dan menetapkan nilai target. Penskalaan Otomatis ElastiCache for Redis membuat dan mengelola alarm CloudWatch yang memicu kebijakan penskalaan dan menghitung penyesuaian penskalaan berdasarkan metrik dan nilai target. Kebijakan penskalaan menambahkan atau menghapus replika secara seragam di semua serpihan yang diperlukan untuk menjaga metrik pada, atau mendekati, nilai target yang ditentukan. Selain menjaga metrik agar mendekati nilai target, kebijakan penskalaan pelacakan target juga menyesuaikan dengan fluktuasi metrik karena fluktuasi pola muatan dan meminimalkan fluktuasi cepat dalam kapasitas armada.

### Kriteria Penskalaan Otomatis untuk replika

Kebijakan Penskalaan Otomatis menentukan metrik yang telah ditentukan berikut untuk klaster Anda:

`ElastiCacheReplicaEngineCPUUtilization`: Ambang batas pemanfaatan AVG EngineCPU digabungkan di semua replika yang digunakan ElastiCache for Redis untuk memicu operasi penskalaan otomatis. Anda dapat menetapkan target pemanfaatan antara 35 persen dan 70 persen.

Ketika layanan mendeteksi bahwa metrik `ElastiCacheReplicaEngineCPUUtilization` Anda sama dengan atau lebih besar dari pengaturan Target, layanan akan meningkatkan replika di seluruh serpihan secara otomatis. ElastiCache for Redis menskalakan ke luar replika klaster Anda dengan memilih angka berdasarkan nilai yang lebih besar di antara dua nilai berikut: Variasi persen dari Target dan satu replika. Untuk menskalakan ke dalam, ElastiCache for Redis tidak akan melakukan penskalaan otomatis kecuali jika nilai metrik keseluruhan di bawah 75 persen dari Target yang Anda tentukan.

Sebagai contoh penskalaan ke luar, jika Anda memiliki 5 serpihan dan 1 replika untuk masing-masing:

Jika Target Anda melebihi 30 persen, ElastiCache for Redis menskalakan ke luar sebesar 1 replika (maks (0,3, default 1)) di semua serpihan, yang menghasilkan 5 serpihan dengan 2 replika untuk masing-masing.

Sebagai contoh penskalaan ke dalam, jika Anda telah memilih nilai Target 60 persen, ElastiCache for Redis tidak akan melakukan penskalaan otomatis sampai metrik kurang dari atau sama dengan 45 persen (25 persen di bawah Target 60 persen).

### Pertimbangan untuk Penskalaan Otomatis

Perhatikan sejumlah pertimbangan berikut:

- Kebijakan penskalaan pelacakan target mengasumsikan bahwa penskalaan ke luar harus dilakukan saat metrik yang ditentukan berada di atas nilai target. Anda tidak dapat menggunakan kebijakan penskalaan pelacakan target untuk menskalakan ke luar jika metrik yang ditentukan berada di bawah nilai target. ElastiCache for Redis menskalakan ke luar replika dengan maksimum (% deviasi dibulatkan dari Target, default 1) dari replika yang ada di semua serpihan di klaster.
- Kebijakan penskalaan pelacakan target tidak melakukan penskalaan saat metrik yang ditentukan tidak memiliki data yang mencukupi. Kebijakan penskalaan pelacakan target tidak melakukan penskalaan ke dalam karena data yang tidak mencukupi tidak ditafsirkan sebagai pemanfaatan yang rendah.

- Anda mungkin melihat kesenjangan antara nilai target dan titik data metrik aktual. Hal ini karena Penskalaan Otomatis ElastiCache for Redis selalu bertindak konservatif dengan membulatkan ke atas atau ke bawah saat menentukan jumlah kapasitas yang dapat ditambahkan atau dihapus. Hal ini mencegah layanan menambahkan kapasitas yang tidak mencukupi atau menghapus kapasitas terlalu banyak.
- Untuk memastikan ketersediaan aplikasi, layanan menskalakan ke luar secara proporsional berdasarkan metrik secepat mungkin, tetapi menskalakan ke dalam secara lebih bertahap dengan penskalaan ke dalam maksimal 1 replika di seluruh serpihan di klaster.
- Anda dapat memiliki beberapa kebijakan penskalaan pelacakan target untuk klaster ElastiCache for Redis, asalkan masing-masing kebijakan tersebut menggunakan metrik yang berbeda. Tujuan Penskalaan Otomatis ElastiCache for Redis adalah untuk selalu memprioritaskan ketersediaan, sehingga perilakunya akan berbeda-beda, tergantung apakah kebijakan pelacakan target siap untuk menskalakan ke luar atau menskalakan ke dalam. Fitur ini akan menskalakan ke luar layanan jika salah satu kebijakan pelacakan target siap untuk diskalakan ke luar. Namun, penskalaan ke dalam akan dilakukan hanya jika semua kebijakan pelacakan target (dengan porsi penskalaan ke dalam diaktifkan) siap untuk diskalakan ke dalam.
- Jangan mengedit atau menghapus alarm CloudWatch yang dikelola Penskalaan Otomatis ElastiCache for Redis untuk kebijakan penskalaan pelacakan target. Penskalaan Otomatis ElastiCache for Redis menghapus alarm secara otomatis saat Anda menghapus kebijakan penskalaan atau menghapus klaster.
- Dengan Penskalaan Otomatis ElastiCache for Redis, Anda masih dapat mengubah replika secara manual di seluruh serpihan. Penyesuaian manual ini tidak memengaruhi alarm CloudWatch saat ini yang dilampirkan ke kebijakan penskalaan, tetapi dapat memengaruhi metrik yang dapat memicu alarm CloudWatch ini.
- Alarm CloudWatch yang dikelola oleh Penskalaan Otomatis ini ditentukan melalui metrik AVG di semua serpihan di klaster. Jadi, serpihan dengan lalu lintas tertinggi dapat menghasilkan salah satu skenario berikut:
  - melakukan penskalaan saat tidak diperlukan karena pemuatan pada beberapa serpihan berlalu lintas tinggi yang memicu alarm CloudWatch
  - tidak melakukan penskalaan saat diperlukan karena AVG agregat di semua serpihan yang memengaruhi alarm agar tidak melanggar ketentuan yang ditetapkan.
- Batas default ElastiCache for Redis pada Simpul per klaster masih berlaku. Jadi, saat memilih Penskalaan Otomatis dan jika Anda mengharapkan simpul maksimum lebih dari batas default, minta peningkatan batas di Batas Layanan AWS dan pilih tipe batas Simpul per klaster per tipe instans.

- Pastikan Anda memiliki cukup ENI (Antarmuka Jaringan Elastis) yang tersedia di VPC Anda, yang diperlukan selama penskalaan ke luar. Untuk informasi selengkapnya, lihat Antarmuka Jaringan Elastis.
- Jika kapasitas yang tersedia dari EC2 tidak memadai, Penskalaan Otomatis ElastiCache for Redis tidak akan menskalakan ke luar hingga kapasitas yang sesuai tersedia atau jika Anda mengubah klaster secara manual ke tipe instans dengan kapasitas yang memadai.
- Penskalaan Otomatis ElastiCache for Redis tidak mendukung penskalaan replika dengan klaster yang memiliki `ReservedMemoryPercent` kurang dari 25 persen. Untuk informasi selengkapnya, lihat Mengelola Memori Cadangan.

Menambahkan kebijakan penskalaan

Anda dapat menambahkan kebijakan penskalaan menggunakan. AWS Management Console

Menambahkan kebijakan penskalaan menggunakan AWS Management Console

Untuk menambahkan kebijakan penskalaan otomatis ke ElastiCache Redis

1. Masuk ke AWS Management Console dan buka ElastiCache konsol Amazon di https://console.aws.amazon.com/elasticache/.
2. Di panel navigasi, pilih Redis.
3. Pilih klaster yang ingin Anda tambahi kebijakan (pilih nama klaster, bukan tombol di sebelah kirinya).
4. Pilih tab Kebijakan Auto Scaling.
5. Pilih tambahkan penskalaan dinamis.
6. Di bagian Kebijakan penskalaan, pilih Tambahkan penskalaan dinamis.
7. Untuk Nama Kebijakan, masukkan nama kebijakan.
8. Untuk Dimensi yang Dapat Diskalakan, pilih Replika dari kotak dialog.
9. Untuk nilai target, ketikkan persentase Avg dari pemanfaatan CPU yang ingin Anda pertahankan di ElastiCache Replicas. Nilai ini harus >=35 dan <=70. Replika klaster ditambahkan atau dihapus untuk menjaga metrik tetap mendekati nilai yang ditentukan.
10. (Opsional) periode pendinginan penskalaan ke dalam atau penskalaan ke luar tidak didukung dari Konsol. Gunakan AWS CLI untuk memodifikasi nilai pendinginan.
11. Untuk kapasitas Minimum, ketikkan jumlah replika minimum yang harus dipertahankan oleh kebijakan Auto Scaling ElastiCache untuk Redis.

12. Untuk kapasitas Maksimum, ketik jumlah maksimum replika yang harus ElastiCache dipertahankan oleh kebijakan Auto Scaling Redis. Nilai ini harus >=5.
13. Pilih Buat.

### Mendaftarkan Target yang Dapat Diskalakan

Anda dapat menerapkan kebijakan penskalaan berdasarkan metrik standar atau kustom. Untuk melakukannya, Anda dapat menggunakan AWS CLI atau Application Auto Scaling API. Langkah pertama adalah mendaftarkan grup replikasi Redis Anda ElastiCache untuk penskalaan otomatis Redis. ElastiCache

Sebelum Anda dapat menggunakan ElastiCache penskalaan otomatis Redis dengan cluster ElastiCache for Redis, Anda mendaftarkan cluster Anda dengan penskalaan otomatis ElastiCache Redis. Anda melakukannya untuk menentukan dimensi dan batas penskalaan yang akan diterapkan ke cluster itu. ElastiCache untuk penskalaan otomatis Redis secara dinamis menskalakan cluster ElastiCache untuk Redis di sepanjang dimensi yang `elasticache:replication-group:Replicas` dapat diskalakan, yang mewakili jumlah replika cluster per pecahan.

#### Menggunakan CLI

Untuk mendaftarkan ElastiCache cluster Anda, gunakan register-scalable-targetperintah dengan parameter berikut:

- --service-namespace – Atur nilai ini ke elasticache.
- --resource-id — Pengidentifikasi sumber daya untuk cluster. ElastiCache Untuk parameter ini, tipe sumber daya adalah ReplicationGroup dan pengidentifikasi unik adalah nama ElastiCache untuk cluster Redis, misalnya. `replication-group/myscalablecluster`
- --scalable-dimension — Atur nilai ini ke `elasticache:replication-group:Replicas`.
- --min-capacity — Jumlah minimum replika yang akan dikelola oleh untuk penskalaan otomatis ElastiCache Redis. Untuk informasi tentang hubungan antara --min-capacity, --max-capacity, dan jumlah replika dalam klaster Anda, lihat Kapasitas minimum dan maksimum.
- --max-capacity — Jumlah maksimum replika yang akan dikelola oleh untuk penskalaan otomatis ElastiCache Redis. Untuk informasi tentang hubungan antara --min-capacity, --max-capacity, dan jumlah replika dalam klaster Anda, lihat Kapasitas minimum dan maksimum.

Example

Dalam contoh berikut, Anda mendaftarkan ElastiCache untuk Redis cluster bernama`myscalablecluster`. Pendaftaran ini menunjukkan bahwa klaster harus diskalakan secara dinamis agar memiliki satu hingga 5 replika.

Untuk Linux, macOS, atau Unix:

```
aws application-autoscaling register-scalable-target \
    --service-namespace elasticache \
    --resource-id replication-group/myscalablecluster \
    --scalable-dimension elasticache:replication-group:Replicas \
    --min-capacity 1 \
    --max-capacity 5 \
```

Untuk Windows:

```
aws application-autoscaling register-scalable-target ^
    --service-namespace elasticache ^
    --resource-id replication-group/myscalablecluster ^
    --scalable-dimension elasticache:replication-group:Replicas ^
    --min-capacity 1 ^
    --max-capacity 5 ^
```

Menggunakan API

Untuk mendaftarkan ElastiCache cluster Anda, gunakan register-scalable-targetperintah dengan parameter berikut:

- ServiceNamespace — Tetapkan nilai ini ke elasticache.
- ResourceId — Pengidentifikasi sumber daya untuk cluster. ElastiCache Untuk parameter ini, tipe sumber daya adalah ReplicationGroup dan pengidentifikasi unik adalah nama ElastiCache untuk cluster Redis, misalnya. `replication-group/myscalablecluster`
- ScalableDimension — Tetapkan nilai ini ke`elasticache:replication-group:Replicas`.
- MinCapacity — Jumlah minimum replika yang akan dikelola oleh ElastiCache untuk penskalaan otomatis Redis. Untuk informasi tentang hubungan antara --min-capacity, --max-capacity, dan jumlah replika dalam klaster Anda, lihat Kapasitas minimum dan maksimum.
- MaxCapacity — Jumlah maksimum replika yang akan dikelola oleh ElastiCache untuk penskalaan otomatis Redis. Untuk informasi tentang hubungan antara --min-capacity, --max-capacity, dan jumlah replika dalam klaster Anda, lihat Kapasitas minimum dan maksimum.

Example

Dalam contoh berikut, Anda mendaftarkan cluster ElastiCache untuk Redis bernama `myscalablecluster` dengan Application Auto Scaling API. Pendaftaran ini menunjukkan bahwa klaster harus diskalakan secara dinamis agar memiliki satu hingga 5 replika.

```
POST / HTTP/1.1
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.RegisterScalableTarget
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS
{
    "ServiceNamespace": "elasticache",
    "ResourceId": "replication-group/myscalablecluster",
    "ScalableDimension": "elasticache:replication-group:Replicas",
    "MinCapacity": 1,
    "MaxCapacity": 5
}
```

Menetapkan kebijakan penskalaan

Konfigurasi kebijakan penskalaan pelacakan target direpresentasikan oleh blok JSON yang digunakan untuk mendefinisikan metrik dan nilai target. Anda dapat menyimpan konfigurasi kebijakan penskalaan sebagai blok JSON dalam file teks. Anda menggunakan file teks tersebut saat menginvokasi AWS CLI atau API Application Auto Scaling. Untuk informasi selengkapnya tentang sintaks konfigurasi kebijakan, lihat TargetTrackingScalingPolicyConfiguration di Referensi API Application Auto Scaling.

Opsi berikut tersedia untuk menetapkan konfigurasi kebijakan penskalaan pelacakan target:

Topik

- Menggunakan metrik yang sudah ditentukan sebelumnya
- Mengedit kebijakan penskalaan
- Menghapus kebijakan penskalaan
- Menggunakan AWS CloudFormation untuk kebijakan Penskalaan Otomatis

- Penskalaan terjadwal

Menggunakan metrik yang sudah ditentukan sebelumnya

Konfigurasi kebijakan penskalaan pelacakan target direpresentasikan oleh blok JSON yang digunakan untuk mendefinisikan metrik dan nilai target. Anda dapat menyimpan konfigurasi kebijakan penskalaan sebagai blok JSON dalam file teks. Anda menggunakan file teks tersebut saat menginvokasi AWS CLI atau API Application Auto Scaling. Untuk informasi selengkapnya tentang sintaks konfigurasi kebijakan, lihat TargetTrackingScalingPolicyConfiguration di Referensi API Application Auto Scaling.

Opsi berikut tersedia untuk menetapkan konfigurasi kebijakan penskalaan pelacakan target:

Topik

- Menggunakan metrik yang sudah ditentukan sebelumnya
- Menggunakan metrik khusus
- Menggunakan periode pendinginan
- Menonaktifkan aktivitas penskalaan ke dalam
- Menerapkan kebijakan penskalaan ke klaster ElastiCache for Redis

Menggunakan metrik yang sudah ditentukan sebelumnya

Dengan menggunakan metrik yang telah ditentukan sebelumnya, Anda dapat dengan cepat menentukan kebijakan penskalaan pelacakan target untuk klaster ElastiCache for Redis yang bekerja dengan pelacakan target di ElastiCache for Redis Auto Scaling. Saat ini, ElastiCache for Redis mendukung metrik yang ditentukan sebelumnya berikut di ElastiCache Replicas Auto Scaling:

`ElastiCacheReplicaEngineCPUUtilization` – Nilai rata-rata metrik EngineCPUUtilization di CloudWatch di seluruh replika di klaster ElastiCache for Redis. Nilai rata-rata metrik `EngineCPUUtilization` di CloudWatch di seluruh replika dalam klaster ElastiCache for Redis. Anda dapat menemukan nilai metrik gabungan di CloudWatch di bagian ElastiCache for Redis `ReplicationGroupId, Role` untuk ReplicationGroupId dan Replika Peran yang diperlukan.

Untuk menggunakan metrik yang sudah ditentukan sebelumnya dalam kebijakan penskalaan, Anda membuat konfigurasi pelacakan target untuk kebijakan penskalaan Anda. Konfigurasi ini harus menyertakan `PredefinedMetricSpecification` untuk metrik yang ditetapkan sebelumnya dan `TargetValue` untuk nilai target metrik tersebut.

### Menggunakan metrik khusus

Dengan menggunakan metrik kustom, Anda dapat menentukan kebijakan penskalaan pelacakan target yang memenuhi persyaratan kustom Anda. Anda dapat menentukan metrik kustom berdasarkan metrik ElastiCache for Redis apa pun yang berubah secara berbanding lurus dengan penskalaan. Tidak semua metrik ElastiCache for Redis bisa berfungsi untuk pelacakan target. Metrik harus berupa metrik penggunaan yang valid dan menjelaskan seberapa sibuk suatu instans. Nilai metrik harus meningkat atau menurun proporsinya terhadap jumlah replika dalam klaster. Peningkatan atau penurunan proporsional ini diperlukan untuk menggunakan data metrik untuk meningkatkan atau menurunkan jumlah replika secara proporsional.

#### Example

Contoh berikut menjelaskan konfigurasi pelacakan target untuk kebijakan penskalaan. Dalam konfigurasi ini, metrik kustom menyesuaikan klaster ElastiCache for Redis berdasarkan penggunaan CPU rata-rata sebesar 50 persen di seluruh replika dalam klaster bernama `my-db-cluster`.

```
{"TargetValue": 50,
    "CustomizedMetricSpecification":
    {"MetricName": "EngineCPUUtilization",
        "Namespace": "AWS/ElastiCache",
        "Dimensions": [
            {"Name": "RelicationGroup","Value": "my-db-cluster"},
            {"Name": "Role","Value": "REPLICA"}
        ],
        "Statistic": "Average",
        "Unit": "Percent"
    }
}
```

### Menggunakan periode pendinginan

Anda dapat menentukan nilai, dalam detik, untuk `ScaleOutCooldown` guna menambahkan periode pendinginan untuk menskalakan klaster Anda ke luar. Demikian pula, Anda dapat menentukan nilai, dalam detik, untuk `ScaleInCooldown` guna menambahkan periode pendinginan untuk menskalakan klaster Anda ke dalam. Untuk informasi selengkapnya tentang `ScaleInCooldown` dan `ScaleOutCooldown`, lihat TargetTrackingScalingPolicyConfiguration di Referensi API Application Auto Scaling. Contoh berikut menjelaskan konfigurasi pelacakan target untuk kebijakan penskalaan. Dalam konfigurasi ini, metrik `ElastiCacheReplicaEngineCPUUtilization` yang sudah ditentukan digunakan untuk menyesuaikan sebuah klaster ElastiCache for Redis berdasarkan

penggunaan CPU rata-rata sebesar 40 persen di seluruh replika di klaster tersebut. Konfigurasi ini menyediakan periode pendinginan penskalaan ke dalam selama 10 menit dan periode pendinginan penskalaan ke luar selama 5 menit.

```
{"TargetValue": 40.0,
    "PredefinedMetricSpecification":
    {"PredefinedMetricType": "ElastiCacheReplicaEngineCPUUtilization"
    },
    "ScaleInCooldown": 600,
    "ScaleOutCooldown": 300
}
```

Menonaktifkan aktivitas penskalaan ke dalam

Anda dapat mencegah konfigurasi kebijakan penskalaan pelacakan target dari penskalaan ke dalam klaster ElastiCache for Redis dengan menonaktifkan aktivitas penskalaan ke dalam. Menonaktifkan aktivitas penskalaan ke dalam mencegah kebijakan penskalaan menghapus replika, namun masih memungkinkan kebijakan penskalaan untuk menambahkannya sesuai kebutuhan.

Anda dapat menentukan nilai Boolean untuk `DisableScaleIn` guna mengaktifkan atau menonaktifkan aktivitas penskalaan ke dalam untuk klaster Anda. Untuk informasi selengkapnya tentang `DisableScaleIn`, lihat TargetTrackingScalingPolicyConfiguration dalam Referensi API Application Auto Scaling.

Example

Contoh berikut menjelaskan konfigurasi pelacakan target untuk kebijakan penskalaan. Dalam konfigurasi ini, metrik `ElastiCacheReplicaEngineCPUUtilization` yang sudah ditentukan menyesuaikan klaster ElastiCache for Redis berdasarkan penggunaan CPU rata-rata sebesar 40 persen di seluruh replika di klaster tersebut. Konfigurasi ini menonaktifkan aktivitas penskalaan ke dalam untuk kebijakan penskalaan.

```
{"TargetValue": 40.0,
    "PredefinedMetricSpecification":
    {"PredefinedMetricType": "ElastiCacheReplicaEngineCPUUtilization"
    },
    "DisableScaleIn": true
}
```

### Menerapkan kebijakan penskalaan ke klaster ElastiCache for Redis

Setelah mendaftarkan klaster Anda dengan penskalaan otomatis ElastiCache for Redis dan menentukan kebijakan penskalaan, Anda menerapkan kebijakan penskalaan ke klaster yang terdaftar. Untuk menerapkan kebijakan penskalaan pada klaster ElastiCache for Redis, Anda dapat menggunakan AWS CLI atau API Application Auto Scaling.

#### Menggunakan AWS CLI

Untuk menerapkan kebijakan penskalaan pada klaster ElastiCache for Redis Anda, gunakan perintah put-scaling-policy dengan parameter berikut:

- --policy-name – Nama kebijakan penskalaan.
- --policy-name – Atur nilai ini ke `TargetTrackingScaling`.
- --resource-id – Pengidentifikasi sumber daya untuk klaster ElastiCache for Redis. Untuk parameter ini, jenis sumber daya adalah ReplicationGroup dan pengidentifikasi unik adalah nama klaster ElastiCache for Redis, misalnya `replication-group/myscalablecluster`.
- --service-namespace – Atur nilai ini ke ElastiCache.
- --scalable-dimension — Atur nilai ini ke `elasticache:replication-group:Replicas`.
- --target-tracking-scaling-policy-configuration – Konfigurasi kebijakan penskalaan target-tracking yang akan digunakan untuk klaster ElastiCache for Redis.

#### Example

Dalam contoh berikut, Anda menerapkan kebijakan penskalaan pelacakan target yang bernama `myscalablepolicy` ke klaster ElastiCache for Redis bernama `myscalablecluster` dengan penskalaan otomatis ElastiCache for Redis. Untuk melakukannya, Anda menggunakan konfigurasi kebijakan yang disimpan dalam file bernama `config.json`.

Untuk Linux, macOS, atau Unix:

```
aws application-autoscaling put-scaling-policy \
    --policy-name myscalablepolicy \
    --policy-type TargetTrackingScaling \
    --resource-id replication-group/myscalablecluster \
    --service-namespace elasticache \
    --scalable-dimension elasticache:replication-group:Replicas \
    --target-tracking-scaling-policy-configuration file://config.json
```

```
{"TargetValue": 40.0,
    "PredefinedMetricSpecification":
    {"PredefinedMetricType": "ElastiCacheReplicaEngineCPUUtilization"
    },
    "DisableScaleIn": true
}
```

Untuk Windows:

```
aws application-autoscaling put-scaling-policy ^
    --policy-name myscalablepolicy ^
    --policy-type TargetTrackingScaling ^
    --resource-id replication-group/myscalablecluster ^
    --service-namespace elasticache ^
    --scalable-dimension elasticache:replication-group:Replicas ^
    --target-tracking-scaling-policy-configuration file://config.json
```

Menggunakan API

Untuk menerapkan kebijakan penskalaan pada klaster ElastiCache for Redis dengan Application Auto Scaling API, gunakan operasi API Application Auto Scaling PutScalingPolicy dengan parameter berikut:

- PolicyName – Nama kebijakan penskalaan.
- PolicyType – Atur nilai ini ke `TargetTrackingScaling`.
- ResourceID – Pengidentifikasi sumber daya untuk klaster ElastiCache for Redis. Untuk parameter ini, jenis sumber daya adalah ReplicationGroup dan pengidentifikasi unik adalah nama klaster ElastiCache for Redis, misalnya `replication-group/myscalablecluster`.
- ServiceNamespace — Atur nilai ini ke ElastiCache.
- ScalableDimension - Atur nilai ini ke `elasticache:replication-group:Replicas`.
- TargetTrackingScalingPolicyConfiguration – Konfigurasi kebijakan penskalaan pelacakan target yang akan digunakan untuk klaster ElastiCache for Redis.

Example

Dalam contoh berikut, Anda menerapkan kebijakan penskalaan pelacakan target yang bernama `scalablepolicy` ke klaster ElastiCache for Redis bernama `myscalablecluster` dengan

penskalaan otomatis ElastiCache for Redis. Anda menggunakan konfigurasi kebijakan berdasarkan pada metrik `ElastiCacheReplicaEngineCPUUtilization` yang telah ditentukan sebelumnya.

```
POST / HTTP/1.1
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.PutScalingPolicy
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS
{
    "PolicyName": "myscalablepolicy",
    "ServiceNamespace": "elasticache",
    "ResourceId": "replication-group/myscalablecluster",
    "ScalableDimension": "elasticache:replication-group:Replicas",
    "PolicyType": "TargetTrackingScaling",
    "TargetTrackingScalingPolicyConfiguration": {
        "TargetValue": 40.0,
        "PredefinedMetricSpecification":
        {
            "PredefinedMetricType": "ElastiCacheReplicaEngineCPUUtilization"
        }
    }
}
```

Mengedit kebijakan penskalaan

Anda dapat mengedit kebijakan penskalaan menggunakan AWS Management Console, AWS CLI, atau API Application Auto Scaling.

Mengedit kebijakan penskalaan menggunakan AWS Management Console

Anda hanya dapat mengedit kebijakan dengan jenis Metrik yang telah ditentukan sebelumnya dengan menggunakan AWS Management Console

1. Masuk ke AWS Management Console dan buka konsol Amazon ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Di panel navigasi, pilih Redis.

3. Pilih klaster yang ingin Anda tambahi kebijakan (pilih nama klaster, bukan tombol di sebelah kirinya).
4. Pilih tab Kebijakan Auto Scaling.
5. Di bagian Kebijakan penskalaan, pilih tombol di samping kebijakan Auto Scaling yang ingin Anda ubah, lalu pilih Ubah.
6. Buat perubahan yang diperlukan pada kebijakan.
7. Pilih Ubah.
8. Buat perubahan pada kebijakan.
9. Pilih Ubah.

Mengedit kebijakan penskalaan menggunakan AWS CLI atau API Application Auto Scaling

Anda dapat menggunakan AWS CLI atau API Application Auto Scaling untuk mengedit kebijakan penskalaan dengan cara yang sama seperti Anda menerapkan kebijakan penskalaan:

- Saat menggunakan API Application Auto Scaling, tentukan nama kebijakan yang ingin Anda edit dalam parameter `PolicyName`. Tentukan nilai baru untuk parameter yang ingin Anda ubah.

Untuk informasi selengkapnya, lihat Menerapkan kebijakan penskalaan ke klaster ElastiCache for Redis.

Menghapus kebijakan penskalaan

Anda dapat menghapus kebijakan penskalaan menggunakan AWS Management Console, Application Auto Scaling API AWS CLI atau Application Auto Scaling

Menghapus kebijakan penskalaan menggunakan AWS Management Console

Anda hanya dapat mengedit kebijakan dengan jenis Metrik yang telah ditentukan sebelumnya dengan menggunakan AWS Management Console

1. Masuk ke AWS Management Console dan buka ElastiCache konsol Amazon di https://console.aws.amazon.com/elasticache/.
2. Di panel navigasi, pilih Redis.
3. Pilih klaster yang kebijakan penskalaan otomatisnya ingin Anda hapus.
4. Pilih tab Kebijakan Auto Scaling.
5. Di bagian Kebijakan penskalaan, pilih kebijakan penskalaan otomatis, lalu pilih Hapus.

### Menghapus kebijakan penskalaan menggunakan AWS CLI atau Application Auto Scaling API

Anda dapat menggunakan Application Auto Scaling API AWS CLI atau Application Auto Scaling untuk menghapus kebijakan penskalaan dari klaster. ElastiCache

#### CLI

Untuk menghapus kebijakan penskalaan dari klaster Redis Anda ElastiCache , gunakan delete-scaling-policyperintah dengan parameter berikut:

- --policy-name – Nama kebijakan penskalaan.
- --resource-id — Pengidentifikasi sumber daya untuk cluster for Redis. ElastiCache Untuk parameter ini, tipe sumber daya adalah ReplicationGroup dan pengidentifikasi unik adalah nama ElastiCache cluster, misalnya`replication-group/myscalablecluster`.
- --service-namespace – Atur nilai ini ke ElastiCache.
- --scalable-dimension — Atur nilai ini ke `elasticache:replication-group:Replicas`.

#### Example

Dalam contoh berikut, Anda menghapus kebijakan penskalaan pelacakan target yang disebut `myscalablepolicy` dari ELC; klaster bernama `myscalablecluster`.

Untuk Linux, macOS, atau Unix:

```
aws application-autoscaling delete-scaling-policy \
    --policy-name myscalablepolicy \
    --resource-id replication-group/myscalablecluster \
    --service-namespace elasticache \
    --scalable-dimension elasticache:replication-group:Replicas \
```

Untuk Windows:

```
aws application-autoscaling delete-scaling-policy ^
    --policy-name myscalablepolicy ^
    --resource-id replication-group/myscalablecluster ^
    --service-namespace elasticache ^
    --scalable-dimension elasticache:replication-group:Replicas ^
```

API

Untuk menghapus kebijakan penskalaan dari klaster Redis, gunakan operasi DeleteScalingPolicyApplication Auto Scaling API dengan parameter berikut: ElastiCache

- PolicyName — Nama kebijakan penskalaan.
- ResourceId — Pengidentifikasi sumber daya untuk cluster for Redis. ElastiCache Untuk parameter ini, tipe sumber daya adalah ReplicationGroup dan pengidentifikasi unik adalah nama ElastiCache cluster, misalnya`replication-group/myscalablecluster`.
- ServiceNamespace — Tetapkan nilai ini ke elasticache.
- ScalableDimension — Tetapkan nilai ini ke`elasticache:replication-group:Replicas`.

Dalam contoh berikut, Anda menghapus kebijakan penskalaan pelacakan target yang diberi nama `myscalablepolicy` dari klaster Redis yang ElastiCache dinamai `myscalablecluster` dengan Application Auto Scaling API.

```
POST / HTTP/1.1
>>>>>>> mainline
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.DeleteScalingPolicy
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS
{
    "PolicyName": "myscalablepolicy",
    "ServiceNamespace": "elasticache",
    "ResourceId": "replication-group/myscalablecluster",
    "ScalableDimension": "elasticache:replication-group:Replicas"
}
```

Menggunakan AWS CloudFormation untuk kebijakan Penskalaan Otomatis

Cuplikan ini menunjukkan cara membuat tindakan terjadwal dan menerapkannya ke sumber daya AWS::ElastiCache::ReplicationGroup menggunakan sumber daya AWS::ApplicationAutoScaling::ScalableTarget. Tindakan tersebut menggunakan fungsi intrinsik Fn::Join dan Ref untuk membangun properti `ResourceId` dengan nama logis sumber daya `AWS::ElastiCache::ReplicationGroup` yang ditentukan dalam templat yang sama.

```
ScalingTarget:
   Type: 'AWS::ApplicationAutoScaling::ScalableTarget'
   Properties:
     MaxCapacity: 0
     MinCapacity: 0
     ResourceId: !Sub replication-group/${logicalName}
     ScalableDimension: 'elasticache:replication-group:Replicas'
     ServiceNamespace: elasticache
     RoleARN: !Sub "arn:aws:iam::${AWS::AccountId}:role/aws-
service-role/elasticache.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ElastiCacheRG"

  ScalingPolicy:
    Type: "AWS::ApplicationAutoScaling::ScalingPolicy"
    Properties:
      ScalingTargetId: !Ref ScalingTarget
      ServiceNamespace: elasticache
      PolicyName: testpolicy
      PolicyType: TargetTrackingScaling
      ScalableDimension: 'elasticache:replication-group:Replicas'
      TargetTrackingScalingPolicyConfiguration:
        PredefinedMetricSpecification:
          PredefinedMetricType: ElastiCacheReplicaEngineCPUUtilization
        TargetValue: 40
```

Penskalaan terjadwal

Penskalaan berdasarkan jadwal memungkinkan Anda menskalakan aplikasi sebagai respons terhadap perubahan permintaan yang dapat diprediksi. Untuk menggunakan penskalaan terjadwal, Anda membuat tindakan terjadwal, yang memberi tahu ElastiCache for Redis untuk melakukan aktivitas penskalaan pada waktu tertentu. Saat Anda membuat tindakan terjadwal, Anda menentukan klaster ElastiCache for Redis, kapan aktivitas penskalaan harus terjadi, kapasitas minimum, dan kapasitas maksimum. Anda dapat membuat tindakan terjadwal yang menskalakan satu kali saja atau menskalakan berdasarkan jadwal berulang.

Anda hanya dapat membuat tindakan terjadwal untuk klaster ElastiCache for Redis yang sudah ada. Anda tidak dapat membuat tindakan terjadwal pada saat yang sama saat Anda membuat klaster.

Untuk informasi selengkapnya tentang terminologi untuk pembuatan, pengelolaan, dan penghapusan tindakan terjadwal, lihat Perintah yang umum digunakan untuk pembuatan, pengelolaan, dan penghapusan tindakan terjadwal

Untuk membuat tindakan terjadwal satu kali:

Mirip dengan dimensi Serpihan. Lihat Penskalaan terjadwal .

Untuk menghapus tindakan terjadwal

Mirip dengan dimensi Serpihan. Lihat Penskalaan terjadwal .

Untuk mengelola penskalaan terjadwal menggunakan AWS CLI

Gunakan API penskalaan otomatis-aplikasi berikut:

- put-scheduled-action
- describe-scheduled-actions
- delete-scheduled-action

Gunakan AWS CloudFormation untuk membuat kebijakan Auto Scaling

Cuplikan ini menunjukkan cara membuat tindakan terjadwal dan menerapkannya ke sumber daya AWS::ElastiCache::ReplicationGroup menggunakan sumber daya AWS::ApplicationAutoScaling::ScalableTarget. Tindakan tersebut menggunakan fungsi intrinsik Fn::Join dan Ref untuk membangun properti `ResourceId` dengan nama logis sumber daya `AWS::ElastiCache::ReplicationGroup` yang ditentukan dalam templat yang sama.

```
ScalingTarget:
   Type: 'AWS::ApplicationAutoScaling::ScalableTarget'
   Properties:
     MaxCapacity: 0
     MinCapacity: 0
     ResourceId: !Sub replication-group/${logicalName}
     ScalableDimension: 'elasticache:replication-group:Replicas'
     ServiceNamespace: elasticache
     RoleARN: !Sub "arn:aws:iam::${AWS::AccountId}:role/aws-
service-role/elasticache.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ElastiCacheRG"
     ScheduledActions:
       - EndTime: '2020-12-31T12:00:00.000Z'
         ScalableTargetAction:
           MaxCapacity: '5'
           MinCapacity: '2'
```

```
        ScheduledActionName: First
        Schedule: 'cron(0 18 * * ? *)'
```

## Memodifikasi mode klaster

Redis adalah basis data dalam memori terdistribusi yang mendukung sharding dan replikasi. Klaster ElastiCache for Redis adalah implementasi terdistribusi dari Redis yang memungkinkan data dipartisi di beberapa simpul Redis. Klaster ElastiCache for Redis memiliki dua mode operasi, mode klaster diaktifkan (CME) dan mode klaster dinonaktifkan (CMD). Di CME, Redis berfungsi sebagai basis data terdistribusi dengan beberapa serpihan dan simpul, sedangkan di CMD, Redis berfungsi sebagai simpul tunggal.

Sebelum bermigrasi dari CMD ke CME, kondisi berikut harus dipenuhi:

> **Important**
>
> Konfigurasi mode klaster hanya dapat diubah dari mode klaster dinonaktifkan ke mode klaster diaktifkan. Konfigurasi ini tidak dapat dikembalikan.

- Klaster mungkin hanya memiliki kunci dalam basis data 0 saja.
- Aplikasi harus menggunakan klien Redis yang mampu menggunakan protokol Klaster dan menggunakan titik akhir konfigurasi.
- Failover otomatis harus diaktifkan pada klaster dengan minimal 1 replika.
- Versi mesin Redis minimum yang diperlukan untuk migrasi adalah 7.0.

Untuk bermigrasi dari CMD ke CME, konfigurasi mode klaster harus diubah dari mode klaster dinonaktifkan ke mode klaster diaktifkan. Ini adalah prosedur dua langkah yang memastikan ketersediaan klaster selama proses migrasi.

> **Note**
>
> Anda perlu menyediakan grup parameter dengan konfigurasi klaster diaktifkan, yaitu parameter klaster diaktifkan diatur ke `yes`. Jika Anda menggunakan grup parameter default, ElastiCache for Redis akan secara otomatis memilih grup parameter default yang sesuai dengan konfigurasi klaster diaktifkan. Nilai parameter klaster diaktifkan diatur ke no untuk klaster CMD. Saat klaster beralih ke mode yang kompatibel, nilai parameter klaster diaktifkan akan diperbarui ke `yes` sebagai bagian dari tindakan modifikasi.

Untuk informasi selengkapnya, lihat Mengonfigurasi parameter mesin menggunakan grup parameter

1. Persiapan – Buat klaster CME uji dan pastikan tumpukan Anda siap untuk berfungsi dengannya. ElastiCache for Redis tidak memiliki cara untuk memverifikasi kesiapan Anda. Untuk informasi selengkapnya, lihat Membuat klaster.
2. Mengubah Konfigurasi Klaster CMD yang ada ke mode klaster yang kompatibel – Dalam mode ini, akan ada serpihan tunggal yang di-deploy, dan ElastiCache for Redis akan berfungsi sebagai simpul tunggal, tetapi juga sebagai klaster serpihan tunggal. Mode yang kompatibel berarti aplikasi klien dapat menggunakan kedua protokol tersebut untuk berkomunikasi dengan klaster. Dalam mode ini, aplikasi harus dikonfigurasi ulang untuk mulai menggunakan protokol Klaster Redis dan titik akhir konfigurasi. Untuk mengubah mode klaster Redis ke mode klaster kompatibel, ikuti langkah-langkah di bawah ini:

   > **Note**
   >
   > Dalam mode kompatibel, operasi modifikasi lainnya seperti penskalaan dan versi mesin tidak diizinkan untuk klaster. Selain itu, parameter (kecuali `cacheParameterGroupName`) tidak dapat dimodifikasi saat menentukan parameter mode klaster dalam permintaan ModifyReplicationGroup.

   a. Dengan AWS Management Console, lihat Mengubah grup replikasi dan atur mode klaster ke Kompatibel
   b. Dengan API, lihat ModifyReplicationGroup dan perbarui parameter `ClusterMode` ke `compatible`.
   c. Dengan AWS CLI, lihat modify-replication-group dan perbarui parameter `cluster-mode` ke `compatible`.

   Setelah mengubah mode klaster Redis menjadi mode klaster kompatibel, API DescribeReplicationGroups akan menampilkan titik akhir konfigurasi klaster ElastiCache for Redis. Titik akhir konfigurasi klaster adalah titik akhir tunggal yang dapat digunakan oleh aplikasi untuk terhubung ke klaster. Untuk informasi selengkapnya, lihat Menemukan titik akhir koneksi.
3. Mengubah Konfigurasi Klaster ke mode klaster diaktifkan – Setelah mode klaster diatur ke mode klaster kompatibel, langkah kedua adalah memodifikasi konfigurasi klaster ke mode klaster

diaktifkan. Dalam mode ini, satu serpihan tunggal sedang berjalan, dan pelanggan sekarang dapat menskalakan klaster mereka atau memodifikasi konfigurasi klaster lainnya.

Untuk mengubah mode klaster ke diaktifkan, ikuti langkah-langkah di bawah:

Sebelum memulai, pastikan bahwa klien Redis Anda telah dimigrasikan untuk menggunakan protokol klaster dan bahwa titik akhir konfigurasi klaster tidak sedang digunakan.

a. Dengan AWS Management Console, lihat Mengubah grup replikasi dan atur mode klaster ke Diaktifkan.
b. Dengan API, lihat ModifyReplicationGroup dan perbarui parameter `ClusterMode` ke `enabled`.
c. Dengan AWS CLI, lihat modify-replication-group dan perbarui parameter `cluster-mode` ke `enabled`.

Setelah mengubah mode klaster ke diaktifkan, titik akhir akan dikonfigurasi sesuai spesifikasi klaster Redis. API DescribeReplicationGroups akan menampilkan parameter mode klaster sebagai `enabled` dan titik akhir klaster yang sekarang tersedia untuk digunakan oleh aplikasi untuk terhubung ke klaster.

Perhatikan bahwa titik akhir klaster akan berubah setelah mode klaster diubah menjadi diaktifkan. Pastikan untuk memperbarui aplikasi Anda dengan titik akhir baru.

Anda juga dapat memilih untuk kembali ke mode klaster dinonaktifkan (CMD) dari mode klaster kompatibel dan mempertahankan konfigurasi asli.

Mengubah Konfigurasi Klaster ke mode klaster dinonaktifkan dari mode klaster kompatibel

1. Dengan AWS Management Console, lihat Mengubah grup replikasi dan atur mode klaster ke Dinonaktifkan.
2. Dengan API, lihat ModifyReplicationGroup dan perbarui parameter `ClusterMode` ke `disabled`.
3. Dengan AWS CLI, lihat modify-replication-group dan perbarui parameter `cluster-mode` ke `disabled`.

Setelah mengubah mode klaster menjadi dinonaktifkan, API DescribeReplicationGroups akan menampilkan parameter mode klaster sebagai `disabled`.

# Replikasi di seluruh Wilayah AWS menggunakan penyimpanan data global

> **Note**
>
> Penyimpanan Data Global saat ini tersedia hanya untuk klaster yang dirancang sendiri.

Dengan fitur Penyimpanan Data Global untuk Redis, Anda dapat bekerja dengan replikasi yang aman, andal, cepat, dan sepenuhnya terkelola di seluruh wilayah AWS. Dengan menggunakan fitur ini, Anda dapat membuat klaster replika baca lintas-Wilayah untuk ElastiCache for Redis guna memungkinkan operasi baca berlatensi rendah dan pemulihan bencana di seluruh Wilayah AWS.

Di bagian berikutnya, Anda dapat menemukan deskripsi tentang cara bekerja dengan penyimpanan data global.

Topik



## Gambaran Umum

Setiap penyimpanan data global adalah kumpulan dari satu atau beberapa klaster yang melakukan replikasi terhadap satu sama lain.

Penyimpanan data global terdiri dari hal berikut:

- Klaster primer (aktif) – Klaster primer menerima proses tulis yang direplikasikan ke semua klaster di dalam penyimpanan data global. Klaster primer juga menerima permintaan baca.
- Klaster sekunder (pasif) – Klaster sekunder hanya menerima permintaan baca dan mereplikasikan pembaruan data dari klaster primer. Klaster sekunder harus berada di Wilayah AWS yang berbeda dari klaster primer.

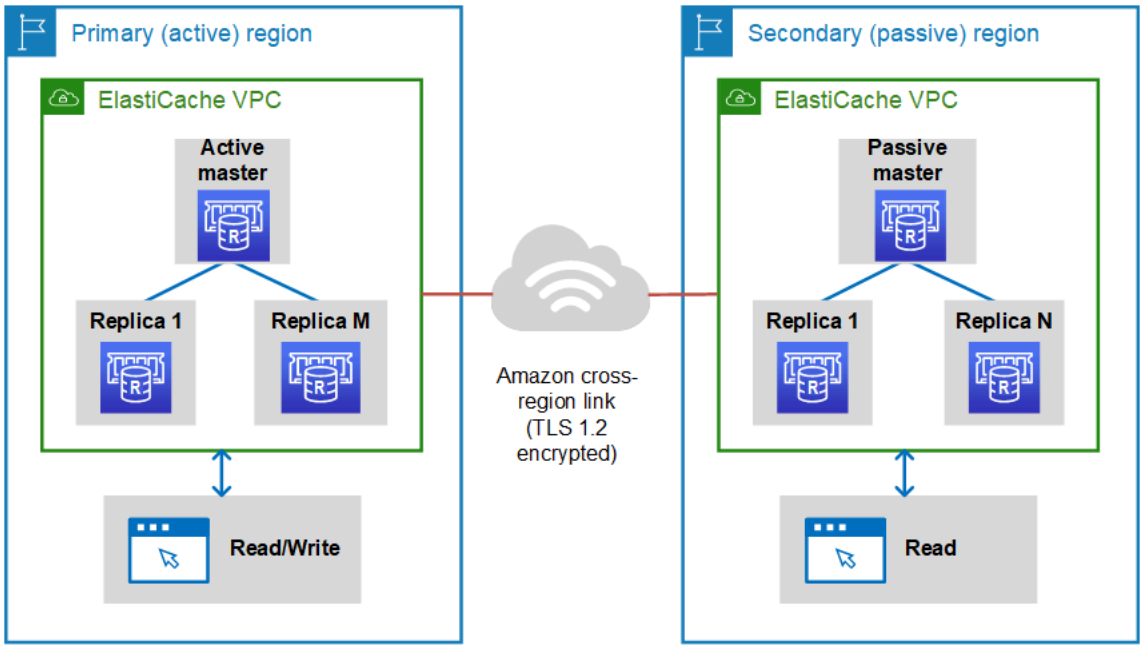
Saat Anda membuat penyimpanan data global di ElastiCache, ElastiCache for Redis secara otomatis mereplikasikan data Anda dari klaster primer ke klaster sekunder. Anda memilih Wilayah AWS

tempat data Redis harus direplikasikan lalu membuat klaster sekunder di Wilayah AWS tersebut. ElastiCache kemudian menyiapkan dan mengelola replikasi data asinkron dan otomatis di antara kedua klaster tersebut.

Menggunakan penyimpanan data global untuk Redis memberikan manfaat berikut:

- Kinerja Geolokal – Dengan menyiapkan klaster replika jarak jauh di Wilayah AWS tambahan dan menyinkronkan data Anda di antaranya, Anda dapat mengurangi latensi akses data di Wilayah AWS tersebut. Penyimpanan data global dapat membantu meningkatkan respons aplikasi Anda dengan melayani proses baca geolokal berlatensi rendah di seluruh Wilayah AWS.
- Pemulihan bencana – Jika klaster primer di penyimpanan data global mengalami penurunan, Anda dapat mempromosikan klaster sekunder sebagai klaster primer baru Anda. Anda dapat melakukannya dengan menyambung ke Wilayah AWS mana pun yang berisi klaster sekunder.

Diagram berikut menunjukkan cara kerja penyimpanan data global.




## Prasyarat dan batasan

Sebelum mulai membahas penyimpanan data global, perhatikan hal berikut:

- Datastores global didukung di AWS Wilayah berikut: Asia Pasifik (Seoul, Tokyo, Singapura, Sydney, Mumbai, dan Osaka), Eropa (Frankfurt, Paris, London, Irlandia, dan Stockholm), AS Timur (Virginia N. dan Ohio), AS Barat (California N. dan Oregon), Amerika Selatan (São Paulo), (AS-Barat dan AS-Timur), Wilayah Kanada (Tengah), China AWS GovCloud (Beijing) dan Ningxia)

- Semua klaster—primer dan sekunder—di penyimpanan data global Anda harus sama dalam jumlah simpul primer, jenis simpul, versi mesin, dan jumlah serpihan (dalam kasus mode klaster diaktifkan). Setiap klaster di penyimpanan data global Anda dapat memiliki jumlah replika baca yang berbeda untuk mengakomodasi lalu lintas baca lokal untuk klaster tersebut.

  Replikasi harus diaktifkan jika Anda berencana menggunakan klaster simpul tunggal yang telah ada.
- Datastores global tidak didukung pada instance yang lebih tua dari m5 atau r5.
- Anda dapat mengatur replikasi untuk klaster utama dari satu AWS Wilayah ke cluster sekunder hingga dua AWS Wilayah lainnya.

  > **Note**
  >
  > Pengecualian hal ini adalah Wilayah Tiongkok (Beijing) dan Tiongkok (Ningxia), di mana replikasi hanya dapat terjadi di antara kedua wilayah itu.
- Anda dapat bekerja dengan penyimpanan data global hanya dalam klaster VPC. Untuk informasi selengkapnya, lihat Pola Akses untuk Mengakses Cache ElastiCache di Amazon VPC. Penyimpanan data global tidak didukung saat Anda menggunakan EC2-Classic. Untuk informasi selengkapnya, lihat EC2-Classic pada Panduan Pengguna Amazon EC2 User Guide untuk Instans Linux.

  > **Note**
  >
  > Untuk saat ini, Anda tidak dapat menggunakan penyimpanan data global di Menggunakan zona lokal dengan ElastiCache .
- ElastiCache tidak mendukung autofailover dari satu AWS Wilayah ke Wilayah lainnya. Jika diperlukan, Anda dapat mempromosikan klaster sekunder secara manual. Untuk contoh, lihat Mempromosikan klaster sekunder menjadi primer.
- Untuk melakukan bootstrap dari data yang telah ada, gunakan klaster yang telah ada sebagai primer untuk membuat penyimpanan data global. Kami tidak mendukung penambahan klaster yang telah ada sebagai sekunder. Proses penambahan klaster sebagai sekunder akan menghapus data, yang dapat mengakibatkan kehilangan data.
- Pembaruan parameter diterapkan untuk semua klaster saat Anda mengubah grup parameter lokal dari klaster yang termasuk dalam penyimpanan data global.

- Anda dapat menskalakan klaster wilayah baik secara vertikal (menaikkan dan menurunkan skala) maupun horizontal (menskalakan ke dalam dan ke luar). Anda dapat menskalakan klaster dengan mengubah penyimpanan data global. Semua klaster regional pada penyimpanan data global kemudian diskalakan tanpa gangguan. Untuk informasi selengkapnya, lihat Penskalaan ElastiCache untuk Redis .
- Penyimpanan data global mendukung enkripsi data diam, enkripsi data bergerak, dan AUTH Redis.
- Datastores global tidak mendukung Internet Protocol versi 6 (IPv6).
- Kunci dukungan datastores global. AWS KMS Untuk informasi selengkapnya, lihat Konsep layanan pengelolaan kunci AWS dalam Panduan Developer AWS Key Management Service .

> **Note**
>
> Penyimpanan data global mendukung olahpesan pub/sub dengan ketentuan berikut:
>
> - Untuk mode klaster dinonaktifkan, pub/sub didukung sepenuhnya. Peristiwa yang diterbitkan di cluster utama AWS Wilayah primer disebarkan ke AWS Wilayah sekunder.
> - Untuk mode klaster diaktifkan, hal berikut berlaku:
>   - Untuk acara yang dipublikasikan yang tidak berada di ruang kunci, hanya pelanggan di AWS Wilayah yang sama yang menerima acara tersebut.
>   - Untuk acara keyspace yang dipublikasikan, pelanggan di semua AWS Wilayah menerima acara.

## Menggunakan penyimpanan data global (konsol)

Untuk membuat penyimpanan data global menggunakan konsol, ikuti proses dua langkah ini:

1. Buat klaster primer, baik dengan menggunakan klaster yang telah ada atau membuat klaster baru. Mesin harus Redis versi 5.0.6 atau lebih baru.
2. Tambahkan hingga dua cluster sekunder di AWS Wilayah yang berbeda, sekali lagi menggunakan mesin Redis 5.0.6 atau yang lebih baru.

Prosedur berikut memandu Anda tentang cara membuat datastore global untuk Redis dan melakukan operasi lain menggunakan konsol ElastiCache for Redis.

Topik

### Membuat penyimpanan data global menggunakan klaster yang telah ada

Dalam skenario ini, Anda menggunakan klaster sudah telah ada untuk berperan sebagai primer dari penyimpanan data global yang baru. Kemudian membuat klaster sekunder baca-saja di Wilayah AWS yang berbeda. Klaster sekunder ini menerima pembaruan otomatis dan asinkron dari klaster primer.

> **Important**
>
> Klaster yang sudah ada harus menggunakan mesin Redis 5.0.6 atau lebih baru.

Untuk membuat penyimpanan data global menggunakan klaster yang sudah ada

1. Masuk ke AWS Management Console dan buka ElastiCache konsol di https://console.aws.amazon.com/elasticache/.
2. Pada panel navigasi, pilih Redis lalu pilih klaster.
3. Untuk Tindakan, pilih Menyiapkan Penyimpanan Data Global.
4. Pada halaman Menyiapkan Penyimpanan Data Global, lakukan hal berikut:
   - Masukkan nilai untuk Sufiks Nama Penyimpanan Data Global: Sufiks ini digunakan untuk menghasilkan nama yang unik untuk penyimpanan data global. Anda dapat mencari penyimpanan data global dengan menggunakan sufiks yang Anda tentukan di sini.
   - (Opsional) Masukkan nilai untuk Deskripsi.
5. Di bawah Detail klaster sekunder, pilih Wilayah AWS yang akan digunakan menyimpan klaster.

6. Di bawah Pengaturan Redis, masukkan nilai untuk Nama dan, secara opsional, masukkan Deskripsi untuk klaster.
7. Biarkan opsi berikut tanpa perubahan. Opsi tersebut sudah diisi untuk disesuaikan dengan konfigurasi klaster primer, Anda tidak dapat mengubahnya.

   - Versi mesin
   - Jenis simpul
   - Grup parameter

     > **Note**
     >
     > ElastiCache membuat otomatis grup parameter baru dari nilai grup parameter yang disediakan dan menerapkan grup parameter baru ke cluster. Gunakan grup parameter baru ini untuk mengubah parameter pada penyimpanan data global. Setiap grup parameter yang dihasilkan otomatis berkaitan dengan hanya satu klaster dan karena itu, hanya satu penyimpanan data global.

   - Jumlah serpihan
   - Enkripsi data diam – Mengaktifkan enkripsi pada data yang disimpan di disk. Untuk informasi selengkapnya, lihat Enkripsi data diam.

     > **Note**
     >
     > Anda dapat menyediakan kunci enkripsi yang berbeda dengan memilih kunci AWS KMS yang Dikelola Pelanggan dan memilih kunci. Untuk informasi selengkapnya, lihat Menggunakan kunci AWS KMS yang Dikelola Pelanggan.

   - Enkripsi Data Bergerak – Mengaktifkan enkripsi pada data selama pengiriman. Untuk informasi selengkapnya, lihat Enkripsi data bergerak. Untuk mesin Redis versi 6.0 dan di atasnya, jika Anda mengaktifkan enkripsi data bergerak, Anda akan diminta untuk menentukan salah satu dari opsi Kontrol Akses berikut:

     - Tidak ada Kontrol Akses – Ini adalah pengaturan default. Artinya menunjukkan tidak adanya batasan.
     - Daftar Kontrol Akses Grup Pengguna – Pilih grup pengguna dengan sejumlah pengguna yang ditentukan dan izin untuk operasi yang tersedia. Untuk informasi selengkapnya, lihat Mengelola Grup Pengguna dengan Konsol dan CLI.

- Pengguna Default Redis AUTH – Mekanisme autentikasi untuk server Redis. Untuk informasi selengkapnya, lihat Redis AUTH.

8. (Opsional) Jika diperlukan, perbarui pengaturan klaster sekunder yang tersisa. Pengaturan sudah diisi sebelumnya dengan nilai yang sama dengan klaster primer, tetapi Anda dapat memperbarui nilainya untuk memenuhi persyaratan khusus untuk klaster tersebut.

   - Port
   - Jumlah replika
   - Grup subnet
   - Zona Ketersediaan Pilihan
   - Grup keamanan
   - Dikelola Pelanggan (kunci AWS KMS)
   - Token AUTH Redis
   - Aktifkan cadangan otomatis
   - Periode retensi cadangan
   - Jendela cadangan
   - Jendela pemeliharaan
   - Topik untuk notifikasi SNS

9. Pilih Buat. Melakukan hal ini akan mengatur status penyimpanan data global ke Sedang Membuat. Status berubah menjadi Mengubah setelah klaster primer dikaitkan dengan penyimpanan data global dan klaster sekunder dalam status Mengaitkan.

   Setelah klaster primer dan klaster sekunder dikaitkan dengan penyimpanan data global, status berubah menjadi Tersedia. Pada tahap ini, Anda memiliki klaster primer yang menerima proses baca dan tulis serta klaster sekunder yang menerima proses baca yang direplikasi dari klaster primer.

   Halaman Redis diperbarui untuk menunjukkan apakah klaster bagian dari penyimpanan data global, termasuk:

   - Penyimpanan Data Global – Nama penyimpanan data global yang menjadi asal klaster.
   - Peran Penyimpanan Data Global – Peran klaster, baik primer atau sekunder.

Anda dapat menambahkan hingga satu cluster sekunder tambahan di AWS Wilayah yang berbeda. Untuk informasi selengkapnya, lihat Menambahkan Wilayah ke penyimpanan data global.

Membuat penyimpanan data global baru menggunakan klaster primer baru

Jika Anda memilih untuk membuat penyimpanan data global dengan klaster baru, gunakan prosedur berikut.

1. Masuk ke AWS Management Console dan buka ElastiCache konsol di https://console.aws.amazon.com/elasticache/.
2. Pada panel navigasi, pilih Penyimpanan Data Global lalu pilih Buat penyimpanan data global.
3. Pada Pengaturan klaster primer, lakukan hal berikut:
    a. Untuk Mode klaster, pilih Diaktifkan atau Dinonaktifkan.
    b. Untuk info Global Datastore masukkan nilai untuk Nama. ElastiCache menggunakan akhiran untuk menghasilkan nama unik untuk datastore global. Anda dapat mencari penyimpanan data global dengan menggunakan sufiks yang Anda tentukan di sini.
    c. (Opsional) Masukkan nilai untuk Deskripsi Penyimpanan Data Global.
4. Di bawah Klaster Wilayah:
    a. Untuk Wilayah, pilih AWS Wilayah yang tersedia.
    b. Pilih Buat klaster wilayah baru atau Gunakan klaster wilayah yang ada
    c. Jika Anda memilih Buat klaster wilayah baru, di bawah Info klaster, masukkan nama dan deskripsi opsional klaster.
    d. Untuk Lokasi, sebaiknya Anda menerima pengaturan default untuk Multi-AZ dan Failover Otomatis.
5. Di bawah Pengaturan klaster
    a. Untuk Versi mesin, pilih versi yang tersedia, yaitu 5.0.6 atau yang lebih baru.
    b. Untuk Port, gunakan port default, 6379. Jika Anda ingin menggunakan port yang berbeda, tuliskan nomor port tersebut.
    c. Untuk Grup parameter, pilih grup parameter atau buat yang baru. Grup parameter mengontrol parameter runtime klaster Anda. Untuk informasi selengkapnya tentang grup parameter, lihat Parameter spesifik Redis dan Membuat grup parameter.

> **Note**
>
> Saat Anda memilih grup parameter untuk menetapkan nilai konfigurasi mesin, grup parameter tersebut diterapkan ke semua klaster di penyimpanan data global. Pada halaman Grup Parameter, atribut Global ya/tidak menunjukkan apakah grup parameter adalah bagian dari penyimpanan data global.

d. Untuk Jenis Simpul, pilih panah bawah
( ).
Pada kotak dialog Ubah jenis simpul, pilih nilai untuk Keluarga instans untuk jenis simpul yang Anda inginkan. Kemudian pilih jenis simpul yang ingin Anda gunakan untuk klaster ini, lalu pilih Simpan.

   Untuk informasi selengkapnya, lihat Memilih ukuran simpul Anda.

   Jika Anda memilih jenis simpul r6gd, tingkatan data diaktifkan secara otomatis. Untuk informasi selengkapnya, lihat Tingkatan data.

e. Jika Anda membuat klaster Redis (mode klaster dinonaktifkan):

   Untuk Jumlah replika, pilih jumlah replika baca yang Anda inginkan untuk klaster ini.

f. Jika Anda membuat klaster Redis (mode klaster diaktifkan):

   i. Untuk Jumlah serpihan, pilih jumlah serpihan (partisi/grup simpul) yang Anda inginkan untuk klaster Redis (mode klaster diaktifkan) ini.

      Untuk beberapa versi Redis (mode klaster aktif), Anda dapat mengubah jumlah serpihan di klaster Anda secara dinamis:

      - Redis 3.2.10 dan yang lebih baru – Jika klaster Anda menjalankan Redis 3.2.10 atau versi di atasnya, Anda dapat mengubah jumlah serpihan di klaster Anda secara dinamis. Untuk informasi selengkapnya, lihat Penskalaan klaster di Redis (Mode Klaster Diaktifkan).
      - Redis versi lainnya – Jika klaster Anda menjalankan versi Redis sebelum versi 3.2.10, ada pendekatan lain. Untuk mengubah jumlah serpihan di klaster Anda, buat klaster baru dengan jumlah serpihan baru. Untuk informasi selengkapnya, lihat Melakukan pemulihan dari cadangan ke dalam cache baru.

ii. Untuk Replika per serpihan, pilih jumlah simpul replika baca yang Anda inginkan dalam setiap serpihan.

Pembatasan berikut berlaku untuk Redis (mode klaster aktif).

- Jika Multi-AZ diaktifkan, pastikan bahwa Anda memiliki setidaknya satu replika per serpihan.
- Replika berjumlah sama untuk setiap serpihan saat membuat klaster menggunakan konsol.
- Jumlah replika baca per serpihan bersifat tetap dan tidak dapat diubah. Jika Anda membutuhkan lebih banyak atau lebih sedikit replika per serpihan (API/CLI: grup simpul), Anda harus membuat klaster baru dengan jumlah replika yang baru. Untuk informasi selengkapnya, lihat Melakukan seeding klaster yang dirancang sendiri dengan cadangan yang dibuat secara eksternal.

6. Untuk pengaturan grup Subnet, pilih subnet yang ingin Anda terapkan ke cluster ini. ElastiCache menyediakan grup subnet IPv4 default atau Anda dapat memilih untuk membuat yang baru. Untuk IPv6, Anda perlu membuat grup subnet dengan blok CIDR IPv6. Jika Anda memilih tumpukan ganda, Anda harus memilih jenis IP Discovery, baik IPv6 atau IPv4.

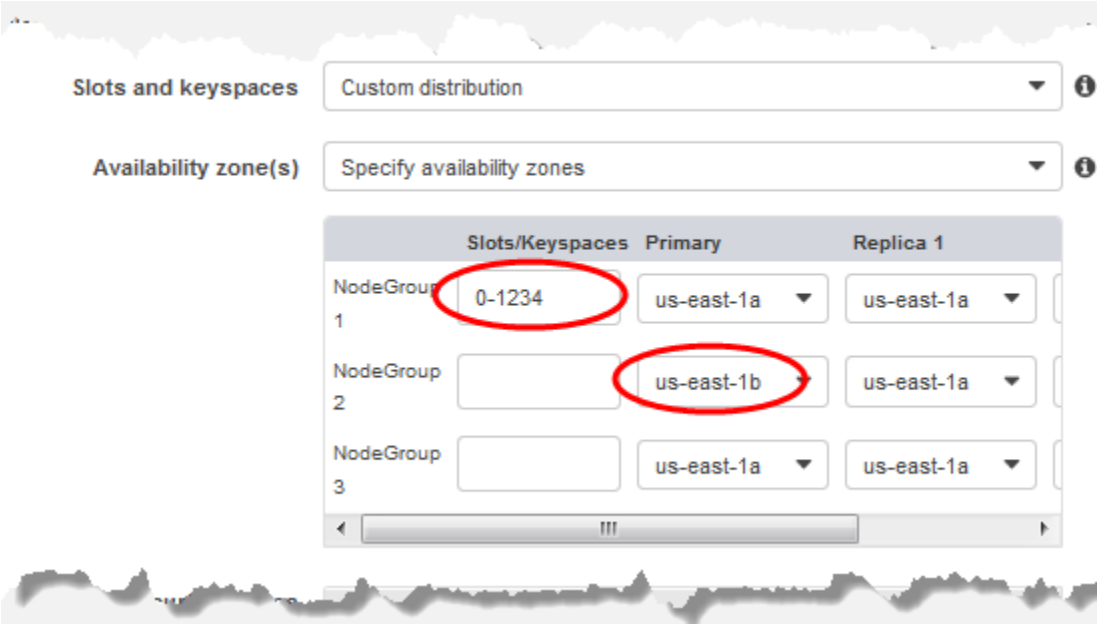
   Untuk informasi selengkapnya, lihat Membuat subnet di VPC Anda.

7. Untuk Penempatan Zona Ketersediaan, Anda memiliki dua opsi:

   - Tidak ada preferensi — ElastiCache memilih Availability Zone.
   - Tentukan zona ketersediaan – Anda menentukan Zona Ketersediaan untuk setiap klaster.

     Jika Anda memilih untuk menentukan Zona Ketersediaan, untuk setiap klaster di setiap serpihan, pilih Zona Ketersediaan dari daftar.

   Untuk informasi selengkapnya, lihat Memilih wilayah dan zona ketersediaan.




Menentukan Ruang Kunci dan Zona Ketersediaan

8. Pilih Berikutnya
9. Di bawah Pengaturan lanjutan Redis

   - Untuk Keamanan:

     i. Untuk mengenkripsi data Anda, Anda memiliki opsi berikut:

     - Enkripsi saat diam – Mengaktifkan enkripsi pada data yang disimpan di disk. Untuk informasi selengkapnya, lihat Enkripsi Data Diam.

       > **Note**
       >
       > Anda memiliki opsi untuk menyediakan kunci enkripsi yang berbeda dengan memilih kunci AWS KMS yang Dikelola Pelanggan dan memilih kunci. Untuk informasi selengkapnya, lihat Menggunakan kunci yang dikelola pelanggan dari AWS KMS.

     - Enkripsi Data Bergerak – Mengaktifkan enkripsi pada data selama pengiriman. Untuk informasi selengkapnya, lihat Enkripsi Data Bergerak. Untuk mesin Redis versi 6.0 dan di atasnya, jika Anda mengaktifkan Enkripsi data bergerak, Anda akan diminta untuk menentukan salah satu dari opsi Kontrol Akses berikut:

- Tidak ada Kontrol Akses – Ini adalah pengaturan default. Ini menunjukkan tidak ada pembatasan akses pengguna ke klaster.
- Daftar Kontrol Akses Grup Pengguna – Pilih grup pengguna dengan kelompok pengguna tertentu yang dapat mengakses klaster. Untuk informasi selengkapnya, lihat Mengelola Grup Pengguna dengan Konsol dan CLI.
- Pengguna Default Redis AUTH – Mekanisme autentikasi untuk server Redis. Untuk informasi selengkapnya, silakan lihat Redis AUTH.

- Redis AUTH - Mekanisme autentikasi untuk server Redis. Untuk informasi selengkapnya, silakan lihat Redis AUTH.

> **Note**
>
> Untuk versi Redis antara 3.2.6 dan seterusnya, kecuali versi 3.2.10, Redis AUTH adalah satu-satunya pilihan.

ii. Untuk Grup keamanan, pilih grup keamanan yang Anda inginkan untuk klaster ini. Grup keamanan bertindak sebagai firewall untuk mengontrol akses jaringan ke klaster Anda. Anda dapat menggunakan grup keamanan default untuk VPC Anda atau membuat yang baru.

Untuk informasi grup keamanan selengkapnya, lihat Grup Keamanan untuk VPC Anda di Panduan Pengguna Amazon VPC.

10. Untuk pencadangan otomatis terjadwal rutin, pilih Aktifkan pencadangan otomatis, lalu masukkan jumlah hari yang diinginkan untuk menyimpan cadangan otomatis sebelum dihapus secara otomatis. Jika Anda tidak ingin melakukan pencadangan otomatis terjadwal rutin, hapus kotak centang Aktifkan pencadangan otomatis. Apa pun pilihannya, Anda selalu bisa membuat pencadangan secara manual.

    Untuk informasi selengkapnya pencadangan dan pemulihan Redis, lihat Melakukan snapshot dan pemulihan.

11. (Opsional) Menentukan jendela pemeliharaan. Jendela pemeliharaan adalah waktu yang biasanya satu jam setiap minggu saat ElastiCache menjadwalkan pemeliharaan sistem untuk klaster Anda. Anda dapat mengizinkan ElastiCache untuk memilih hari dan waktu untuk jendela pemeliharaan Anda (Tidak ada preferensi), atau Anda dapat memilih hari, waktu, dan durasi sendiri (Tentukan jendela pemeliharaan). Jika Anda memilih Tentukan jendela pemeliharaan dari

daftar, pilih Hari mulai, Waktu mulai, dan Durasi (dalam jam) untuk jendela pemeliharaan. Waktu yang digunakan adalah UCT.

Untuk informasi selengkapnya, lihat Mengelola pemeliharaan.

12. (Opsional) Untuk Log:
    - Di bawah Format log, pilih salah satu dari Teks atau JSON.
    - Di bawah Jenis Tujuan, pilih CloudWatch Log atau Kinesis Firehose.
    - Di bawah Tujuan log, pilih Buat baru dan masukkan nama grup CloudWatch log Log atau nama aliran Firehose Anda, atau pilih Pilih yang ada, lalu pilih nama grup CloudWatch log Log atau nama aliran Firehose Anda,
13. Untuk Tag, untuk membantu mengelola kluster dan ElastiCache sumber daya lainnya, Anda dapat menetapkan metadata Anda sendiri ke setiap sumber daya dalam bentuk tag. Untuk informasi selengkapnya, lihat Menandai sumber daya ElastiCache Anda.
14. Tinjau semua entri dan pilihan Anda, lalu lakukan koreksi yang diperlukan. Saat Anda siap, pilih Berikutnya.
15. Setelah Anda mengonfigurasi klaster di langkah sebelumnya, sekarang Anda akan mengonfigurasi detail klaster sekunder.
16. Di bawah kluster Regional, pilih AWS Wilayah tempat klaster berada.
17. Di bawah Info klaster, masukkan nama dan deskripsi opsional klaster.
18. Opsi berikut sudah diisi sebelumnya untuk menyesuaikan konfigurasi klaster primer dan tidak dapat diubah:
    - Lokasi
    - Versi mesin
    - Jenis instans
    - Jenis simpul
    - Jumlah serpihan
    - Grup parameter

> **Note**
>
> ElastiCache membuat otomatis grup parameter baru dari nilai grup parameter yang disediakan dan menerapkan grup parameter baru ke cluster. Gunakan grup parameter baru ini untuk mengubah parameter pada penyimpanan data global. Setiap grup

parameter yang dihasilkan otomatis berkaitan dengan hanya satu klaster dan karena itu, hanya satu penyimpanan data global.

- Enkripsi data diam – Mengaktifkan enkripsi pada data yang disimpan di disk. Untuk informasi selengkapnya, lihat Enkripsi data diam.

> **Note**
>
> Anda dapat menyediakan kunci enkripsi yang berbeda dengan memilih kunci AWS KMS yang Dikelola Pelanggan dan memilih kunci. Untuk informasi selengkapnya, lihat Menggunakan kunci AWS KMS yang Dikelola Pelanggan.

- Enkripsi Data Bergerak – Mengaktifkan enkripsi pada data selama pengiriman. Untuk informasi selengkapnya, lihat Enkripsi data bergerak. Untuk mesin Redis versi 6.4 dan di atasnya, jika Anda mengaktifkan enkripsi data bergerak, Anda akan diminta untuk menentukan salah satu dari opsi Kontrol Akses berikut:
  - Tidak ada Kontrol Akses – Ini adalah pengaturan default. Ini menunjukkan tidak ada pembatasan akses pengguna ke klaster.
  - Daftar Kontrol Akses Grup Pengguna – Pilih grup pengguna dengan kelompok pengguna tertentu yang dapat mengakses klaster. Untuk informasi selengkapnya, lihat Mengelola Grup Pengguna dengan Konsol dan CLI.
  - Pengguna Default Redis AUTH – Mekanisme autentikasi untuk server Redis. Untuk informasi selengkapnya, lihat Redis AUTH.

> **Note**
>
> Untuk versi Redis antara 4.0.2, saat Enkripsi data bergerak pertama kali didukung, dan 6.0.4, AUTH Redis adalah satu-satunya pilihan.

Pengaturan klaster sekunder lainnya telah diisi dengan nilai yang sama dengan klaster primer, namun yang berikut ini dapat diperbarui demi memenuhi persyaratan untuk klaster tersebut:

- Port
- Jumlah replika
- Grup subnet

- Zona Ketersediaan Pilihan
- Grup keamanan
- Dikelola Pelanggan (kunci AWS KMS)
- Token AUTH Redis
- Aktifkan cadangan otomatis
- Periode retensi cadangan
- Jendela cadangan
- Jendela pemeliharaan
- Topik untuk notifikasi SNS

19. Pilih Buat. Tindakan ini mengatur status penyimpanan data global ke Sedang Membuat. Setelah klaster primer dan klaster sekunder dikaitkan dengan penyimpanan data global, status berubah menjadi Tersedia. Anda memiliki klaster primer yang menerima proses baca dan tulis serta klaster sekunder yang menerima proses baca yang direplikasi dari klaster primer.

    Halaman Redis juga diperbarui untuk menunjukkan apakah klaster bagian dari penyimpanan data global, termasuk yang berikut:

    - Penyimpanan Data Global – Nama penyimpanan data global yang menjadi asal klaster.
    - Peran Penyimpanan Data Global – Peran klaster, baik primer atau sekunder.

Anda dapat menambahkan hingga satu cluster sekunder tambahan di AWS Wilayah yang berbeda. Untuk informasi selengkapnya, lihat Menambahkan Wilayah ke penyimpanan data global.

Melihat detail penyimpanan data global

Anda dapat melihat detail penyimpanan data global yang telah ada dan juga mengubahnya pada halaman Penyimpanan Data Global.

Untuk melihat detail penyimpanan data global

1. Masuk ke AWS Management Console dan buka ElastiCache konsol di https://console.aws.amazon.com/elasticache/.
2. Pada panel navigasi, pilih Penyimpanan Data Global lalu pilih penyimpanan data global yang tersedia.

Kemudian Anda dapat memeriksa properti penyimpanan data global berikut:

- Nama Penyimpanan Data Global: Nama penyimpanan data global
- Deskripsi: Deskripsi tentang penyimpanan data global
- Status: Opsinya meliputi:
  - Sedang Membuat
  - Mengubah
  - Tersedia
  - Deleting
  - Primer-Saja - Status ini menunjukkan bahwa penyimpanan data global hanya berisi klaster primer. Dapat berarti semua klaster sekunder telah dihapus atau tidak berhasil dibuat.
- Mode Klaster: Diaktifkan atau dinonaktifkan
- Versi Mesin Redis: Versi mesin Redis yang menjalankan penyimpanan data global
- Jenis Simpul Instans: Jenis simpul yang digunakan untuk penyimpanan data global
- Enkripsi data diam: Diaktifkan atau dinonaktifkan
- Enkripsi data bergerak: Diaktifkan atau dinonaktifkan
- AUTH Redis: Diaktifkan atau dinonaktifkan

Anda dapat membuat perubahan berikut pada penyimpanan data global:

- Menambahkan Wilayah ke penyimpanan data global
- Menghapus Wilayah dari penyimpanan data global
- Mempromosikan klaster sekunder menjadi primer
- Mengubah penyimpanan data global

Halaman Penyimpanan Data Global juga mencantumkan klaster individual yang membentuk penyimpanan data global dan properti berikut untuk masing-masing:

- Wilayah - AWS Wilayah tempat cluster disimpan
- Peran - Bisa primer atau sekunder
- Nama klaster: Nama klaster
- Status - Opsinya meliputi:
  - Mengaitkan - Klaster sedang dalam proses dikaitkan dengan penyimpanan data global
  - Dikaitkan - Klaster sudah dikaitkan dengan penyimpanan data global

  - Memisahkan - Proses menghapus klaster sekunder dari penyimpanan data global menggunakan nama penyimpanan data global. Setelah ini, cluster sekunder tidak lagi menerima pembaruan dari cluster primer tetapi tetap sebagai cluster mandiri di AWS Wilayah itu.
  - Terpisah- Klaster sekunder telah dihapus dari penyimpanan data global dan sekarang menjadi klaster mandiri di Wilayah AWS .
- Kelambatan Replika Datastore Global - Menunjukkan satu nilai per AWS Wilayah sekunder di datastore global. Ini adalah ketertinggalan antara simpul primer dari Wilayah sekunder dan simpul primer dari Wilayah primer. Untuk Redis dengan mode klaster diaktifkan, lag ini menunjukkan penundaan maksimum (dalam detik) di antara serpihan.

### Menambahkan Wilayah ke penyimpanan data global

Anda dapat menambahkan hingga satu AWS Wilayah tambahan ke datastore global yang ada. Dalam skenario ini, Anda membuat klaster hanya-baca di AWS Wilayah terpisah yang menerima pembaruan otomatis dan asinkron dari cluster utama.

Untuk menambahkan AWS Region ke datastore global

1. Masuk ke AWS Management Console dan buka ElastiCache konsol di https://console.aws.amazon.com/elasticache/.
2. Pada panel navigasi, pilih Penyimpanan Data Global lalu untuk Nama Penyimpanan Data Global pilih satu penyimpanan data global.
3. Pilih Tambah Wilayah, dan pilih AWS Wilayah tempat klaster sekunder berada.
4. Di bawah Pengaturan Redis, masukkan nilai untuk Nama dan, secara opsional, masukkan Deskripsi untuk klaster.
5. Biarkan opsi berikut tanpa perubahan. Opsi sudah diisi untuk disesuaikan dengan konfigurasi klaster primer, dan Anda tidak dapat mengubahnya.
   - Versi mesin
   - Jenis instans
   - Jenis simpul
   - Jumlah serpihan
   - Grup parameter

> **Note**
>
> ElastiCache membuat otomatis grup parameter baru dari nilai grup parameter yang disediakan dan menerapkan grup parameter baru ke cluster. Gunakan grup parameter baru ini untuk mengubah parameter pada penyimpanan data global. Setiap grup parameter yang dihasilkan otomatis berkaitan dengan hanya satu klaster dan karena itu, hanya satu penyimpanan data global.

- Enkripsi data diam

> **Note**
>
> Anda dapat menyediakan kunci enkripsi yang berbeda dengan memilih kunci AWS KMS yang Dikelola Pelanggan dan memilih kunci.

- Enkripsi dalam bergerak
- AUTH Redis

6. (Opsional) Perbarui pengaturan klaster sekunder yang tersisa. Pengaturan sudah diisi sebelumnya dengan nilai yang sama dengan klaster primer, tetapi Anda dapat memperbarui nilainya untuk memenuhi persyaratan khusus untuk klaster tersebut.

   - Port
   - Jumlah replika
   - Grup subnet
   - Zona Ketersediaan Pilihan
   - Grup keamanan
   - Kunci AWS KMS yang Dikelola Pelanggan)
   - Token AUTH Redis
   - Aktifkan cadangan otomatis
   - Periode retensi cadangan
   - Jendela cadangan
   - Jendela pemeliharaan
   - Topik untuk notifikasi SNS

7. Pilih Tambahkan.

### Mengubah penyimpanan data global

Anda dapat mengubah properti klaster wilayah. Hanya satu operasi perubahan dapat berlangsung pada penyimpanan data global, dengan pengecualian pada operasi meningkatkan klaster sekunder menjadi primer. Untuk informasi selengkapnya, lihat Mempromosikan klaster sekunder menjadi primer.

Untuk mengubah penyimpanan data global

1. Masuk ke AWS Management Console dan buka ElastiCache konsol di https://console.aws.amazon.com/elasticache/.
2. Pada panel navigasi, pilih Penyimpanan Data Global lalu untuk Nama Penyimpanan Data Global, pilih satu penyimpanan data global.
3. Pilih Ubah dan pilih di antara opsi berikut:

   - Ubah deskripsi – Memperbarui deskripsi penyimpanan data global
   - Ubah versi mesin – Hanya tersedia mesin Redis versi 5.0.6 atau yang lebih baru.
   - Ubah jenis simpul – Menskalakan klaster wilayah baik secara vertikal (menaikkan dan menurunkan) dan secara horizontal (menskalakan ke dalam dan ke luar). Opsi mencakup keluarga simpul R5 dan M5. Untuk informasi selengkapnya tentang jenis simpul, lihat Tipe simpul yang didukung.
   - Ubah Failover Otomatis – Aktifkan atau nonaktifkan Failover Otomatis. Saat Anda mengaktifkan failover dan node primer di kluster regional ditutup secara tak terduga, ElastiCache gagal ke salah satu replika regional. Untuk informasi selengkapnya, lihat Failover otomatis.

   Untuk kluster Redis dengan mode klaster diaktifkan:

   - Tambahkan serpihan – Masukkan jumlah serpihan untuk ditambahkan dan secara opsional tentukan satu atau beberapa Zona Ketersediaan.
   - Hapus pecahan — Pilih pecahan yang akan dihapus di setiap AWS Wilayah.
   - Seimbangkan ulang serpihan – Menyeimbangkan ulang distribusi slot untuk memastikan distribusi yang merata di seluruh serpihan yang ada dalam klaster.

Untuk memodifikasi parameter datastore global, modifikasi grup parameter dari setiap cluster anggota untuk datastore global. ElastiCache menerapkan perubahan ini ke semua cluster dalam

datastore global itu secara otomatis. Untuk memodifikasi grup parameter cluster tersebut, gunakan konsol Redis atau operasi ModifyCacheClusterAPI. Untuk informasi selengkapnya, lihat Mengubah grup parameter. Saat Anda mengubah grup parameter dari klaster yang berada dalam penyimpanan data global, maka itu akan diterapkan ke semua klaster di dalam penyimpanan data global.

Untuk mengatur ulang seluruh grup parameter atau parameter tertentu, gunakan operasi ResetCacheParameterGroupAPI.

Mempromosikan klaster sekunder menjadi primer

Jika klaster utama atau AWS Wilayah menjadi tidak tersedia atau mengalami masalah kinerja, Anda dapat mempromosikan klaster sekunder ke primer. Promosi boleh dilakukan setiap saat, bahkan saat perubahan lain sedang berlangsung. Anda juga dapat menjalankan beberapa promosi secara paralel dan penyimpanan data global tetap akan menyelesaikannya ke satu primer. Jika Anda mempromosikan beberapa cluster sekunder secara bersamaan, ElastiCache untuk Redis tidak menjamin mana yang akhirnya diselesaikan menjadi primer.

Untuk mempromosikan klaster sekunder menjadi primer

1. Masuk ke AWS Management Console dan buka ElastiCache konsol di https://console.aws.amazon.com/elasticache/.
2. Pada panel navigasi, pilih Penyimpanan Data Global di bawah Redis.
3. Pilih nama penyimpanan data global untuk melihat detailnya.
4. Pilih klaster Sekunder.
5. Pilih Promosikan menjadi primer.

   Anda kemudian diminta untuk mengonfirmasi keputusan Anda dengan peringatan berikut ini:

   ```
   Promoting a region to primary will make the cluster in this region as
   read/writable. Are you sure you want to promote the secondary cluster
   to primary?

   The current primary cluster in primary region will become secondary
   and will stop accepting writes after this operation completes. Please
   ensure you update your application stack to direct traffic to the new
   primary region.
   ```

6. Pilih Konfirmasi jika Anda ingin melanjutkan promosi atau Batalkan jika tidak.

Jika Anda memilih untuk mengonfirmasi, penyimpanan data global Anda akan berubah status menjadi Mengubah  dan tidak tersedia sampai promosi selesai.

Menghapus Wilayah dari penyimpanan data global

Anda dapat menghapus AWS Region dari datastore global dengan menggunakan prosedur berikut.

Untuk menghapus AWS Region dari datastore global

1. Masuk ke AWS Management Console dan buka ElastiCache konsol di https://console.aws.amazon.com/elasticache/.
2. Pada panel navigasi, pilih Penyimpanan Data Global di bawah Redis.
3. Pilih penyimpanan data global.
4. Pilih Wilayah yang ingin dihapus.
5. Pilih Hapus wilayah.

   > **Note**
   >
   > Opsi ini hanya tersedia untuk klaster sekunder.

   Kemudian Anda akan diminta untuk mengonfirmasi keputusan Anda dengan peringatan berikut ini: `Removing the region will remove your only available cross region replica for the primary cluster. Your primary cluster will no longer be set up for disaster recovery and improved read latency in remote region. Are you sure you want to remove the selected region from the global datastore?`
6. Pilih Konfirmasi jika Anda ingin melanjutkan promosi atau Batalkan jika tidak.

Jika Anda memilih konfirmasi, AWS Wilayah akan dihapus dan klaster sekunder tidak lagi menerima pembaruan replikasi.

Menghapus penyimpanan data global

Untuk menghapus penyimpanan data global, semua klaster sekunder harus dihapus terlebih dahulu. Untuk informasi selengkapnya, lihat Menghapus Wilayah dari penyimpanan data global. Melakukan hal ini membuat penyimpanan data global berstatus primer-saja.

Untuk menghapus penyimpanan data global

1. Masuk ke AWS Management Console dan buka ElastiCache konsol di https://console.aws.amazon.com/elasticache/.
2. Pada panel navigasi, pilih Penyimpanan Data Global di bawah Redis.
3. Di bawah Nama Penyimpanan Data Global pilih penyimpanan data global yang ingin dihapus, lalu pilih Hapus.

   Kemudian Anda akan diminta untuk mengonfirmasi keputusan Anda dengan peringatan berikut ini: `Are you sure you want to delete this Global Datastore?`
4. Pilih Hapus.

Penyimpanan data global berubah status menjadi Menghapus.

## Menggunakan penyimpanan data global (CLI)

Anda dapat menggunakan AWS Command Line Interface (AWS CLI) untuk mengontrol beberapa layanan AWS dari baris perintah dan mengotomatiskan layanan tersebut melalui skrip. Anda dapat menggunakan AWS CLI untuk operasi ad hoc (satu kali).

Mengunduh dan mengonfigurasi AWS CLI

AWS CLI berjalan di Windows, macOS, atau Linux. Gunakan prosedur berikut untuk mengunduh dan mengonfigurasinya.

Mengunduh, menginstal, dan mengonfigurasi CLI

1. Unduh AWS CLI pada halaman web AWS Command Line Interface.
2. Ikuti petunjuk untuk Menginstal AWS CLI dan Mengonfigurasi AWS CLI pada Panduan Pengguna AWS Command Line Interface.

Menggunakan AWS CLI dengan penyimpanan data global

Gunakan operasi CLI berikut untuk bekerja dengan penyimpanan data global:

- create-global-replication-group

```
aws elasticache create-global-replication-group \
   --global-replication-group-id-suffix my global datastore  \
```

```
  --primary-replication-group-id sample-repl-group  \
  --global-replication-group-description an optional description of the global
datastore
```

Amazon ElastiCache secara otomatis menerapkan prefiks pada ID penyimpanan data global setelah pembuatan selesai. Setiap Wilayah AWS memiliki prefiks tersendiri. Misalnya, ID penyimpanan data global yang dibuat di Wilayah AS Barat (California Utara) dimulai dengan “virxk” bersama dengan sufiks yang Anda berikan. Sufiks, dikombinasikan dengan prefiks yang dihasilkan otomatis, menjamin keunikan nama penyimpanan data global di beberapa Wilayah.

Tabel berikut mencantumkan setiap prefiks ID Wilayah AWS dan penyimpanan data globalnya.

| Nama Wilayah/Wilayah | prefix |
|---|---|
| Wilayah AS Timur (Ohio)<br>`us-east-2` | `fpkhr` |
| Wilayah AS Timur (Virginia Utara)<br>`us-east-1` | `ldgnf` |
| Wilayah AS Barat (California Utara)<br>`us-west-1` | `virxk` |
| Wilayah AS Barat (Oregon)<br>`us-west-2` | `sgaui` |
| Wilayah Kanada (Pusat)<br>`ca-central-1` | `bxodz` |
| Wilayah Asia Pasifik (Mumbai)<br>`ap-south-1` | `erpgt` |

| Nama Wilayah/Wilayah | prefix |
| --- | --- |
| Wilayah Asia Pasifik (Tokyo)<br>`ap-northeast-1` | `quwsw` |
| Wilayah Asia Pasifik (Seoul)<br>`ap-northeast-2` | `lfqnh` |
| Wilayah Asia Pasifik (Osaka)<br>`ap-northeast-3` | `nlapn` |
| Wilayah Asia Pasifik (Singapura)<br>`ap-southeast-1` | `vlqxn` |
| Wilayah Asia Pasifik (Sydney)<br>`ap-southeast-2` | `vbgxd` |
| Wilayah Eropa (Frankfurt)<br>`eu-central-1` | `iudkw` |
| Wilayah Eropa (Irlandia)<br>`eu-west-1` | `gxeiz` |
| Wilayah Eropa (London)<br>`eu-west-2` | `okuqm` |
| Wilayah Eropa (Paris)<br>`eu-west-3` | `fgjhi` |
| Wilayah Amerika Selatan (Sao Paulo)<br>`sa-east-1` | `juxlw` |

| Nama Wilayah/Wilayah | prefix |
| --- | --- |
| Wilayah Tiongkok (Beijing)<br>`cn-north-1` | `emvgo` |
| Wilayah Tiongkok (Ningxia)<br>`cn-northwest-1` | `ckbem` |
| Wilayah Asia Pasifik (Hong Kong)<br>`ap-east-1` | `knjmp` |
| AWS GovCloud (US-West)<br>`us-gov-west-1` | `sgwui` |

- create-replication-group – Gunakan operasi ini untuk membuat klaster sekunder untuk penyimpanan data global dengan menyediakan nama penyimpanan data global untuk parameter `--global-replication-group-id`.

```
aws elasticache create-replication-group \
  --replication-group-id secondary replication group name \
  --replication-group-description “Replication group description" \
  --global-replication-group-id global datastore name
```

Saat memanggil operasi ini dan meneruskan nilai `--global-replication-group-id`, ElastiCache for Redis akan mengambil nilai dari grup replikasi primer dari grup replikasi global untuk parameter berikut. Jangan memasukkan nilai untuk parameter ini:

`"PrimaryClusterId",`

`"AutomaticFailoverEnabled",`

`"NumNodeGroups",`

`"CacheParameterGroupName",`

`"CacheNodeType",`

```
  "Engine",

  "EngineVersion",

  "CacheSecurityGroupNames",

  "EnableTransitEncryption",

  "AtRestEncryptionEnabled",

  "SnapshotArns",

  "SnapshotName"
```

- describe-global-replication-groups

```
aws elasticache describe-global-replication-groups \
   --global-replication-group-id my global datastore  \
   --show-member-info an optional parameter that returns a list of the primary and
 secondary clusters that make up the global datastore
```

- modify-global-replication-group

```
aws elasticache modify-global-replication-group \
   --global-replication-group-id my global datastore  \
   --automatic-failover-enabled \
   --cache-node-type node type \
   --cache-parameter-group-name parameter group name \
   --engine-version engine version \
   --–apply-immediately \
   --global-replication-group-description description
```

- delete-global-replication-group

```
aws elasticache delete-global-replication-group \
   --global-replication-group-id my global datastore  \
   --retain-primary-replication-group defaults to true
```

- disassociate-global-replication-group

```
aws elasticache disassociate-global-replication-group \
   --global-replication-group-id my global datastore  \
```

```
   --replication-group-id my secondary cluster  \
   --replication-group-region the AWS Region in which the secondary cluster resides
```

- failover-global-replication-group

```
aws elasticache failover-replication-group \
   --global-replication-group-id my global datastore \
   --primary-region The AWS Region of the primary cluster \
   --primary-replication-group-id  The name of the global datastore, including the
 suffix.
```

- increase-node-groups-in-global-replication-group

```
aws elasticache increase-node-groups-in-global-replication-group \
   --apply-immediately yes \
   --global-replication-group-id global-replication-group-name \
   --node-group-count 3
```

- decrease-node-groups-in-global-replication-group

```
aws elasticache decrease-node-groups-in-global-replication-group \
   --apply-immediately yes \
   --global-replication-group-id global-replication-group-name \
   --node-group-count 3
```

- rebalance-shards-in-global-replication-group

```
aws elasticache rebalance-shards-in-global-replication-group \
   --apply-immediately yes \
   --global-replication-group-id global-replication-group-name
```

Gunakan bantuan untuk mengeluarkan daftar semua perintah ElastiCache for Redis yang tersedia.

```
aws elasticache help
```

Anda juga dapat menggunakan bantuan untuk mengetahui penjelasan perintah tertentu dan mempelajari lebih lanjut perihal penggunaannya:

```
aws elasticache create-global-replication-group help
```

# Ketersediaan tinggi menggunakan grup replikasi

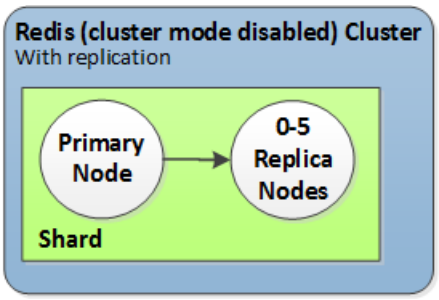
Klaster Redis Amazon ElastiCache simpul-tunggal adalah entitas dalam memori dengan layanan perlindungan data yang terbatas (AOF). Jika klaster Anda gagal untuk alasan apapun, Anda akan kehilangan semua data dari klaster. Namun, jika Anda menjalankan mesin Redis, Anda dapat mengelompokkan 2 hingga 6 simpul ke dalam satu klaster dengan beberapa replika di mana 1 hingga 5 simpul baca-saja berisi data replikasi dari simpul primer baca/tulis tunggal dari grup tersebut. Dalam skenario ini, jika salah satu simpul gagal untuk alasan apapun, Anda tidak kehilangan semua data Anda karena data direplikasikan pada satu atau lebih simpul lainnya. Karena latensi replikasi, beberapa data mungkin hilang jika yang mengalami kegagalan adalah simpul baca/tulis primer.

Seperti yang terlihat pada grafik berikut, struktur replikasi terkandung di dalam serpihan (disebut grup simpul dalam API/CLI) yang terkandung dalam klaster Redis. Klaster Redis (mode klaster dinonaktifkan) selalu memiliki satu serpihan. Klaster Redis (mode klaster diaktifkan) dapat memiliki hingga 500 serpihan dengan data klaster dipartisi di serpihan. Anda dapat membuat klaster dengan jumlah serpihan lebih banyak dan jumlah replika lebih sedikit dengan jumlah hingga 90 simpul per klaster. Konfigurasi klaster ini dapat berkisar dari 90 serpihan dan 0 replika hingga 15 serpihan dan 5 replika, yang merupakan jumlah replika maksimum yang diperbolehkan.

Batas simpul atau serpihan dapat ditingkatkan hingga maksimum 500 per klaster jika versi mesin Redis adalah 5.0.6 atau lebih tinggi. Sebagai contoh, Anda dapat memilih untuk membuat konfigurasi dari sebuah klaster dengan 500 simpul yang berkisar antara 83 serpihan (satu primer dan 5 replika per serpihan) dan 500 serpihan (primer tunggal dan tidak ada replika). Pastikan ada cukup alamat IP yang tersedia untuk mengakomodasi peningkatan. Perangkap umum termasuk subnet dalam grup subnet memiliki rentang CIDR yang terlalu kecil atau subnet digunakan bersama dan banyak digunakan oleh klaster lain. Untuk informasi selengkapnya, lihat Membuat grup subnet.

Untuk versi sebelum 5.0.6, batasnya adalah 250 per klaster.

Untuk meminta penambahan batas, lihat Kuota Layanan AWS dan pilih jenis batas Simpul per klaster per jenis instans.



Klaster Redis (mode klaster dinonaktifkan) dengan satu serpihan dan 0 hingga 5 simpul replika

Jika klaster dengan replika mengaktifkan Multi-AZ dan simpul primer gagal, maka primer akan failover ke replika baca. Karena data diperbarui pada simpul replika secara asinkron, mungkin terjadi sedikit kehilangan data karena latensi dalam memperbarui simpul replika. Untuk informasi selengkapnya, lihat Mitigasi Kegagalan saat Menjalankan Redis.

Topik

- Memahami replikasi Redis
- Replikasi: Redis (Mode Klaster Dinonaktifkan) vs Redis (Mode Klaster Diaktifkan)
- Meminimalkan waktu henti di ElastiCache for Redis dengan Multi-AZ
- Cara penerapan sinkronisasi dan pencadangan
- Membuat grup replikasi Redis
- Melihat detail grup replikasi
- Menemukan titik akhir grup replikasi
- Mengubah grup replikasi
- Menghapus grup replikasi
- Mengubah jumlah replika
- Menaikkan replika baca menjadi primer, untuk grup replikasi Redis (mode klaster dinonaktifkan)

## Memahami replikasi Redis

Redis menerapkan replikasi dalam dua cara:

- Dengan serpihan tunggal yang berisi semua data klaster di setiap simpul—Redis (mode klaster dinonaktifkan)
- Dengan data yang dipartisi ke hingga 500 serpihan—Redis (mode klaster diaktifkan)

Setiap serpihan dalam grup replikasi memiliki simpul primer baca/tulis tunggal dan hingga 5 simpul replika baca-saja. Anda dapat membuat klaster dengan jumlah serpihan lebih banyak dan jumlah replika lebih sedikit dengan jumlah hingga 90 simpul per klaster. Konfigurasi klaster ini dapat berkisar dari 90 serpihan dan 0 replika hingga 15 serpihan dan 5 replika, yang merupakan jumlah replika maksimum yang diperbolehkan.

Batas simpul atau serpihan dapat ditingkatkan hingga maksimum 500 per klaster jika versi mesin Redis adalah 5.0.6 atau lebih tinggi. Sebagai contoh, Anda dapat memilih untuk membuat konfigurasi dari sebuah klaster dengan 500 simpul yang berkisar antara 83 serpihan (satu primer dan 5 replika per serpihan) dan 500 serpihan (primer tunggal dan tidak ada replika). Pastikan ada cukup alamat IP yang tersedia untuk mengakomodasi peningkatan. Perangkap umum termasuk subnet dalam grup subnet memiliki rentang CIDR yang terlalu kecil atau subnet digunakan bersama dan banyak digunakan oleh klaster lain. Untuk informasi selengkapnya, lihat Membuat grup subnet.

Untuk versi sebelum 5.0.6, batasnya adalah 250 per klaster.

Untuk meminta penambahan batas, lihat Kuota Layanan AWS dan pilih jenis batas Simpul per klaster per jenis instans.

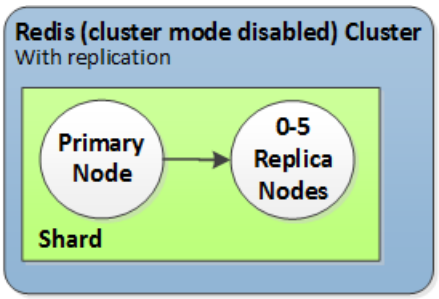
Topik

- Redis (Mode Klaster Dinonaktifkan)
- Redis (mode klaster diaktifkan)

### Redis (Mode Klaster Dinonaktifkan)

Klaster Redis (mode klaster dinonaktifkan) memiliki serpihan tunggal, yang di dalamnya terdapat kumpulan simpul Redis; satu simpul primer baca/tulis dan hingga lima simpul replika sekunder baca-saja. Setiap replika baca memelihara salinan data dari simpul primer klaster. Mekanisme replikasi asinkron digunakan untuk menjaga sinkronisasi replika baca dengan primer. Aplikasi dapat membaca

dari simpul apa pun di dalam klaster. Aplikasi hanya dapat menulis ke simpul primer. Replika baca meningkatkan throughput baca dan menjaga kehilangan data dalam kasus kegagalan simpul.




Klaster Redis (mode klaster dinonaktifkan) dengan serpihan dan simpul replika tunggal

Anda dapat menggunakan klaster Redis (mode klaster dinonaktifkan) dengan simpul replika untuk menskalakan solusi Redis Anda untuk ElastiCache guna menangani aplikasi yang bersifat intensif membaca atau untuk mendukung sejumlah besar klien yang membaca dari klaster yang sama secara bersamaan.

Semua simpul dalam klaster Redis (mode klaster dinonaktifkan) harus berada di wilayah yang sama.

Saat Anda menambahkan replika baca ke klaster, semua data dari primer akan disalin ke simpul baru. Sejak saat itu, setiap kali data ditulis ke primer, perubahan akan disebarkan secara asinkron ke semua replika baca.

Untuk meningkatkan toleransi kesalahan dan mengurangi waktu henti proses tulis, aktifkan Multi-AZ dengan Failover Otomatis untuk klaster Redis (mode klaster dinonaktifkan) dengan replika Anda. Untuk informasi selengkapnya, lihat Meminimalkan waktu henti di ElastiCache for Redis dengan Multi-AZ.

Anda dapat mengubah peran simpul di klaster Redis (mode klaster dinonaktifkan), dengan primer dan salah satu replika saling bertukar peran. Anda mungkin memutuskan untuk melakukan ini untuk alasan penyetelan kinerja. Misalnya, dengan aplikasi web yang memiliki aktivitas menulis yang berat, Anda dapat memilih simpul dengan latensi jaringan terendah. Untuk informasi selengkapnya, lihat Menaikkan replika baca menjadi primer, untuk grup replikasi Redis (mode klaster dinonaktifkan).

Redis (mode klaster diaktifkan)

Klaster Redis (modus klaster diaktifkan) terdiri dari 1 sampai 500 serpihan (API/CLI: grup simpul). Setiap serpihan memiliki simpul primer dan hingga lima simpul replika baca saja. Konfigurasi klaster ini dapat berkisar dari 90 serpihan dan 0 replika hingga 15 serpihan dan 5 replika, yang merupakan jumlah replika maksimum yang diperbolehkan.

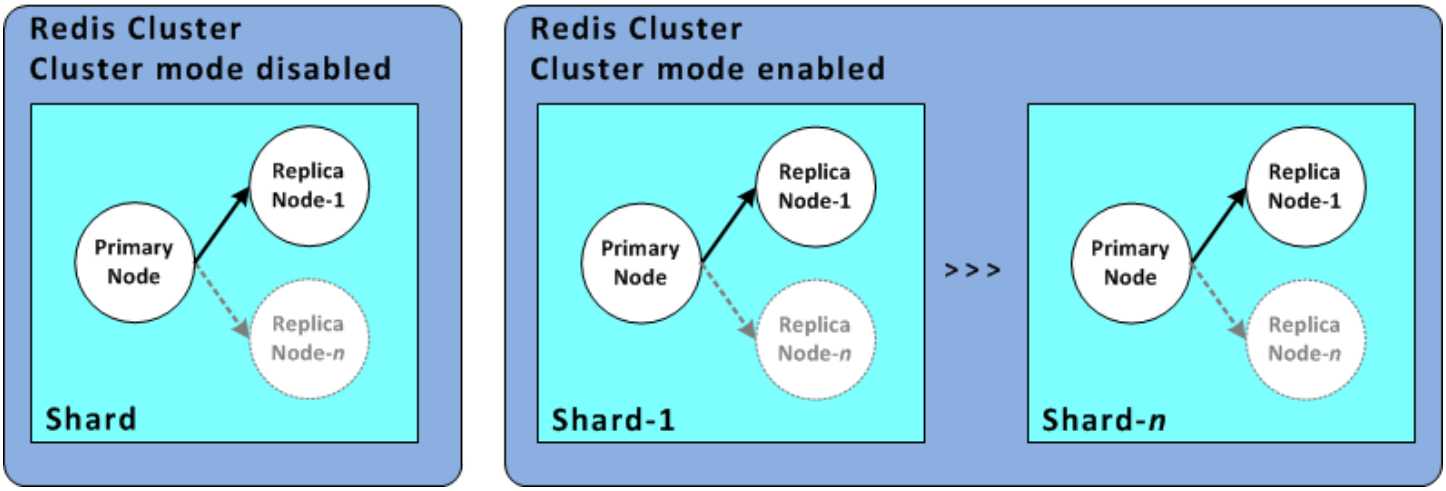
Batas simpul atau serpihan dapat ditingkatkan hingga maksimum 500 per klaster jika versi mesin Redis adalah 5.0.6 atau lebih tinggi. Sebagai contoh, Anda dapat memilih untuk membuat konfigurasi dari sebuah klaster dengan 500 simpul yang berkisar antara 83 serpihan (satu primer dan 5 replika per serpihan) dan 500 serpihan (primer tunggal dan tidak ada replika). Pastikan ada cukup alamat IP yang tersedia untuk mengakomodasi peningkatan. Perangkap umum termasuk subnet dalam grup subnet memiliki rentang CIDR yang terlalu kecil atau subnet digunakan bersama dan banyak digunakan oleh klaster lain. Untuk informasi selengkapnya, lihat Membuat grup subnet.

Untuk versi sebelum 5.0.6, batasnya adalah 250 per klaster.

Untuk meminta penambahan batas, lihat Kuota Layanan AWS dan pilih jenis batas Simpul per klaster per jenis instans.

Setiap replika baca di dalam serpihan memelihara salinan data dari primer pada serpihan ini. Mekanisme replikasi asinkron digunakan untuk menjaga sinkronisasi replika baca dengan primer. Aplikasi dapat membaca dari simpul apa pun di dalam klaster. Aplikasi hanya dapat menulis ke simpul primer. Replika baca meningkatkan skalabilitas baca dan menjaga kehilangan data. Data dipartisi di seluruh serpihan di dalam klaster Redis (mode klaster diaktifkan).

Aplikasi menggunakan titik akhir konfigurasi dari klaster Redis (mode klaster diaktifkan) untuk menyambung dengan simpul di dalam klaster. Untuk informasi selengkapnya, lihat Menemukan titik akhir koneksi.




Klaster Redis (mode klaster diaktifkan) dengan beberapa serpihan dan simpul replika

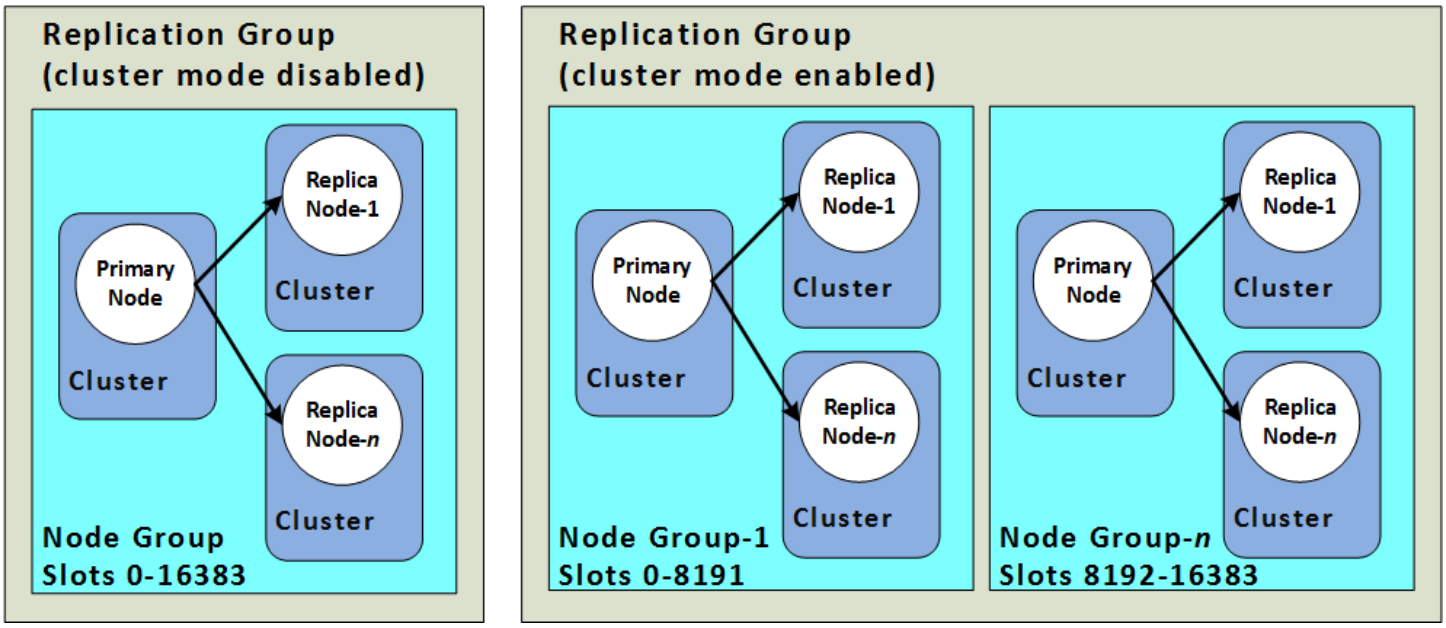
Semua simpul dalam klaster Redis (mode klaster dinonaktifkan) harus berada di wilayah yang sama. Untuk meningkatkan toleransi kesalahan, Anda dapat menyediakan baik replika primer dan replika baca pada beberapa Zona Ketersediaan di dalam wilayah tersebut.

Saat ini, di Redis (mode klaster diaktifkan), ada beberapa keterbatasan.

- Anda tidak dapat menaikkan salah satu simpul replika menjadi primer secara manual.

## Replikasi: Redis (Mode Klaster Dinonaktifkan) vs Redis (Mode Klaster Diaktifkan)

Dimulai dengan Redis versi 3.2, Anda memiliki kemampuan untuk membuat salah satu dari dua jenis klaster Redis yang berbeda (API/CLI: grup replikasi). Klaster Redis (mode klaster dinonaktifkan) selalu memiliki serpihan tunggal (API/CLI: grup simpul) dengan hingga 5 simpul replika baca. Klaster Redis (mode klaster diaktifkan) memiliki hingga 500 serpihan dengan 1 hingga 5 simpul replika baca di dalam setiap serpihan.




Klaster Redis (mode klaster dinonaktifkan) dan klaster Redis (mode klaster diaktifkan)

Tabel berikut merangkum perbedaan penting antara klaster Redis (mode klaster dinonaktifkan) dan klaster Redis (mode klaster diaktifkan).

Membandingkan Klaster Redis (Mode Klaster Dinonaktifkan) dan Klaster Redis (Mode Klaster Diaktifkan)

| Fitur | Redis (mode klaster dinonaktifkan) | Redis (mode klaster diaktifkan) |
|---|---|---|
| Dapat diubah | Ya. Mendukung penambahan dan penghapusan simpul | Terbatas. Untuk informasi selengkapnya, lihat Versi |

| Fitur | Redis (mode klaster dinonaktifkan) | Redis (mode klaster diaktifkan) |
| --- | --- | --- |
| | replika, dan menaikkan skala jenis simpul. | mesin dan pemutakhiran dan Penskalaan klaster di Redis (Mode Klaster Diaktifkan). |
| Pembuatan Partisi Data | Tidak | Ya |
| Serpihan | 1 | 1 hingga 500 |
| Replika baca | 0 hingga 5<br><br>**Important**<br>Jika tidak ada replika dan simpul gagal, Anda akan mengalami kehilangan data total. | 0 hingga 5 per serpihan.<br><br>**Important**<br>Jika tidak ada replika dan salah satu simpul gagal, Anda akan mengalami kehilangan data total di serpihan itu. |
| Multi-AZ | Ya, dengan setidaknya 1 replika.<br><br>Opsional. Aktif secara default. | Ya<br><br>Opsional. Aktif secara default. |
| Snapshot (Cadangan) | Ya, membuat file .rdb tunggal. | Ya, membuat file .rdb yang unik untuk setiap serpihan. |
| Memulihkan | Ya, menggunakan file .rdb tunggal dari klaster Redis (mode klaster dinonaktifkan). | Ya, menggunakan file .rdb baik dari klaster Redis (mode klaster dinonaktifkan) atau dari klaster Redis (mode klaster diaktifkan). |
| Didukung pada | Semua versi Redis | Redis 3.2 dan berikutnya |

| Fitur | Redis (mode klaster dinonaktifkan) | Redis (mode klaster diaktifkan) |
|---|---|---|
| Mesin dapat ditingkatkan | Ya, dengan beberapa batasan. Untuk informasi selengkapnya, lihat Versi mesin dan pemutakhiran . | Ya, dengan beberapa batasan. Untuk informasi selengkapnya, lihat Versi mesin dan pemutakhiran . |
| Enkripsi | Versi 3.2.6 (dijadwalkan untuk EOL, lihat Redis versi akhir dari jadwal hidup) dan 4.0.10 atau yang lebih baru. | Versi 3.2.6 (dijadwalkan untuk EOL, lihat Redis versi akhir dari jadwal hidup) dan 4.0.10 atau yang lebih baru. |
| Layak untuk HIPAA | Versi 3.2.6 (dijadwalkan untuk EOL, lihat Redis versi akhir dari jadwal hidup) dan 4.0.10 atau yang lebih baru. | Versi 3.2.6 (dijadwalkan untuk EOL, lihat Redis versi akhir dari jadwal hidup) dan 4.0.10 atau yang lebih baru. |
| Kepatuhan PCI DSS | Versi 3.2.6 (dijadwalkan untuk EOL, lihat Redis versi akhir dari jadwal hidup) dan 4.0.10 atau yang lebih baru. | Versi 3.2.6 (dijadwalkan untuk EOL, lihat Redis versi akhir dari jadwal hidup) dan 4.0.10 atau yang lebih baru. |
| Resharding online | T/A | Versi 3.2.10 (dijadwalkan untuk EOL, lihat Redis versi akhir dari jadwal hidup) danyang lebih baru. |

Sebaiknya pilih yang mana?

Ketika memilih antara Redis (mode klaster dinonaktifkan) atau Redis (mode klaster diaktifkan), pertimbangkan faktor berikut:

- Penskalaan v. pembuatan partisi – Bisnis membutuhkan perubahan. Anda perlu menyediakan baik untuk permintaan puncak ataupun menskalakan mengikuti perubahan permintaan. Redis (mode klaster dinonaktifkan) mendukung penskalaan. Anda dapat menskalakan kapasitas baca dengan menambahkan atau menghapus simpul replika, atau Anda dapat menskalakan kapasitas dengan menaikkan skala ke jenis simpul yang lebih besar. Kedua operasi ini membutuhkan waktu. Untuk

informasi selengkapnya, lihat Penskalaan Redis (Mode Cluster Dinonaktifkan) cluster dengan simpul replika.

Klaster Redis (mode klaster diaktifkan) mendukung pembuatan partisi data Anda secara menyeluruh hingga 500 grup simpul. Anda dapat mengubah jumlah serpihan secara dinamis mengikuti perubahan kebutuhan bisnis Anda. Salah satu keuntungan dari pembuatan partisi adalah bahwa Anda menyebarkan beban Anda terhadap lebih banyak titik akhir, yang mengurangi kemacetan akses selama permintaan puncak. Selain itu, Anda dapat mengakomodasi kumpulan data yang lebih besar karena data dapat disebarkan ke beberapa server. Untuk informasi tentang penskalaan partisi Anda, lihat Penskalaan klaster di Redis (Mode Klaster Diaktifkan).

- Ukuran simpul vs. jumlah simpul – Karena klaster Redis (mode klaster dinonaktifkan) hanya memiliki satu serpihan, jenis simpul harus cukup besar untuk mengakomodasi semua data klaster ditambah overhead yang diperlukan. Di sisi lain, karena Anda dapat membuat partisi data Anda tersebar di beberapa serpihan jika menggunakan klaster Redis (mode klaster diaktifkan), jenis simpul dapat lebih kecil, meskipun Anda akan membutuhkan lebih banyak simpul. Untuk informasi selengkapnya, lihat Memilih ukuran simpul Anda.

- Membaca v. menulis – Jika beban primer pada klaster Anda adalah aplikasi yang membaca data, Anda dapat menskalakan klaster Redis (mode klaster dinonaktifkan) dengan menambahkan dan menghapus replika baca. Meski demikian, terdapat jumlah maksimum sebanyak 5 replika baca. Jika beban pada klaster Anda adalah lebih berat ke proses tulis, Anda dapat mendapatkan manfaat dari tambahan titik akhir tulis dari klaster Redis (mode klaster diaktifkan) dengan beberapa serpihan.

Apapun jenis klaster yang Anda pilih untuk diterapkan, pastikan untuk memilih jenis simpul yang memadai untuk kebutuhan Anda saat ini dan di masa depan.

## Meminimalkan waktu henti di ElastiCache for Redis dengan Multi-AZ

Ada sejumlah kejadian ketika ElastiCache for Redis mungkin perlu mengganti simpul primer; hal ini termasuk jenis pemeliharaan terencana tertentu dan peristiwa kegagalan simpul primer atau Availability Zone yang dapat terjadi.

Penggantian ini mengakibatkan waktu henti pada klaster, tetapi jika Multi-AZ diaktifkan, waktu henti dapat dikurangi. Peran simpul primer akan secara otomatis melakukan fail over ke salah satu replika baca. Anda tidak perlu membuat dan menyediakan simpul primer baru, karena ElastiCache akan mengatasi hal ini secara transparan. Failover dan promosi replika ini memastikan Anda dapat melanjutkan penulisan ke primer baru segera setelah promosi selesai.

ElastiCache juga menyebarkan nama Domain Name Service (DNS) dari replika yang dipromosikan. Hal ini dilakukan karena jika aplikasi Anda menulis ke titik akhir primer, tidak perlu ada perubahan titik akhir dalam aplikasi Anda. Jika Anda membaca dari setiap titik akhir, pastikan mengubah titik akhir baca dari replika yang dipromosikan menjadi primer ke titik akhir dari replika yang baru.

Dalam kasus penggantian simpul terencana yang dimulai karena pembaruan pemeliharaan atau pembaruan secara mandiri, perlu diperhatikan hal berikut:

- Untuk Klaster ElastiCache for Redis, penggantian simpul terencana selesai sementara klaster melayani permintaan tulis yang masuk.
- Untuk klaster dengan mode Klaster Redis dinonaktifkan dan Multi-AZ diaktifkan yang berjalan pada mesin 5.0.6 atau yang lebih baru, penggantian simpul terencana selesai sementara klaster melayani permintaan tulis yang masuk.
- Untuk klaster dengan mode Klaster Redis dinonaktifkan dan Multi-AZ diaktifkan yang berjalan pada mesin 4.0.10 atau sebelumnya, Anda mungkin mengalami gangguan tulis singkat yang berkaitan dengan pembaruan DNS. Gangguan ini mungkin memakan waktu hingga beberapa detik. Proses ini jauh lebih cepat daripada membuat dan menetapkan primer baru, yang akan terjadi jika Anda tidak mengaktifkan Multi-AZ.

Anda dapat mengaktifkan Multi-AZ menggunakan Konsol Manajemen ElastiCache, AWS CLI, atau API ElastiCache.

Mengaktifkan Multi-AZ ElastiCache pada klaster Redis Anda (dalam API dan CLI, grup replikasi) meningkatkan toleransi kesalahan Anda. Hal ini benar terutama dalam kasus ketika klaster primer baca/tulis dari klaster Anda menjadi tidak terjangkau atau gagal karena alasan apa pun. Multi-AZ hanya didukung pada klaster Redis yang memiliki lebih dari satu simpul di setiap serpihan.

Topik

- Mengaktifkan Multi-AZ
- Skenario kegagalan dengan respons Multi-AZ
- Menguji failover otomatis
- Keterbatasan Multi-AZ Redis

## Mengaktifkan Multi-AZ

Anda dapat mengaktifkan Multi-AZ ketika Anda membuat atau memodifikasi klaster (API atau CLI, replikasi grup) menggunakan konsol ElastiCache, AWS CLI, atau API ElastiCache.

Anda dapat mengaktifkan Multi-AZ hanya pada klaster Redis (mode klaster dinonaktifkan) yang memiliki setidaknya satu replika baca yang tersedia. Klaster tanpa replika baca tidak menyediakan ketersediaan tinggi atau toleransi kesalahan. Untuk informasi tentang cara membuat klaster dengan replikasi, lihat Membuat grup replikasi Redis. Untuk informasi tentang cara menambahkan replika baca ke klaster dengan replikasi, lihat Menambahkan replika baca, untuk grup replikasi Redis (Mode Klaster Dinonaktifkan).

Topik

- Mengaktifkan Multi-AZ (Konsol)
- Mengaktifkan Multi-AZ (AWS CLI)
- Mengaktifkan Multi-AZ (API ElastiCache)

### Mengaktifkan Multi-AZ (Konsol)

Anda dapat mengaktifkan Multi-AZ menggunakan konsol ElastiCache saat Anda membuat klaster Redis yang baru atau dengan memodifikasi klaster Redis yang ada dengan replikasi.

Multi-AZ diaktifkan secara default pada klaster Redis (mode klaster diaktifkan).

> **Important**
>
> ElastiCache akan secara otomatis mengaktifkan Multi-AZ hanya jika klaster berisi setidaknya satu replika pada Availability Zone yang berbeda dari primer di semua serpihan.

Mengaktifkan Multi-AZ saat membuat klaster menggunakan konsol ElastiCache

Lihat informasi selengkapnya tentang prosedur ini, lihat Membuat klaster Redis (Mode Klaster Dinonaktifkan) (Konsol). Pastikan untuk memiliki satu atau beberapa replika dan mengaktifkan Multi-AZ.

Mengaktifkan Multi-AZ pada klaster yang telah ada (Konsol)

Untuk informasi selengkapnya tentang proses ini, lihat Memodifikasi Klaster Menggunakan AWS Management Console.

Mengaktifkan Multi-AZ (AWS CLI)

Contoh kode berikut menggunakan AWS CLI untuk mengaktifkan Multi-AZ untuk grup replikasi `redis12`.

> **Important**
>
> Replikasi grup `redis12` harus sudah ada dan memiliki setidaknya satu replika baca yang tersedia.

Untuk Linux, macOS, atau Unix:

```
aws elasticache modify-replication-group \
    --replication-group-id redis12 \
    --automatic-failover-enabled \
    --multi-az-enabled \
    --apply-immediately
```

Untuk Windows:

```
aws elasticache modify-replication-group ^
    --replication-group-id redis12 ^
    --automatic-failover-enabled ^
    --multi-az-enabled ^
    --apply-immediately
```

Output JSON dari perintah ini akan terlihat seperti berikut.

```
{
    "ReplicationGroup": {
```

```
        "Status": "modifying",
        "Description": "One shard, two nodes",
        "NodeGroups": [
            {
                "Status": "modifying",
                "NodeGroupMembers": [
                    {
                        "CurrentRole": "primary",
                        "PreferredAvailabilityZone": "us-west-2b",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Port": 6379,
                            "Address":
"redis12-001.v5r9dc.0001.usw2.cache.amazonaws.com"
                        },
                        "CacheClusterId": "redis12-001"
                    },
                    {
                        "CurrentRole": "replica",
                        "PreferredAvailabilityZone": "us-west-2a",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Port": 6379,
                            "Address":
"redis12-002.v5r9dc.0001.usw2.cache.amazonaws.com"
                        },
                        "CacheClusterId": "redis12-002"
                    }
                ],
                "NodeGroupId": "0001",
                "PrimaryEndpoint": {
                    "Port": 6379,
                    "Address": "redis12.v5r9dc.ng.0001.usw2.cache.amazonaws.com"
                }
            }
        ],
        "ReplicationGroupId": "redis12",
        "SnapshotRetentionLimit": 1,
        "AutomaticFailover": "enabling",
        "MultiAZ": "enabled",
        "SnapshotWindow": "07:00-08:00",
        "SnapshottingClusterId": "redis12-002",
        "MemberClusters": [
            "redis12-001",
```

```
            "redis12-002"
        ],
        "PendingModifiedValues": {}
    }
}
```

Untuk informasi selengkapnya, lihat topik ini di Referensi Perintah AWS CLI:

- create-cache-cluster
- create-replication-group
- modify-replication-group di Referensi Perintah AWS CLI.

### Mengaktifkan Multi-AZ (API ElastiCache)

Contoh kode berikut menggunakan API ElastiCache untuk mengaktifkan Multi-AZ untuk grup replikasi `redis12`.

> **Note**
>
> Untuk menggunakan contoh ini, grup replikasi `redis12` harus sudah ada dan memiliki setidaknya satu replika baca yang tersedia.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=ModifyReplicationGroup
    &ApplyImmediately=true
    &AutoFailover=true
    &MultiAZEnabled=true
    &ReplicationGroupId=redis12
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20140401T192317Z
    &X-Amz-Credential=<credential>
```

Untuk informasi lain, lihat topik ini di Referensi API ElastiCache:

- CreateCacheCluster
- CreateReplicationGroup

- ModifyReplicationGroup

## Skenario kegagalan dengan respons Multi-AZ

Sebelum dikenalkannya Multi-AZ, ElastiCache mendeteksi dan mengganti simpul gagal dari klaster dengan membuat ulang dan menyediakan kembali simpul yang gagal. Jika Anda mengaktifkan Multi-AZ, simpul primer yang gagal akan melakukan fail over ke replika dengan ketertinggalan replikasi terkecil. Replika yang dipilih dipromosikan secara otomatis menjadi primer, hal ini jauh lebih cepat daripada membuat dan menetapkan kembali simpul primer baru. Proses ini biasanya memakan waktu hanya beberapa detik hingga Anda dapat menulis lagi ke klaster.

Saat Multi-AZ diaktifkan, ElastiCache terus memantau status simpul primer. Jika simpul primer gagal, salah satu tindakan berikut akan dilakukan bergantung pada jenis kegagalan.

Topik



### Skenario kegagalan ketika hanya simpul primer yang gagal

Jika hanya simpul primer yang gagal, replika baca dengan ketertinggalan replikasi terkecil akan dipromosikan menjadi primer. Replika baca pengganti kemudian dibuat dan ditetapkan di Zona Ketersediaan yang sama dengan primer yang gagal.

Jika hanya simpul primer yang gagal, Multi-AZ ElastiCache melakukan hal berikut:

1. Simpul primer yang gagal akan dibuat offline.
2. Replika baca dengan ketertinggalan replikasi terkecil akan dipromosikan menjadi primer.

   Proses tulis dapat dilanjutkan segera setelah proses promosi selesai, biasanya hanya beberapa detik. Jika aplikasi Anda menulis ke titik akhir primer, Anda tidak perlu mengubah titik akhir untuk tulis atau baca. ElastiCache menyebarkan nama DNS dari replika yang dipromosikan.
3. Replika baca pengganti diluncurkan dan ditetapkan.

   Replika baca pengganti diluncurkan pada Zona Ketersediaan yang sama dengan simpul primer yang gagal sehingga distribusi simpul tetap terpelihara.
4. Replika melakukan sinkronisasi dengan simpul primer yang baru.

Setelah replika baru tersedia, perhatikan efeknya berikut ini:

- Titik akhir primer – Anda tidak perlu membuat perubahan apa pun pada aplikasi Anda, karena nama DNS dari simpul primer baru disebarkan ke titik akhir primer.
- Titik akhir baca – Titik akhir pembaca diperbarui secara otomatis untuk mengarah ke simpul replika yang baru.

Untuk informasi tentang cara menemukan titik akhir klaster, lihat topik berikut:

- Menemukan Titik Akhir Klaster Redis (Mode Klaster Dinonaktifkan) (Konsol)
- Menemukan Titik Akhir untuk Grup Replikasi (AWS CLI)
- Menemukan Titik Akhir untuk Grup Replikasi (API ElastiCache)

Skenario kegagalan ketika simpul primer dan beberapa replika baca gagal

Jika primer dan setidaknya satu replika baca gagal, replika yang tersedia dengan ketertinggalan replikasi terkecil akan dipromosikan menjadi klaster primer. Replika baca yang baru juga dibuat dan ditetapkan di Zona Ketersediaan yang sama dengan simpul yang gagal dan replika yang dipromosikan menjadi primer.

Jika hanya simpul primer dan beberapa replika baca yang gagal, Multi-AZ ElastiCache melakukan hal berikut:

1. Simpul primer yang gagal dan replika baca yang gagal akan dibuat offline.
2. Replika baca dengan ketertinggalan replikasi terkecil akan dipromosikan menjadi simpul primer.

   Proses tulis dapat dilanjutkan segera setelah proses promosi selesai, biasanya hanya beberapa detik. Jika aplikasi Anda menulis ke titik akhir primer, Anda tidak perlu mengubah titik akhir untuk tulis. ElastiCache menyebarkan nama DNS dari replika yang dipromosikan.
3. Replika pengganti dibuat dan ditetapkan.

   Replika pengganti dibuat di Zona Ketersediaan dari simpul yang gagal sehingga distribusi simpul tetap terpelihara.
4. Semua klaster melakukan sinkronisasi dengan simpul primer baru.

Lakukan perubahan berikut pada aplikasi Anda setelah simpul yang baru tersedia:

- Titik akhir primer – Jangan membuat perubahan apa pun pada aplikasi Anda. Nama DNS dari simpul primer baru disebarkan ke titik akhir primer.
- Titik akhir baca – Titik akhir baca diperbarui secara otomatis untuk mengarah ke simpul replika yang baru.

Untuk informasi tentang menemukan titik akhir grup replikasi, lihat topik berikut:

- Menemukan Titik Akhir Klaster Redis (Mode Klaster Dinonaktifkan) (Konsol)
- Menemukan Titik Akhir untuk Grup Replikasi (AWS CLI)
- Menemukan Titik Akhir untuk Grup Replikasi (API ElastiCache)

Skenario kegagalan ketika seluruh klaster gagal

Jika semuanya gagal, semua simpul dibuat kembali dan ditetapkan pada Zona Ketersediaan yang sama dengan simpul asli.

Dalam skenario ini, semua data dalam klaster akan hilang karena kegagalan dari setiap simpul dalam klaster. Kejadian ini jarang terjadi.

Ketika seluruh klaster gagal, Multi-AZ ElastiCache melakukan hal berikut:

1. Simpul primer dan replika baca yang gagal akan dibuat offline.
2. Simpul primer pengganti dibuat dan ditetapkan.
3. Replika pengganti dibuat dan ditetapkan.

   Penggantinya dibuat di Zona Ketersediaan dari simpul yang gagal sehingga distribusi simpul tetap dipertahankan.

   Karena seluruh klaster gagal, data menjadi hilang dan semua simpul baru mulai dari baru.

Karena setiap simpul pengganti memiliki titik akhir yang sama dengan simpul yang digantikannya, Anda tidak perlu membuat perubahan titik akhir pada aplikasi Anda.

Untuk informasi tentang menemukan titik akhir grup replikasi, lihat topik berikut:

- Menemukan Titik Akhir Klaster Redis (Mode Klaster Dinonaktifkan) (Konsol)

- Menemukan Titik Akhir untuk Grup Replikasi (AWS CLI)
- Menemukan Titik Akhir untuk Grup Replikasi (API ElastiCache)

Sebaiknya buat simpul primer dan replika baca di Zona Ketersediaan yang berbeda untuk meningkatkan tingkat toleransi kesalahan Anda.

### Menguji failover otomatis

Setelah mengaktifkan failover otomatis, Anda dapat mengujinya menggunakan konsol ElastiCache, AWS CLI, dan API ElastiCache.

Saat menguji, perhatikan hal berikut:

- Anda dapat menggunakan operasi ini untuk menguji failover otomatis pada hingga lima serpihan (disebut grup simpul pada API ElastiCache dan AWS CLI) dalam periode kapan pun dari 24 jam bergulir.
- Jika Anda memanggil operasi ini pada serpihan di klaster yang berbeda (disebut grup replikasi pada API dan CLI), Anda dapat membuat panggilan secara bersamaan.
- Dalam beberapa kasus, Anda dapat memanggil operasi ini beberapa kali pada serpihan yang berbeda dalam grup replikasi Redis (mode klaster diaktifkan) yang sama. Dalam kasus tersebut, penggantian simpul pertama harus selesai sebelum panggilan berikutnya dapat dibuat.
- Untuk menentukan apakah penggantian simpul sudah selesai, periksa peristiwa menggunakan konsol Amazon ElastiCache, AWS CLI, atau API ElastiCache. Cari peristiwa berikut yang berkaitan dengan failover otomatis, yang tercantum berikut ini dengan urutan dari yang paling mungkin terjadi:

  1. Pesan grup replikasi: `Test Failover API called for node group <node-group-id>`
  2. Pesan klaster cache: `Failover from primary node <primary-node-id> to replica node <node-id> completed`
  3. Pesan grup replikasi: `Failover from primary node <primary-node-id> to replica node <node-id> completed`
  4. Pesan klaster cache: `Recovering cache nodes <node-id>`
  5. Pesan klaster cache: `Finished recovery for cache nodes <node-id>`

  Untuk informasi selengkapnya tentang IAM, lihat hal berikut:

  - Melihat peristiwa ElastiCache di Panduan Pengguna ElastiCache
  - DescribeEvents dalam Referensi API ElastiCache
  - describe-events di Referensi Perintah AWS CLI.
- API ini dirancang untuk menguji perilaku aplikasi Anda jika terjadi failover ElastiCache. Hal ini tidak dirancang untuk menjadi alat operasional untuk memulai failover guna mengatasi masalah dengan klaster. Selain itu, dalam kondisi tertentu seperti peristiwa operasional skala besar, AWS mungkin memblokir API ini.

Topik

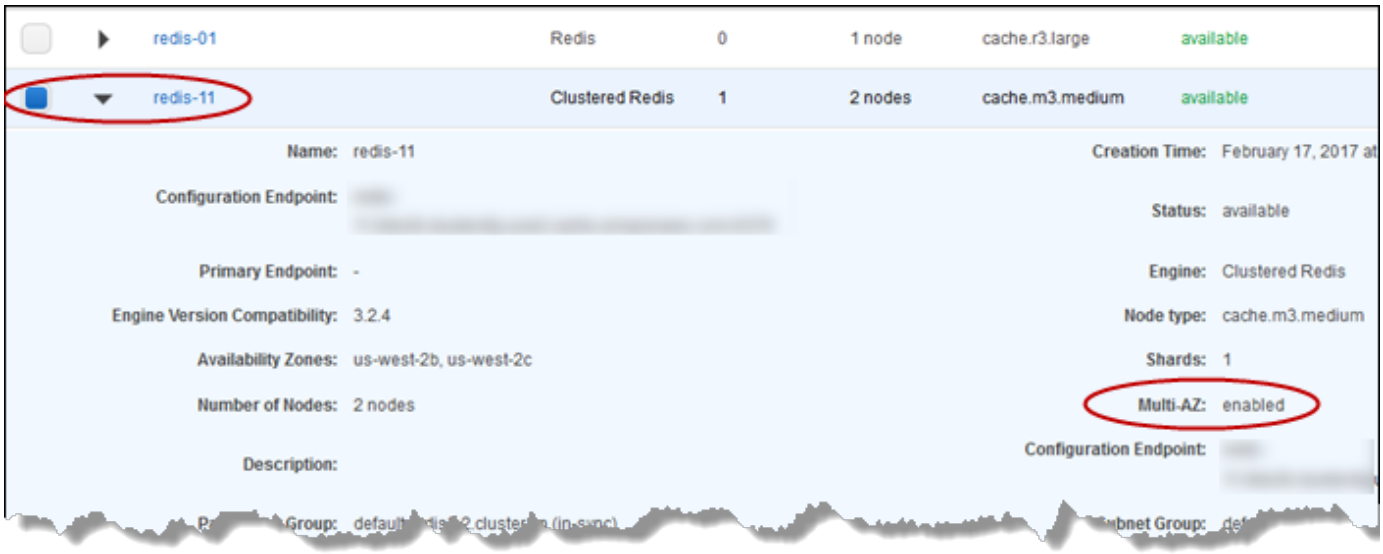


## Menguji failover otomatis menggunakan AWS Management Console

Gunakan prosedur berikut untuk menguji failover otomatis dengan konsol.

Untuk menguji failover otomatis

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Di panel navigasi, pilih Redis.
3. Dari daftar klaster Redis, pilih kotak di sebelah kiri klaster yang ingin diuji. Klaster ini harus memiliki setidaknya satu simpul replika baca.
4. Pada bagian Detail, lakukan konfirmasi bahwa klaster ini sudah mengaktifkan Multi-AZ. Jika klaster tidak mengaktifkan Multi-AZ, pilih klaster yang berbeda atau modifikasi klaster ini untuk mengaktifkan Multi-AZ. Untuk informasi selengkapnya, lihat Menggunakan AWS Management Console.




5. Untuk Redis (mode klaster dinonaktifkan), pilih nama klaster.

   Untuk Redis (mode klaster diaktifkan), lakukan hal berikut:

   a. Pilih nama klaster.

b. Di halaman Serpihan, untuk serpihan (disebut grup simpul pada API dan CLI) yang ingin dilakukan uji failover, pilih nama serpihan ini.

6. Di halaman Simpul, pilih Failover Primer.
7. Pilih Lanjutkan untuk melakukan fail over primer, atau Batalkan untuk membatalkan fail over simpul primer.

   Selama proses failover, konsol terus menunjukkan status simpul sebagai tersedia. Untuk memantau progres pengujian failover Anda, pilih Peristiwa dari panel navigasi konsol. Di tab Peristiwa, perhatikan peristiwa yang menunjukkan failover Anda telah dimulai (`Test Failover API called`) dan selesai (`Recovery completed`).

### Menguji failover otomatis menggunakan AWS CLI

Anda dapat menguji failover otomatis di setiap klaster yang mengaktifkan Multi-AZ menggunakan operasi AWS CLI `test-failover`.

Parameter

- `--replication-group-id` – Wajib. Grup replikasi (di konsol, klaster) yang akan diuji.
- `--node-group-id` – Wajib. Nama grup simpul yang ingin diuji failover otomatis. Anda dapat menguji maksimum lima grup simpul dalam periode 24 jam bergulir.

Contoh berikut menggunakan AWS CLI untuk menguji failover otomatis pada grup simpul `redis00-0003` di klaster Redis (mode klaster diaktifkan) `redis00`.

Example Menguji failover otomatis

Untuk Linux, macOS, atau Unix:

```
aws elasticache test-failover \
   --replication-group-id redis00 \
   --node-group-id redis00-0003
```

Untuk Windows:

```
aws elasticache test-failover ^
```

```
    --replication-group-id redis00 ^
    --node-group-id redis00-0003
```

Output dari perintah sebelumnya akan terlihat seperti berikut.

```
{
    "ReplicationGroup": {
        "Status": "available",
        "Description": "1 shard, 3 nodes (1 + 2 replicas)",
        "NodeGroups": [
            {
                "Status": "available",
                "NodeGroupMembers": [
                    {
                        "CurrentRole": "primary",
                        "PreferredAvailabilityZone": "us-west-2c",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Port": 6379,
                            "Address":
 "redis1x3-001.7ekv3t.0001.usw2.cache.amazonaws.com"
                        },
                        "CacheClusterId": "redis1x3-001"
                    },
                    {
                        "CurrentRole": "replica",
                        "PreferredAvailabilityZone": "us-west-2a",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Port": 6379,
                            "Address":
 "redis1x3-002.7ekv3t.0001.usw2.cache.amazonaws.com"
                        },
                        "CacheClusterId": "redis1x3-002"
                    },
                    {
                        "CurrentRole": "replica",
                        "PreferredAvailabilityZone": "us-west-2b",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Port": 6379,
                            "Address":
 "redis1x3-003.7ekv3t.0001.usw2.cache.amazonaws.com"
```

```
                    },
                    "CacheClusterId": "redis1x3-003"
                }
            ],
            "NodeGroupId": "0001",
            "PrimaryEndpoint": {
                "Port": 6379,
                "Address": "redis1x3.7ekv3t.ng.0001.usw2.cache.amazonaws.com"
            }
        }
    ],
    "ClusterEnabled": false,
    "ReplicationGroupId": "redis1x3",
    "SnapshotRetentionLimit": 1,
    "AutomaticFailover": "enabled",
    "MultiAZ": "enabled",
    "SnapshotWindow": "11:30-12:30",
    "SnapshottingClusterId": "redis1x3-002",
    "MemberClusters": [
        "redis1x3-001",
        "redis1x3-002",
        "redis1x3-003"
    ],
    "CacheNodeType": "cache.m3.medium",
    "DataTiering": "disabled",
    "PendingModifiedValues": {}
  }
}
```

Untuk memantau progres failover Anda, gunakan operasi AWS CLI `describe-events`.

Untuk informasi selengkapnya tentang IAM, lihat hal berikut:

- test-failover di Referensi Perintah AWS CLI.
- describe-events di Referensi Perintah AWS CLI.

### Menguji failover otomatis menggunakan API ElastiCache

Anda dapat menguji failover otomatis pada setiap klaster yang mengaktifkan Multi-AZ menggunakan operasi API ElastiCache `TestFailover`.

Parameter

- `ReplicationGroupId` – Wajib. Grup replikasi (di konsol, klaster) yang akan diuji.
- `NodeGroupId` – Wajib. Nama grup simpul yang ingin diuji failover otomatis. Anda dapat menguji maksimum lima grup simpul dalam periode 24 jam bergulir.

Contoh berikut menguji failover otomatis pada grup simpul `redis00-0003` pada grup replikasi (pada konsol, klaster) `redis00`.

Example Menguji failover otomatis

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=TestFailover
    &NodeGroupId=redis00-0003
    &ReplicationGroupId=redis00
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20140401T192317Z
    &X-Amz-Credential=<credential>
```

Untuk melacak kemajuan failover Anda, gunakan operasi API `DescribeEvents` ElastiCache.

Untuk informasi selengkapnya tentang IAM, lihat hal berikut:

- TestFailover dalam Referensi API ElastiCache
- DescribeEvents dalam Referensi API ElastiCache

Keterbatasan Multi-AZ Redis

Waspadai keterbatasan berikut pada Multi-AZ Redis:

- Multi-AZ didukung pada Redis versi 2.8.6 dan yang lebih baru.
- Redis Multi-AZ tidak didukung pada jenis simpul T1.
- Replikasi Redis bersifat asinkron. Karena itu, saat simpul primer melakukan fail over ke replika, sejumlah kecil data mungkin hilang karena ketertinggalan replikasi.

Saat memilih replika untuk dipromosikan menjadi primer, ElastiCache memilih replika dengan ketertinggalan replikasi paling sedikit. Dengan kata lain, yang dipilih adalah replika yang terkini. Hal ini membantu meminimalkan jumlah data yang hilang. Replika dengan ketertinggalan replikasi terkecil dapat berada di Availability Zone yang sama atau berbeda dari simpul primer yang gagal.

- Saat Anda secara manual mempromosikan replika baca menjadi primer pada Redis (dengan mode klaster dinonaktifkan), Anda dapat melakukannya hanya saat Multi-AZ dan failover otomatis dinonaktifkan. Untuk mempromosikan replika baca menjadi primer, lakukan langkah berikut:
  1. Nonaktifkan Multi-AZ pada klaster.
  2. Nonaktifkan failover otomatis pada klaster. Anda dapat melakukannya menggunakan konsol Redis dengan menghapus tanda centang pada Failover Otomatis untuk grup replikasi itu. Anda dapat melakukan ini dengan menggunakan AWS CLI dengan mengatur property `AutomaticFailoverEnabled` menjadi `false` saat memanggil operasi `ModifyReplicationGroup`.
  3. Promosikan replika baca menjadi primer.
  4. Aktifkan kembali Multi-AZ.
- Multi-AZ ElastiCache for Redis dan file append-only (AOF) bersifat saling eksklusif. Jika Anda mengaktifkan salah satunya, Anda tidak dapat mengaktifkan yang lain.
- Kegagalan simpul dapat disebabkan oleh peristiwa langka saat seluruh Availability Zone gagal. Dalam kasus ini, replika untuk menggantikan primer yang gagal dibuat hanya saat Availability Zone sudah dipulihkan. Sebagai contoh, misalkan grup replikasi dengan primer berada di Zona Ketersediaan-a, sedangkan replika ada di Zona Ketersediaan-b dan Zona Ketersediaan-c. Jika primer gagal, replika dengan ketertinggalan replikasi terkecil akan dipromosikan menjadi klaster primer. Setelah itu, ElastiCache membuat replika baru di AZ-a (tempat primer yang gagal) hanya saat AZ-a sudah dipulihkan dan tersedia.
- Booting ulang primer yang dilakukan oleh pelanggan tidak memicu failover otomatis. Booting ulang lain dan kegagalan akan memicu failover otomatis.
- Saat primer di-booting ulang, data dihapus dari primer saat primer kembali online. Saat replika baca melihat klaster primer yang bersih tanpa data, replika baca akan menghapus salinan datanya, yang menyebabkan hilangnya data.
- Setelah replika baca dipromosikan, replika lain melakukan sinkronisasi dengan primer yang baru. Setelah sinkronisasi awal, isi replikasi dihapus dan replika menyinkronkan data dari primer yang baru. Proses sinkronisasi ini menyebabkan gangguan singkat, sehingga replika tidak dapat diakses. Proses sinkronisasi juga menyebabkan peningkatan beban sementara pada primer pada saat melakukan sinkronisasi dengan replika. Perilaku ini adalah sifat asli Redis dan tidak unik untuk

Multi-AZ ElastiCache. Untuk perincian tentang perilaku Redis ini, lihat Replication di situs web Redis.

**Important**

Untuk Redis versi 2.8.22 dan lebih baru, Anda tidak dapat membuat replika eksternal. Untuk Redis versi sebelum 2.8.22, sebaiknya Anda tidak menghubungkan replikasi eksternal Redis dengan klaster ElastiCache for Redis yang mengaktifkan Multi-AZ. Konfigurasi yang tidak didukung ini dapat membuat masalah yang mencegah ElastiCache melakukan failover dan pemulihan dengan benar. Untuk menghubungkan replika Redis eksternal ke klaster ElastiCache, pastikan bahwa Multi-AZ tidak diaktifkan sebelum Anda membuat koneksi.

## Cara penerapan sinkronisasi dan pencadangan

Semua versi Redis yang didukung mendukung pencadangan dan sinkronisasi antara simpul primer dan replika. Namun, cara pencadangan dan sinkronisasi diterapkan bervariasi tergantung pada versi Redis.

### Redis Versi 2.8.22 dan Lebih Baru

Replikasi Redis, pada versi 2.8.22 dan lebih baru, mempunyai dua pilihan metode. Untuk informasi selengkapnya, lihat Versi Redis Sebelum 2.8.22 dan Melakukan snapshot dan pemulihan.

Selama proses forkless, jika beban tulis sedang berat, proses tulis ke klaster akan ditunda untuk memastikan bahwa Anda tidak menumpuk terlalu banyak perubahan dan dengan demikian menghalangi keberhasilan snapshot.

### Versi Redis Sebelum 2.8.22

Pencadangan dan sinkronisasi Redis pada versi sebelum 2.8.22 adalah proses tiga langkah.

1. Fork, dan dalam proses di latar belakang, mengurutkan data klaster secara bersambungan di disk. Ini membuat snapshot pada suatu titik dalam waktu.
2. Di latar depan, mengumpulkan log perubahan pada buffer keluaran klien.

   > **Important**
   >
   > Jika log perubahan melebihi ukuran buffer keluaran klien, pencadangan atau sinkronisasi akan gagal. Untuk informasi selengkapnya, lihat Memastikan bahwa Anda memiliki cukup memori untuk membuat snapshot Redis.

3. Akhirnya, mengirimkan data cache dan kemudian log perubahan ke simpul replika.

## Membuat grup replikasi Redis

Anda memiliki opsi berikut untuk membuat klaster dengan simpul replika. Opsi pertama berlaku jika Anda sudah memiliki klaster Redis (mode klaster dinonaktifkan) tersedia dan tidak terkait dengan klaster apa pun yang memiliki replika untuk digunakan sebagai simpul primer. Opsi lain berlaku jika Anda perlu membuat simpul primer dengan klaster dan replika baca. Saat ini, klaster Redis (mode klaster diaktifkan) harus dibuat dari awal.

Opsi 1: Membuat Grup Replikasi Menggunakan Klaster Redis (Mode Klaster Dinonaktifkan) Tersedia

Gunakan opsi ini untuk memanfaatkan klaster Redis (mode klaster dinonaktifkan) simpul tunggal yang telah ada. Anda menetapkan simpul yang telah ada ini sebagai simpul primer dalam klaster baru, dan kemudian secara tersendiri menambahkan 1 hingga 5 replika baca untuk klaster ini. Jika klaster yang telah ada ini aktif, replika baca melakukan sinkronisasi dengan klaster itu begitu replika dibuat. Lihat Membuat Grup Replikasi Menggunakan Klaster Redis (Mode Klaster Dinonaktifkan) Tersedia.

> **Important**
>
> Anda tidak dapat membuat klaster Redis (mode klaster diaktifkan) menggunakan klaster yang telah ada. Untuk membuat klaster Redis (mode klaster diaktifkan) (API/CLI: grup replikasi) menggunakan konsol ElastiCache, lihat Membuat klaster Redis (mode klaster diaktifkan) (Konsol).

Opsi 2: Membuat grup replikasi Redis dari awal

Gunakan opsi ini jika Anda belum memiliki klaster Redis (mode klaster dinonaktifkan) untuk digunakan sebagai simpul primer dari klaster, atau jika Anda ingin membuat klaster Redis (mode klaster diaktifkan). Lihat Membuat grup replikasi Redis dari awal.

### Membuat Grup Replikasi Menggunakan Klaster Redis (Mode Klaster Dinonaktifkan) Tersedia

Klaster yang tersedia adalah klaster Redis simpul tunggal yang telah ada. Saat ini, Redis (mode klaster diaktifkan) tidak mendukung pembuatan klaster dengan replika menggunakan klaster simpul tunggal yang tersedia. Jika Anda ingin membuat klaster Redis (mode klaster diaktifkan), lihat Membuat sebuah klaster Redis (Mode Klaster Diaktifkan) (Konsol).

Prosedur berikut hanya dapat digunakan jika Anda memiliki klaster simpul tunggal Redis (mode klaster dinonaktifkan). Simpul klaster ini menjadi simpul primer dalam klaster baru. Jika Anda tidak memiliki klaster Redis (mode klaster dinonaktifkan) yang dapat Anda gunakan sebagai klaster primer baru, lihat Membuat grup replikasi Redis dari awal.

#### Membuat Grup Replikasi Menggunakan Klaster Redis yang Tersedia (Konsol)

Lihat topik Menggunakan AWS Management Console.

#### Membuat Grup Replikasi Menggunakan Klaster Cache Redis yang Tersedia (AWS CLI)

Ada dua langkah untuk membuat grup replikasi dengan replika baca jika menggunakan Klaster Cache Redis yang tersedia untuk primer saat menggunakan AWS CLI.

Jika menggunakan AWS CLI, Anda membuat grup replikasi yang menentukan simpul mandiri yang tersedia sebagai simpul primer dari klaster, `--primary-cluster-id` dan jumlah simpul yang Anda inginkan dalam klaster menggunakan perintah CLI, `create-replication-group`. Sertakan parameter berikut.

--replication-group-id

Nama grup replikasi yang Anda buat. Nilai parameter ini digunakan sebagai dasar untuk nama simpul yang ditambahkan dengan nomor 3 digit yang berurutan ditambahkan ke belakang `--replication-group-id`. Sebagai contoh, `sample-repl-group-001`.

Kendala penamaan grup replikasi Redis (mode klaster dinonaktifkan) adalah sebagai berikut:

- Harus berisi 1–40 karakter alfanumerik atau tanda hubung.
- Harus dimulai dengan huruf.
- Tidak dapat berisi dua tanda hubung berturut-turut.
- Tidak dapat diakhiri dengan sebuah tanda hubung.

--replication-group-description

Deskripsi grup replikasi.

--num-node-groups

Jumlah simpul yang Anda inginkan dalam klaster ini. Nilai ini mencakup simpul primer. Parameter ini memiliki nilai maksimum sebesar enam.

--primary-cluster-id

Nama dari simpul klaster Redis (mode klaster dinonaktifkan) yang tersedia yang ingin Anda jadikan simpul primer dalam grup replikasi ini.

Perintah berikut membuat grup replikasi `sample-repl-group` menggunakan klaster Redis (mode kluster dinonaktifkan) `redis01` yang tersedia sebagai simpul primer dari grup replikasi. Ini membuat 2 simpul baru yang merupakan replika baca. Pengaturan dari `redis01` (yaitu grup parameter, grup keamanan, jenis simpul, versi mesin, dan seterusnya.) akan diterapkan untuk semua simpul di dalam grup replikasi.

Untuk Linux, macOS, atau Unix:

```
aws elasticache create-replication-group \
   --replication-group-id sample-repl-group \
   --replication-group-description "demo cluster with replicas" \
   --num-cache-clusters 3 \
   --primary-cluster-id redis01
```

Untuk Windows:

```
aws elasticache create-replication-group ^
   --replication-group-id sample-repl-group ^
   --replication-group-description "demo cluster with replicas" ^
   --num-cache-clusters 3 ^
   --primary-cluster-id redis01
```

Untuk informasi dan parameter tambahan yang mungkin ingin Anda gunakan, lihat topik AWS CLI create-replication-group.

Selanjutnya, tambahkan replika baca ke grup replikasi

Setelah grup replikasi dibuat, tambahkan satu hingga lima replika baca ke grup untuk menggunakan perintah `create-cache-cluster`, pastikan untuk menyertakan parameter berikut.

--cache-cluster-id

Nama klaster yang Anda tambahkan ke grup replikasi.

Kendala penamaan klaster adalah sebagai berikut:

- Harus berisi 1–40 karakter alfanumerik atau tanda hubung.
- Harus dimulai dengan huruf.
- Tidak dapat berisi dua tanda hubung berturut-turut.
- Tidak dapat diakhiri dengan sebuah tanda hubung.

--replication-group-id

Nama grup replikasi yang dituju untuk menambahkan klaster cache ini.

Ulangi perintah ini untuk setiap replika baca yang ingin Anda tambahkan ke grup replikasi, dengan mengubah nilai dari parameter `--cache-cluster-id` saja.

> **Note**
>
> Ingat, grup replikasi tidak dapat memiliki lebih dari lima replika baca. Jika Anda mencoba menambahkan replika baca ke grup replikasi yang sudah memiliki lima replika baca, maka operasi ini akan gagal.

Kode berikut menambahkan replika baca `my-replica01` ke grup replikasi `sample-repl-group`. Pengaturan dari klaster primer–grup parameter, grup keamanan, jenis simpul, dan sebagainya.–akan diterapkan ke simpul begitu simpul ditambahkan ke grup replikasi.

Untuk Linux, macOS, atau Unix:

```
aws elasticache create-cache-cluster \
   --cache-cluster-id my-replica01 \
   --replication-group-id sample-repl-group
```

Untuk Windows:

```
aws elasticache create-cache-cluster ^
   --cache-cluster-id my-replica01 ^
   --replication-group-id sample-repl-group
```

Keluaran dari perintah ini akan terlihat seperti ini.

```
{
    "ReplicationGroup": {
        "Status": "creating",
        "Description": "demo cluster with replicas",
        "ClusterEnabled": false,
        "ReplicationGroupId": "sample-repl-group",
        "SnapshotRetentionLimit": 1,
        "AutomaticFailover": "disabled",
        "SnapshotWindow": "00:00-01:00",
        "SnapshottingClusterId": "redis01",
        "MemberClusters": [
            "sample-repl-group-001",
            "sample-repl-group-002",
            "redis01"
        ],
        "CacheNodeType": "cache.m4.large",
        "DataTiering": "disabled",
        "PendingModifiedValues": {}
    }
}
```

Untuk informasi tambahan, lihat topik AWS CLI:

- create-replication-group
- modify-replication-group

Menambahkan replika ke klaster Redis (Mode Klaster Dinonaktifkan) mandiri (API ElastiCache)

Jika menggunakan API ElastiCache, Anda membuat grup replikasi yang menentukan simpul mandiri yang tersedia sebagai simpul primer dari klaster, `PrimaryClusterId` dan jumlah simpul yang Anda inginkan dalam klaster menggunakan perintah CLI, `CreateReplicationGroup`. Sertakan parameter berikut.

ReplicationGroupId

Nama grup replikasi yang Anda buat. Nilai parameter ini digunakan sebagai dasar untuk nama simpul yang ditambahkan dengan nomor 3 digit yang berurutan ditambahkan ke belakang `ReplicationGroupId`. Sebagai contoh, `sample-repl-group-001`.

Kendala penamaan grup replikasi Redis (mode klaster dinonaktifkan) adalah sebagai berikut:

- Harus berisi 1–40 karakter alfanumerik atau tanda hubung.
- Harus dimulai dengan huruf.
- Tidak dapat berisi dua tanda hubung berturut-turut.
- Tidak dapat diakhiri dengan sebuah tanda hubung.

ReplicationGroupDescription

Deskripsi klaster dengan replika.

NumCacheClusters

Jumlah simpul yang Anda inginkan dalam klaster ini. Nilai ini mencakup simpul primer. Parameter ini memiliki nilai maksimum sebesar enam.

PrimaryClusterId

Nama dari simpul klaster Redis (mode klaster dinonaktifkan) yang tersedia yang ingin Anda jadikan simpul primer dalam klaster ini.

Perintah berikut membuat klaster dengan replika `sample-repl-group` menggunakan klaster Redis (mode kluster dinonaktifkan) `redis01` yang tersedia sebagai simpul primer dari grup replikasi. Ini membuat 2 simpul baru yang merupakan replika baca. Pengaturan dari `redis01` (yaitu grup parameter, grup keamanan, jenis simpul, versi mesin, dan seterusnya.) akan diterapkan untuk semua simpul di dalam grup replikasi.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=CreateReplicationGroup
   &Engine=redis
   &EngineVersion=6.0
   &ReplicationGroupDescription=Demo%20cluster%20with%20replicas
   &ReplicationGroupId=sample-repl-group
   &PrimaryClusterId=redis01
   &Version=2015-02-02
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

Untuk informasi tambahan, lihat topik ElastiCache APL:

- CreateReplicationGroup
- ModifyReplicationGroup

Selanjutnya, tambahkan replika baca ke grup replikasi

Setelah grup replikasi dibuat, tambahkan satu hingga lima replika baca ke grup untuk menggunakan operasi `CreateCacheCluster`, pastikan untuk menyertakan parameter berikut.

CacheClusterId

Nama klaster yang Anda tambahkan ke grup replikasi.

Kendala penamaan klaster adalah sebagai berikut:

- Harus berisi 1–40 karakter alfanumerik atau tanda hubung.
- Harus dimulai dengan huruf.
- Tidak dapat berisi dua tanda hubung berturut-turut.
- Tidak dapat diakhiri dengan sebuah tanda hubung.

ReplicationGroupId

Nama grup replikasi yang dituju untuk menambahkan klaster cache ini.

Ulangi operasi ini untuk setiap replika baca yang ingin Anda tambahkan ke grup replikasi, dengan mengubah nilai dari parameter `CacheClusterId` saja.

Kode berikut menambahkan replika baca `myReplica01` ke grup replikasi `myReplGroup` Pengaturan klaster primer–grup parameter, grup keamanan, jenis simpul, dan sebagainya.–akan diterapkan ke simpul begitu simpul ditambahkan ke grup replikasi.

```
https://elasticache.us-west-2.amazonaws.com/
 ?Action=CreateCacheCluster
 &CacheClusterId=myReplica01
 &ReplicationGroupId=myReplGroup
 &SignatureMethod=HmacSHA256
 &SignatureVersion=4
 &Version=2015-02-02
 &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
 &X-Amz-Credential=[your-access-key-id]/20150202/us-west-2/elasticache/aws4_request
 &X-Amz-Date=20150202T170651Z
```

```
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=[signature-value]
```

Untuk informasi dan parameter tambahan yang mungkin ingin Anda gunakan, lihat topik CreateCacheCluster.

## Membuat grup replikasi Redis dari awal

Berikutnya, Anda dapat menemukan cara membuat grup replikasi Redis tanpa menggunakan klaster Redis yang telah ada sebagai primer. Anda dapat membuat grup replikasi Redis (mode klaster dinonaktifkan) atau Redis (mode klaster diaktifkan) dari awal menggunakan konsol ElastiCache, AWS CLI, atau API ElastiCache.

Sebelum melanjutkan, putuskan apakah Anda ingin membuat grup replikasi Redis (mode klaster dinonaktifkan) atau Redis (mode klaster diaktifkan). Untuk panduan dalam memutuskan, lihat Replikasi: Redis (Mode Klaster Dinonaktifkan) vs Redis (Mode Klaster Diaktifkan).

Topik

- Membuat grup replikasi Redis (Mode Klaster Dinonaktifkan) dari awal
- Membuat grup replikasi di Redis (Mode Klaster Diaktifkan) dari awal

### Membuat grup replikasi Redis (Mode Klaster Dinonaktifkan) dari awal

Anda dapat membuat grup replikasi Redis (mode klaster dinonaktifkan) dari awal menggunakan konsol ElastiCache, AWS CLI, atau API ElastiCache. Grup replikasi Redis (mode klaster dinonaktifkan) selalu memiliki satu grup simpul, klaster primer, dan hingga lima replika baca. Grup replikasi Redis (mode klaster dinonaktifkan) tidak mendukung partisi data Anda.

> **Note**
>
> Batas simpul/serpihan dapat ditambah hingga maksimum 500 per klaster. Untuk meminta penambahan batas, lihat Kuota Layanan AWS dan sertakan jenis instans dalam permintaan.

Untuk membuat grup replikasi Redis (mode klaster dinonaktifkan) dari awal, lakukan salah satu pendekatan berikut:

#### Membuat grup replikasi Redis (Mode Klaster Dinonaktifkan) dari awal (AWS CLI)

Prosedur berikut membuat grup replikasi Redis (mode klaster dinonaktifkan) menggunakan AWS CLI.

Saat Anda membuat grup replikasi Redis (mode klaster dinonaktifkan) dari awal, Anda membuat grup replikasi dan semua simpulnya dengan satu panggilan ke perintah AWS CLI `create-replication-group`. Sertakan parameter berikut.

--replication-group-id

Nama grup replikasi yang Anda buat.

Kendala penamaan grup replikasi Redis (mode klaster dinonaktifkan) adalah sebagai berikut:

- Harus berisi 1–40 karakter alfanumerik atau tanda hubung.
- Harus dimulai dengan huruf.
- Tidak dapat berisi dua tanda hubung berturut-turut.
- Tidak dapat diakhiri dengan sebuah tanda hubung.

--replication-group-description

Deskripsi grup replikasi.

--num-cache-clusters

Jumlah simpul yang ingin dibuat dengan grup replikasi ini, digabungkan dengan primer dan replika baca.

Jika Anda mengaktifkan Multi-AZ (`--automatic-failover-enabled`), nilai dari `--num-cache-clusters` harus minimal 2.

--cache-node-type

Jenis simpul untuk setiap simpul di dalam grup replikasi.

Jenis simpul berikut didukung oleh ElastiCache. Secara umum, tipe generasi saat ini memberikan lebih banyak memori dan daya komputasi dengan biaya lebih rendah dibandingkan dengan tipe generasi sebelumnya yang setara.

Untuk informasi selengkapnya tentang detail performa untuk setiap tipe simpul, lihat Tipe Instans Amazon EC2.

--data-tiering-enabled

Atur parameter ini jika Anda menggunakan jenis simpul r6gd. Jika Anda tidak ingin tingkatan data, atur `--no-data-tiering-enabled`. Untuk informasi selengkapnya, lihat Tingkatan data.

--cache-parameter-group

Tentukan grup parameter yang sesuai dengan versi mesin Anda. Jika Anda menjalankan Redis 3.2.4 atau yang lebih baru, tentukan grup parameter `default.redis3.2` atau grup parameter yang berasal dari `default.redis3.2` untuk membuat grup replikasi Redis (mode klaster dinonaktifkan). Untuk informasi selengkapnya, lihat Parameter spesifik Redis.

--network-type

Baik `ipv4`, `ipv6` atau `dual-stack`. Jika Anda memilih dual-stack, Anda harus mengatur parameter `--IpDiscovery` ke salah satu `ipv4` atau `ipv6`.

--engine

redis

--engine-version

Untuk mendapatkan kumpulan fitur terbanyak, pilih versi mesin terbaru.

Nama simpul akan diambil dari nama grup replikasi dengan tambahan di belakang `-00#` pada nama grup replikasi. Sebagai contoh, menggunakan nama grup replikasi `myReplGroup`, maka nama untuk primer menjadi `myReplGroup-001` dan replika baca menjadi `myReplGroup-002` hingga `myReplGroup-006`.

Jika Anda ingin mengaktifkan enkripsi bergerak atau diam pada grup replikasi ini, tambahkan salah satu atau kedua parameter `--transit-encryption-enabled` atau `--at-rest-encryption-enabled` dan penuhi ketentuan berikut.

- Grup replikasi Anda harus menjalankan Redis 3.2.6 atau 4.0.10.
- Grup replikasi harus dibuat di dalam Amazon VPC.
- Anda juga harus menyertakan parameter `--cache-subnet-group`.
- Anda juga harus menyertakan parameter `--auth-token` dengan nilai string yang ditentukan oleh pelanggan untuk token (kata sandi) AUTH Anda yang diperlukan untuk melakukan operasi pada grup replikasi ini.

Operasi berikut membuat grup replikasi Redis (mode klaster dinonaktifkan) `sample-repl-group` dengan tiga simpul, satu primer dan dua replika.

Untuk Linux, macOS, atau Unix:

```
aws elasticache create-replication-group \
   --replication-group-id sample-repl-group \
   --replication-group-description "Demo cluster with replicas" \
   --num-cache-clusters 3 \
   --cache-node-type cache.m4.large \
   --engine redis
```

Untuk Windows:

```
aws elasticache create-replication-group ^
   --replication-group-id sample-repl-group ^
   --replication-group-description "Demo cluster with replicas" ^
   --num-cache-clusters 3 ^
   --cache-node-type cache.m4.large ^
   --engine redis
```

Keluaran dari perintah ini adalah seperti ini.

```
{
    "ReplicationGroup": {
        "Status": "creating",
        "Description": "Demo cluster with replicas",
```

```
        "ClusterEnabled": false,
        "ReplicationGroupId": "sample-repl-group",
        "SnapshotRetentionLimit": 0,
        "AutomaticFailover": "disabled",
        "SnapshotWindow": "01:30-02:30",
        "MemberClusters": [
            "sample-repl-group-001",
            "sample-repl-group-002",
            "sample-repl-group-003"
        ],
        "CacheNodeType": "cache.m4.large",
        "DataTiering": "disabled",
        "PendingModifiedValues": {}
    }
}
```

Untuk informasi dan parameter tambahan yang mungkin ingin Anda gunakan, lihat topik AWS CLI create-replication-group.

Membuat grup replikasi Redis (mode klaster dinonaktifkan) dari awal (API ElastiCache)

Prosedur berikut membuat grup replikasi Redis (mode klaster dinonaktifkan) menggunakan API ElastiCache.

Saat Anda membuat grup replikasi Redis (mode klaster dinonaktifkan) dari awal, Anda membuat grup replikasi dan semua simpulnya dengan satu panggilan ke operasi `CreateReplicationGroup` API ElastiCache. Sertakan parameter berikut.

ReplicationGroupId

Nama grup replikasi yang Anda buat.

Kendala penamaan grup replikasi Redis (mode klaster diaktifkan) adalah sebagai berikut:

- Harus berisi 1–40 karakter alfanumerik atau tanda hubung.
- Harus dimulai dengan huruf.
- Tidak dapat berisi dua tanda hubung berturut-turut.
- Tidak dapat diakhiri dengan sebuah tanda hubung.

ReplicationGroupDescription

Deskripsi grup replikasi.

NumCacheClusters

Jumlah simpul yang ingin dibuat dengan grup replikasi ini, digabungkan dengan primer dan replika baca.

Jika Anda mengaktifkan Multi-AZ (`AutomaticFailoverEnabled=true`), nilai dari `NumCacheClusters` harus minimal 2.

CacheNodeType

Jenis simpul untuk setiap simpul di dalam grup replikasi.

Jenis simpul berikut didukung oleh ElastiCache. Secara umum, tipe generasi saat ini memberikan lebih banyak memori dan daya komputasi dengan biaya lebih rendah dibandingkan dengan tipe generasi sebelumnya yang setara.

Untuk informasi selengkapnya tentang detail performa untuk setiap tipe simpul, lihat Tipe Instans Amazon EC2.

--data-tiering-enabled

Atur parameter ini jika Anda menggunakan jenis simpul r6gd. Jika Anda tidak ingin tingkatan data, atur `--no-data-tiering-enabled`. Untuk informasi selengkapnya, lihat Tingkatan data.

CacheParameterGroup

Tentukan grup parameter yang sesuai dengan versi mesin Anda. Jika Anda menjalankan Redis 3.2.4 atau yang lebih baru, tentukan grup parameter `default.redis3.2` atau grup parameter yang berasal dari `default.redis3.2` untuk membuat grup replikasi Redis (mode klaster dinonaktifkan). Untuk informasi selengkapnya, lihat Parameter spesifik Redis.

--network-type

Baik `ipv4`, `ipv` atau `dual-stack`. Jika Anda memilih dual-stack, Anda harus mengatur parameter `--IpDiscovery` ke salah satu `ipv4` atau `ipv6`.

Mesin

redis

VersiMesin

6.0

Nama simpul akan diambil dari nama grup replikasi dengan tambahan di belakang `-00#` pada nama grup replikasi. Sebagai contoh, menggunakan nama grup replikasi `myReplGroup`, maka nama untuk primer menjadi `myReplGroup-001` dan replika baca menjadi `myReplGroup-002` hingga `myReplGroup-006`.

Jika Anda ingin mengaktifkan enkripsi bergerak atau diam pada grup replikasi ini, tambahkan salah satu atau kedua parameter `TransitEncryptionEnabled=true` atau `AtRestEncryptionEnabled=true` dan penuhi ketentuan berikut.

- Grup replikasi Anda harus menjalankan Redis 3.2.6 atau 4.0.10.
- Grup replikasi harus dibuat di dalam Amazon VPC.
- Anda juga harus menyertakan parameter `CacheSubnetGroup`.
- Anda juga harus menyertakan parameter `AuthToken` dengan nilai string yang ditentukan oleh pelanggan untuk token (kata sandi) AUTH Anda yang diperlukan untuk melakukan operasi pada grup replikasi ini.

Operasi berikut membuat grup replikasi Redis (mode klaster dinonaktifkan) `myReplGroup` dengan tiga simpul, satu primer dan dua replika.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=CreateReplicationGroup
   &CacheNodeType=cache.m4.large
   &CacheParameterGroup=default.redis6.x
   &Engine=redis
   &EngineVersion=6.0
   &NumCacheClusters=3
   &ReplicationGroupDescription=test%20group
   &ReplicationGroupId=myReplGroup
   &Version=2015-02-02
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

Untuk informasi dan parameter tambahan yang mungkin ingin Anda gunakan, lihat topik CreateReplicationGroup API ElastiCache.

## Membuat grup replikasi di Redis (Mode Klaster Diaktifkan) dari awal

Anda dapat membuat klaster Redis (mode klaster diaktifkan) (API/CLI: grup replikasi) menggunakan konsol ElastiCache, AWS CLI, atau API ElastiCache. Grup replikasi Redis (mode klaster diaktifkan) memiliki dari 1 hingga 500 serpihan (API/CLI: grup simpul), simpul primer di setiap serpihan, dan maksimum hingga 5 replika baca di setiap serpihan. Anda dapat membuat klaster dengan jumlah serpihan lebih banyak dan jumlah replika lebih sedikit dengan jumlah hingga 90 simpul per klaster. Konfigurasi klaster ini dapat berkisar dari 90 serpihan dan 0 replika hingga 15 serpihan dan 5 replika, yang merupakan jumlah replika maksimum yang diperbolehkan.

Batas simpul atau serpihan dapat ditingkatkan hingga maksimum 500 per klaster jika versi mesin Redis adalah 5.0.6 atau lebih tinggi. Sebagai contoh, Anda dapat memilih untuk membuat konfigurasi dari sebuah klaster dengan 500 simpul yang berkisar antara 83 serpihan (satu primer dan 5 replika per serpihan) dan 500 serpihan (primer tunggal dan tidak ada replika). Pastikan alamat IP yang tersedia mencukupi untuk mengakomodasi peningkatan tersebut. Kesalahan umum termasuk subnet dalam grup subnet memiliki rentang CIDR yang terlalu kecil atau subnet digunakan bersama dan banyak digunakan oleh klaster lain. Untuk informasi selengkapnya, lihat Membuat grup subnet.

Untuk versi di bawah 5.0.6, batasnya adalah 250 per klaster.

Untuk meminta penambahan batas, lihat Kuota Layanan AWS dan pilih jenis batas Simpul per klaster per jenis instans.

### Membuat sebuah Klaster di Redis (Mode Klaster Diaktifkan)

- Membuat sebuah klaster Redis (Mode Klaster Diaktifkan) (Konsol)
- Membuat grup replikasi di Redis (Mode Klaster Diaktifkan) dari awal (AWS CLI)
- Membuat grup replikasi di Redis (Mode Klaster Diaktifkan) dari awal (API ElastiCache)

### Membuat sebuah klaster Redis (Mode Klaster Diaktifkan) (Konsol)

Untuk membuat klaster Redis (mode klaster diaktifkan), lihat Membuat klaster Redis (mode klaster diaktifkan) (Konsol). Pastikan untuk mengaktifkan mode klaster, Mode Klaster diaktifkan (Menskalakan ke Luar), dan tentukan setidaknya dua serpihan dan satu simpul replika di setiap serpihan.

### Membuat grup replikasi di Redis (Mode Klaster Diaktifkan) dari awal (AWS CLI)

Prosedur berikut membuat grup replikasi Redis (mode klaster dinonaktifkan) menggunakan AWS CLI.

Saat Anda membuat grup replikasi Redis (mode klaster dinonaktifkan) dari awal, Anda membuat grup replikasi dan semua simpulnya dengan satu panggilan ke perintah AWS CLI `create-replication-group`. Sertakan parameter berikut.

--replication-group-id

Nama grup replikasi yang Anda buat.

Kendala penamaan grup replikasi Redis (mode klaster diaktifkan) adalah sebagai berikut:

- Harus berisi 1–40 karakter alfanumerik atau tanda hubung.
- Harus dimulai dengan huruf.
- Tidak dapat berisi dua tanda hubung berturut-turut.
- Tidak dapat diakhiri dengan sebuah tanda hubung.

--replication-group-description

Deskripsi grup replikasi.

--cache-node-type

Jenis simpul untuk setiap simpul di dalam grup replikasi.

Jenis simpul berikut didukung oleh ElastiCache. Secara umum, tipe generasi saat ini memberikan lebih banyak memori dan daya komputasi dengan biaya lebih rendah dibandingkan dengan tipe generasi sebelumnya yang setara.

Untuk informasi selengkapnya tentang detail performa untuk setiap tipe simpul, lihat Tipe Instans Amazon EC2.

--data-tiering-enabled

Atur parameter ini jika Anda menggunakan jenis simpul r6gd. Jika Anda tidak ingin tingkatan data, atur `--no-data-tiering-enabled`. Untuk informasi selengkapnya, lihat Tingkatan data.

--cache-parameter-group

Tentukan grup parameter `default.redis6.x.cluster.on` atau grup parameter yang berasal dari `default.redis6.x.cluster.on` untuk membuat grup replikasi Redis (mode klaster diaktifkan). Untuk informasi selengkapnya, lihat Perubahan parameter Redis 6.x.

--engine

redis

--engine-version

3.2.4

--num-node-groups

Jumlah grup simpul dalam grup replikasi ini. Nilai yang valid adalah 1 sampai 500.

> **Note**
>
> Batas simpul/serpihan dapat ditambah hingga maksimum 500 per klaster. Untuk meminta penambahan batas, lihat Kuota LayananAWS dan pilih jenis batas "Simpul per klaster per jenis instans".

--replicas-per-node-group

Jumlah simpul replika di setiap grup simpul. Nilai yang valid adalah 0 sampai 5.

--network-type

Baik `ipv4`, `ipv` atau `dual-stack`. Jika Anda memilih dual-stack, Anda harus mengatur parameter `--IpDiscovery` ke salah satu `ipv4` atau `ipv6`.

Jika Anda ingin mengaktifkan enkripsi bergerak atau diam pada grup replikasi ini, tambahkan salah satu atau kedua parameter `--transit-encryption-enabled` atau `--at-rest-encryption-enabled` dan penuhi ketentuan berikut.

- Grup replikasi Anda harus menjalankan Redis 3.2.6 atau 4.0.10.
- Grup replikasi harus dibuat di dalam Amazon VPC.
- Anda juga harus menyertakan parameter `--cache-subnet-group`.
- Anda juga harus menyertakan parameter `--auth-token` dengan nilai string yang ditentukan oleh pelanggan untuk token (kata sandi) AUTH Anda yang diperlukan untuk melakukan operasi pada grup replikasi ini.

Operasi berikut membuat grup replikasi Redis (mode klaster diaktifkan) `sample-repl-group` dengan tiga grup simpul/serpihan (--num-node-group), masing-masing dengan tiga simpul, satu primer dan dua replika baca (--replicas-per-node-group).

Untuk Linux, macOS, atau Unix:

```
aws elasticache create-replication-group \
   --replication-group-id sample-repl-group \
   --replication-group-description "Demo cluster with replicas" \
   --num-node-groups 3 \
   --replicas-per-node-group 2 \
   --cache-node-type cache.m4.large \
   --engine redis \
   --security-group-ids SECURITY_GROUP_ID \
   --cache-subnet-group-name SUBNET_GROUP_NAME>
```

Untuk Windows:

```
aws elasticache create-replication-group ^
   --replication-group-id sample-repl-group ^
   --replication-group-description "Demo cluster with replicas" ^
   --num-node-groups 3 ^
   --replicas-per-node-group 2 ^
   --cache-node-type cache.m4.large ^
   --engine redis ^
   --security-group-ids SECURITY_GROUP_ID ^
   --cache-subnet-group-name SUBNET_GROUP_NAME>
```

Perintah sebelumnya menghasilkan keluaran berikut.

```
{
    "ReplicationGroup": {
        "Status": "creating",
        "Description": "Demo cluster with replicas",
        "ReplicationGroupId": "sample-repl-group",
        "SnapshotRetentionLimit": 0,
        "AutomaticFailover": "enabled",
        "SnapshotWindow": "05:30-06:30",
        "MemberClusters": [
            "sample-repl-group-0001-001",
            "sample-repl-group-0001-002",
            "sample-repl-group-0001-003",
            "sample-repl-group-0002-001",
            "sample-repl-group-0002-002",
            "sample-repl-group-0002-003",
            "sample-repl-group-0003-001",
```

```
            "sample-repl-group-0003-002",
            "sample-repl-group-0003-003"
        ],
        "PendingModifiedValues": {}
    }
}
```

Saat Anda membuat grup replikasi Redis (mode klaster diaktifkan) dari awal, Anda dapat mengonfigurasi setiap serpihan di klaster menggunakan parameter `--node-group-configuration` seperti yang ditunjukkan pada contoh berikut yang membuat konfigurasi untuk dua grup simpul (Konsol: serpihan). Serpihan pertama memiliki dua simpul, satu primer dan satu replika baca. Serpihan kedua memiliki tiga simpul, satu primer dan dua replika baca.

--node-group-configuration

Konfigurasi untuk setiap grup simpul. Parameter `--node-group-configuration` terdiri dari bidang berikut.

- `PrimaryAvailabilityZone` – Zona Ketersediaan yang menjadi lokasi dari simpul primer dari grup simpul ini. Jika parameter ini dihilangkan, ElastiCache memilih Zona Ketersediaan untuk simpul primer.

  Contoh: as-barat-2a.

- `ReplicaAvailabilityZones` – Sebuah daftar dipisahkan koma untuk Zona Ketersediaan yang menjadi lokasi replika baca. Jumlah Zona Ketersediaan di dalam daftar harus sesuai dengan nilai dari `ReplicaCount`. Jika parameter ini dihilangkan, ElastiCache memilih Zona Ketersediaan untuk simpul replika.

  Contoh: "as-barat-2a,as-barat-2b,as-barat-2c"

- `ReplicaCount` – Jumlah simpul replika di setiap grup simpul.

- `Slots` – String yang menentukan keyspace untuk grup simpul. String menggunakan format `startKey-endKey`. Jika parameter ini dihilangkan, ElastiCache mengalokasikan kunci dengan spesifikasi yang sama di antara grup simpul.

  Contoh: "0-4999"

Operasi berikut membuat grup replikasi Redis (mode klaster diaktifkan) `new-group` dengan grup/serpihan yang terdiri dari dua simpul (`--num-node-groups`). Tidak seperti contoh sebelumnya, setiap grup simpul dikonfigurasi berbeda dari grup simpul lainnya (`--node-group-configuration`).

Untuk Linux, macOS, atau Unix:

```
aws elasticache create-replication-group \
  --replication-group-id new-group \
  --replication-group-description "Sharded replication group" \
  --engine redis \
  --snapshot-retention-limit 8 \
  --cache-node-type cache.m4.medium \
  --num-node-groups 2 \
  --node-group-configuration \
      "ReplicaCount=1,Slots=0-8999,PrimaryAvailabilityZone='us-
east-1c',ReplicaAvailabilityZones='us-east-1b'" \
      "ReplicaCount=2,Slots=9000-16383,PrimaryAvailabilityZone='us-
east-1a',ReplicaAvailabilityZones='us-east-1a','us-east-1c'"
```

Untuk Windows:

```
aws elasticache create-replication-group ^
  --replication-group-id new-group ^
  --replication-group-description "Sharded replication group" ^
  --engine redis ^
  --snapshot-retention-limit 8 ^
  --cache-node-type cache.m4.medium ^
  --num-node-groups 2 ^
  --node-group-configuration \
      "ReplicaCount=1,Slots=0-8999,PrimaryAvailabilityZone='us-
east-1c',ReplicaAvailabilityZones='us-east-1b'" \
      "ReplicaCount=2,Slots=9000-16383,PrimaryAvailabilityZone='us-
east-1a',ReplicaAvailabilityZones='us-east-1a','us-east-1c'"
```

Operasi sebelumnya menghasilkan keluaran berikut.

```
{
    "ReplicationGroup": {
        "Status": "creating",
        "Description": "Sharded replication group",
```

```
        "ReplicationGroupId": "rc-rg",
        "SnapshotRetentionLimit": 8,
        "AutomaticFailover": "enabled",
        "SnapshotWindow": "10:00-11:00",
        "MemberClusters": [
            "rc-rg-0001-001",
            "rc-rg-0001-002",
            "rc-rg-0002-001",
            "rc-rg-0002-002",
            "rc-rg-0002-003"
        ],
        "PendingModifiedValues": {}
    }
}
```

Untuk informasi dan parameter tambahan yang mungkin ingin Anda gunakan, lihat topik AWS CLI create-replication-group.

Membuat grup replikasi di Redis (Mode Klaster Diaktifkan) dari awal (API ElastiCache)

Prosedur berikut membuat grup replikasi Redis (mode klaster dinonaktifkan) menggunakan API ElastiCache.

Saat Anda membuat grup replikasi Redis (mode klaster diaktifkan) dari awal, Anda membuat grup replikasi dan semua simpulnya dengan satu panggilan ke operasi `CreateReplicationGroup` API ElastiCache. Sertakan parameter berikut.

ReplicationGroupId

Nama grup replikasi yang Anda buat.

Kendala penamaan grup replikasi Redis (mode klaster diaktifkan) adalah sebagai berikut:

- Harus berisi 1–40 karakter alfanumerik atau tanda hubung.
- Harus dimulai dengan huruf.
- Tidak dapat berisi dua tanda hubung berturut-turut.
- Tidak dapat diakhiri dengan sebuah tanda hubung.

ReplicationGroupDescription

Deskripsi grup replikasi.

NumNodeGroups

Jumlah grup simpul yang ingin Anda buat dengan grup replikasi ini. Nilai yang valid adalah 1 sampai 500.

ReplicasPerNodeGroup

Jumlah simpul replika di setiap grup simpul. Nilai yang valid adalah 1 sampai 5.

NodeGroupConfiguration

Konfigurasi untuk setiap grup simpul. Parameter `NodeGroupConfiguration` terdiri dari bidang berikut.

- `PrimaryAvailabilityZone` – Zona Ketersediaan yang menjadi lokasi dari simpul primer dari grup simpul ini. Jika parameter ini dihilangkan, ElastiCache memilih Zona Ketersediaan untuk simpul primer.

  Contoh: as-barat-2a.
- `ReplicaAvailabilityZones` – Sebuah daftar Zona Ketersediaan yang menjadi lokasi replika baca. Jumlah Zona Ketersediaan di dalam daftar harus sesuai dengan nilai dari `ReplicaCount`. Jika parameter ini dihilangkan, ElastiCache memilih Zona Ketersediaan untuk simpul replika.
- `ReplicaCount` – Jumlah simpul replika di setiap grup simpul.
- `Slots` – String yang menentukan keyspace untuk grup simpul. String menggunakan format `startKey-endKey`. Jika parameter ini dihilangkan, ElastiCache mengalokasikan kunci dengan spesifikasi yang sama di antara grup simpul.

  Contoh: “0-4999"

CacheNodeType

Jenis simpul untuk setiap simpul di dalam grup replikasi.

Jenis simpul berikut didukung oleh ElastiCache. Secara umum, tipe generasi saat ini memberikan lebih banyak memori dan daya komputasi dengan biaya lebih rendah dibandingkan dengan tipe generasi sebelumnya yang setara.

Untuk informasi selengkapnya tentang detail performa untuk setiap tipe simpul, lihat Tipe Instans Amazon EC2.

--data-tiering-enabled

Atur parameter ini jika Anda menggunakan jenis simpul r6gd. Jika Anda tidak ingin tingkatan data, atur `--no-data-tiering-enabled`. Untuk informasi selengkapnya, lihat Tingkatan data.

CacheParameterGroup

Tentukan grup parameter `default.redis6.x.cluster.on` atau grup parameter yang berasal dari `default.redis6.x.cluster.on` untuk membuat grup replikasi Redis (mode klaster aktif). Untuk informasi selengkapnya, lihat Perubahan parameter Redis 6.x.

--network-type

Baik `ipv4`, `ipv` atau `dual-stack`. Jika Anda memilih dual-stack, Anda harus mengatur parameter `--IpDiscovery` ke salah satu `ipv4` atau `ipv6`.

Mesin

redis

VersiMesin

6.0

Jika Anda ingin mengaktifkan enkripsi saat dalam pengiriman atau penyimpanan pada grup replikasi ini, tambahkan salah satu atau kedua parameter `TransitEncryptionEnabled=true` atau `AtRestEncryptionEnabled=true` dan penuhi ketentuan berikut.

- Grup replikasi Anda harus menjalankan Redis 3.2.6 atau 4.0.10.
- Grup replikasi harus dibuat di dalam Amazon VPC.
- Anda juga harus menyertakan parameter `CacheSubnetGroup`.
- Anda juga harus menyertakan parameter `AuthToken` dengan nilai string yang ditentukan oleh pelanggan untuk token (kata sandi) AUTH Anda yang diperlukan untuk melakukan operasi pada grup replikasi ini.

Jeda baris ditambahkan agar dapat lebih mudah dibaca.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=CreateReplicationGroup
   &CacheNodeType=cache.m4.large
   &CacheParemeterGroup=default.redis6.xcluster.on
   &Engine=redis
```

```
    &EngineVersion=6.0
    &NumNodeGroups=3
    &ReplicasPerNodeGroup=2
    &ReplicationGroupDescription=test%20group
    &ReplicationGroupId=myReplGroup
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &X-Amz-Credential=<credential>
```

Untuk informasi dan parameter tambahan yang mungkin ingin Anda gunakan, lihat topik CreateReplicationGroup API ElastiCache.

## Melihat detail grup replikasi

Ada kalanya Anda mungkin ingin melihat detail grup replikasi. Anda dapat menggunakan konsol ElastiCache, AWS CLI untuk ElastiCache, atau API ElastiCache. Proses konsol berbeda untuk Redis (mode klaster dinonaktifkan) dan Redis (mode klaster diaktifkan).

Melihat Detail Grup Replikasi

- Melihat detail untuk Redis (Mode Klaster Dinonaktifkan) dengan replika
  - Melihat Detail untuk Grup Replikasi Redis (Mode Klaster Dinonaktifkan) (Konsol)
  - Melihat Detail untuk grup replikasi Redis (Mode Klaster Dinonaktifkan) (AWS CLI)
  - Melihat Detail untuk Grup Replikasi Redis (Mode Klaster Dinonaktifkan) (API ElastiCache)
- Melihat detail grup replikasi: Redis (Mode Klaster Diaktifkan)
  - Melihat detail untuk klaster Redis (Mode Klaster Diaktifkan) (Konsol)
  - Melihat detail untuk klaster Redis (Mode Klaster Diaktifkan) (AWS CLI)
  - Melihat detail untuk klaster Redis (Mode Klaster Diaktifkan) (API ElastiCache)
- Melihat detail grup replikasi (AWS CLI)
- Melihat detail grup replikasi (API ElastiCache)

### Melihat detail untuk Redis (Mode Klaster Dinonaktifkan) dengan replika

Anda dapat melihat detail klaster Redis (mode klaster dinonaktifkan) dengan replika (API/CLI: grup replikasi) menggunakan konsol ElastiCache, AWS CLI untuk ElastiCache, atau API ElastiCache.

Melihat Detail Klaster Redis (Mode Klaster Dinonaktifkan)

- Melihat Detail untuk Grup Replikasi Redis (Mode Klaster Dinonaktifkan) (Konsol)
- Melihat Detail untuk grup replikasi Redis (Mode Klaster Dinonaktifkan) (AWS CLI)
- Melihat Detail untuk Grup Replikasi Redis (Mode Klaster Dinonaktifkan) (API ElastiCache)

Melihat Detail untuk Grup Replikasi Redis (Mode Klaster Dinonaktifkan) (Konsol)

Untuk melihat detail klaster Redis (mode klaster dinonaktifkan) dengan replika menggunakan konsol ElastiCache, lihat topik Melihat detail dari klaster Redis (Mode Klaster Dinonaktifkan) (Konsol).

Melihat Detail untuk grup replikasi Redis (Mode Klaster Dinonaktifkan) (AWS CLI)

Untuk contoh AWS CLI yang menampilkan detail grup replikasi Redis (mode klaster dinonaktifkan), lihat Melihat detail grup replikasi (AWS CLI).

Melihat Detail untuk Grup Replikasi Redis (Mode Klaster Dinonaktifkan) (API ElastiCache)

Untuk contoh API ElastiCache yang menampilkan detail grup replikasi Redis (mode klaster dinonaktifkan), lihat Melihat detail grup replikasi (API ElastiCache).

Melihat detail grup replikasi: Redis (Mode Klaster Diaktifkan)

Melihat detail untuk klaster Redis (Mode Klaster Diaktifkan) (Konsol)

Untuk melihat detail klaster Redis (mode klaster diaktifkan) menggunakan konsol ElastiCache, lihat Melihat detail untuk klaster Redis (Mode Klaster Diaktifkan) (Konsol).

Melihat detail untuk klaster Redis (Mode Klaster Diaktifkan) (AWS CLI)

Untuk contoh ElastiCache CLI yang menampilkan detail grup replikasi Redis (mode klaster diaktifkan), lihat Melihat detail grup replikasi (AWS CLI).

Melihat detail untuk klaster Redis (Mode Klaster Diaktifkan) (API ElastiCache)

Untuk contoh API ElastiCache yang menampilkan detail grup replikasi Redis (mode klaster diaktifkan), lihat Melihat detail grup replikasi (API ElastiCache).

Melihat detail grup replikasi (AWS CLI)

Anda dapat melihat detail untuk grup replikasi menggunakan perintah AWS CLI `describe-replication-groups`. Gunakan parameter opsional berikut untuk menyaring daftar. Menghilangkan parameter mengembalikan detail hingga 100 grup replikasi.

Parameter Opsional

- `--replication-group-id` – Gunakan parameter ini untuk mencantumkan detail grup replikasi tertentu. Jika grup replikasi yang ditentukan memiliki lebih dari satu grup simpul, hasil yang ditampilkan akan dikelompokkan berdasarkan grup simpul.
- `--max-items` – Gunakan parameter ini untuk membatasi jumlah grup replikasi yang dicantumkan. Nilai dari `--max-items` tidak boleh kurang dari 20 atau lebih besar dari 100.

Example

Kode berikut mencantumkan detail hingga 100 grup replikasi.

```
aws elasticache describe-replication-groups
```

Kode berikut mencantumkan detail untuk `sample-repl-group`.

```
aws elasticache describe-replication-groups --replication-group-id sample-repl-group
```

Kode berikut mencantumkan detail untuk `sample-repl-group`.

```
aws elasticache describe-replication-groups --replication-group-id sample-repl-group
```

Kode berikut mencantumkan detail hingga 25 grup replikasi.

```
aws elasticache describe-replication-groups --max-items 25
```

Keluaran dari operasi ini terlihat seperti berikut (format JSON).

```
{
   "ReplicationGroups": [
     {
       "Status": "available",
       "Description": "test",
       "NodeGroups": [
         {
            "Status": "available",
               "NodeGroupMembers": [
                  {
                     "CurrentRole": "primary",
                     "PreferredAvailabilityZone": "us-west-2a",
```

```
                  "CacheNodeId": "0001",
                  "ReadEndpoint": {
                     "Port": 6379,
                     "Address": "rg-name-001.1abc4d.0001.usw2.cache.amazonaws.com"
                  },
                  "CacheClusterId": "rg-name-001"
               },
               {
                  "CurrentRole": "replica",
                  "PreferredAvailabilityZone": "us-west-2b",
                  "CacheNodeId": "0001",
                  "ReadEndpoint": {
                     "Port": 6379,
                     "Address": "rg-name-002.1abc4d.0001.usw2.cache.amazonaws.com"
                  },
                  "CacheClusterId": "rg-name-002"
               },
               {
                  "CurrentRole": "replica",
                  "PreferredAvailabilityZone": "us-west-2c",
                  "CacheNodeId": "0001",
                  "ReadEndpoint": {
                     "Port": 6379,
                     "Address": "rg-name-003.1abc4d.0001.usw2.cache.amazonaws.com"
                  },
                  "CacheClusterId": "rg-name-003"
               }
            ],
            "NodeGroupId": "0001",
            "PrimaryEndpoint": {
               "Port": 6379,
               "Address": "rg-name.1abc4d.ng.0001.usw2.cache.amazonaws.com"
            }
         }
      ],
      "ReplicationGroupId": "rg-name",
      "AutomaticFailover": "enabled",
      "SnapshottingClusterId": "rg-name-002",
      "MemberClusters": [
         "rg-name-001",
         "rg-name-002",
         "rg-name-003"
      ],
      "PendingModifiedValues": {}
```

```
      },
      {
      ... some output omitted for brevity
      }
   ]
}
```

Untuk informasi selengkapnya, lihat AWS CLI topik untuk ElastiCache describe-replication-groups.

Melihat detail grup replikasi (API ElastiCache)

Anda dapat melihat detail untuk replikasi menggunakan operasi AWS CLI `DescribeReplicationGroups`. Gunakan parameter opsional berikut untuk menyaring daftar. Menghilangkan parameter mengembalikan detail hingga 100 grup replikasi.

Parameter Opsional

- `ReplicationGroupId` – Gunakan parameter ini untuk mencantumkan detail grup replikasi tertentu. Jika grup replikasi yang ditentukan memiliki lebih dari satu grup simpul, hasil yang ditampilkan akan dikelompokkan berdasarkan grup simpul.
- `MaxRecords` – Gunakan parameter ini untuk membatasi jumlah grup replikasi yang dicantumkan. Nilai dari `MaxRecords` tidak boleh kurang dari 20 atau lebih besar dari 100. Defaultnya adalah 100.

Example

Kode berikut mencantumkan detail hingga 100 grup replikasi.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeReplicationGroups
   &Version=2015-02-02
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

Kode berikut mencantumkan detail untuk `myReplGroup`.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeReplicationGroups
```

```
   &ReplicationGroupId=myReplGroup
   &Version=2015-02-02
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

Kode berikut mencantumkan detail hingga 25 klaster.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeReplicationGroups
   &MaxRecords=25
   &Version=2015-02-02
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

Untuk informasi selengkapnya, lihat DescribeReplicationGroups dalam topik referensi API ElastiCache.

## Menemukan titik akhir grup replikasi

Aplikasi dapat menyambung ke simpul apa pun di dalam grup replikasi, asalkan memiliki titik akhir DNS dan nomor port untuk simpul tersebut. Tergantung pada apakah Anda menjalankan grup replikasi Redis (mode klaster dinonaktifkan) atau Redis (mode klaster diaktifkan), Anda akan tertarik pada titik akhir yang berbeda.

Redis (Mode Klaster Dinonaktifkan)

Klaster Redis (mode klaster dinonaktifkan) dengan replika memiliki tiga jenis titik akhir; titik akhir primer, titik akhir pembaca dan titik akhir simpul. Titik akhir primer adalah nama DNS yang selalu diterjemahkan sebagai simpul primer di klaster. Titik akhir primer kebal terhadap perubahan klaster Anda, seperti menaikkan replika baca menjadi peran primer. Untuk aktivitas tulis, sebaiknya aplikasi Anda menyambung ke titik akhir primer.

Titik akhir pembaca akan secara merata membagi koneksi masuk ke titik akhir di antara semua replika baca di dalam sebuah klaster ElastiCache for Redis. Faktor lain seperti saat aplikasi membuat koneksi atau cara aplikasi menggunakan ulang koneksi akan menentukan distribusi lalu lintas. Titik akhir pembaca tetap mengikuti perubahan klaster secara langsung saat replika ditambahkan atau dihapus. Anda dapat menempatkan beberapa replika baca dari klaster ElastiCache for Redis pada Zona Ketersediaan (AZ) AWS yang berbeda untuk memastikan ketersediaan tinggi dari titik akhir pembaca.

> **Note**
>
> Titik akhir pembaca bukan penyeimbang beban. Titik akhir pembaca adalah catatan DNS yang akan diterjemahkan sebagai alamat IP dari salah satu simpul replika dengan mode round robin.

Untuk aktivitas baca, aplikasi juga dapat terhubung ke simpul apa pun di klaster. Tidak seperti titik akhir primer, titik akhir simpul diterjemahkan ke titik akhir tertentu. Jika Anda membuat perubahan di dalam klaster Anda, seperti menambahkan atau menghapus replika, Anda harus memperbarui titik akhir simpul di aplikasi Anda.

Redis (Mode Klaster Diaktifkan)

Klaster Redis (mode klaster diaktifkan) dengan replika, karena mereka memiliki beberapa serpihan (API/CLI: grup simpul), yang berarti mereka juga memiliki beberapa simpul primer, memiliki struktur

titik akhir yang berbeda dari klaster Redis (mode klaster dinonaktifkan). Redis (mode klaster diaktifkan) memiliki titik akhir konfigurasi yang "mengetahui" semua titik akhir primer dan simpul di klaster. Aplikasi Anda menyambung ke titik akhir konfigurasi. Setiap saat aplikasi Anda menulis ke atau membaca dari titik akhir konfigurasi klaster, Redis, di belakang layar, menentukan serpihan yang menyediakan kunci dan titik akhir yang mana di dalam serpihan itu yang akan digunakan. Ini semua cukup transparan untuk aplikasi Anda.

Anda dapat menemukan titik akhir untuk klaster menggunakan konsol ElastiCache, AWS CLI, atau API ElastiCache.

Menemukan Titik Akhir Grup Replikasi

Untuk menemukan titik akhir untuk grup replikasi Anda, lihat salah satu topik berikut:

- Menemukan Titik Akhir Klaster Redis (Mode Klaster Dinonaktifkan) (Konsol)
- Menemukan Titik Akhir Klaster Redis (Mode Klaster Dinonaktifkan) (Konsol)
- Menemukan Titik Akhir untuk Grup Replikasi (AWS CLI)
- Menemukan Titik Akhir untuk Grup Replikasi (API ElastiCache)

# Mengubah grup replikasi

⚠ Kendala Penting

- Saat ini, ElastiCache mendukung perubahan terbatas dari grup replikasi Redis (mode klaster diaktifkan), misalnya mengubah versi mesin, menggunakan operasi API `ModifyReplicationGroup` (CLI: `modify-replication-group`). Anda dapat mengubah jumlah serpihan (grup simpul) dalam klaster Redis (mode klaster diaktifkan) dengan operasi API `ModifyReplicationGroupShardConfiguration` (CLI: `modify-replication-group-shard-configuration`). Untuk informasi selengkapnya, lihat Penskalaan klaster di Redis (Mode Klaster Diaktifkan).

  Perubahan lain untuk klaster Redis (mode klaster diaktifkan) mengharuskan Anda membuat klaster dengan klaster baru yang memasukkan perubahan tersebut.
- Anda dapat meningkatkan klaster serta grup replikasi Redis (mode klaster dinonaktifkan) dan Redis (mode klaster diaktifkan) ke versi mesin yang lebih baru. Namun, Anda tidak dapat menurunkan ke versi mesin yang lebih lama kecuali dengan menghapus klaster atau grup replikasi yang telah ada dan membuatnya lagi. Untuk informasi selengkapnya, lihat Versi mesin dan pemutakhiran .
- Anda dapat meningkatkan klaster ElastiCache for Redis yang sudah ada yang menggunakan mode klaster dinonaktifkan untuk menggunakan mode klaster aktif, menggunakan konsol, API ModifyReplicationGroup atau perintah CLI modify-replication-group, seperti yang ditunjukkan pada contoh di bawah ini. Atau Anda dapat mengikuti langkah-langkah di Mengubah mode klaster.

Anda dapat mengubah pengaturan klaster Redis (mode klaster dinonaktifkan) menggunakan konsol ElastiCache, AWS CLI, atau API ElastiCache. Saat ini, ElastiCache mendukung sejumlah modifikasi terbatas pada grup replikasi Redis (mode klaster diaktifkan). Perubahan lain mengharuskan Anda membuat cadangan dari grup replikasi saat ini dan kemudian menggunakan cadangan itu untuk menyemai grup replikasi Redis (mode klaster diaktifkan) yang baru.

Topik

- Menggunakan AWS Management Console
- Menggunakan AWS CLI
- Menggunakan API ElastiCache

Menggunakan AWS Management Console

Untuk mengubah sebuah klaster Redis (mode klaster dinonaktifkan), lihat Memodifikasi sebuah cluster ElastiCache .

Menggunakan AWS CLI

Berikut ini adalah AWS CLI contoh perintah `modify-replication-group`. Anda dapat menggunakan perintah yang sama untuk membuat perubahan lain pada grup replikasi.

Mengaktifkan Multi-AZ pada grup replikasi Redis yang telah ada:

Untuk Linux, macOS, atau Unix:

```
aws elasticache modify-replication-group \
   --replication-group-id myReplGroup \
   --multi-az-enabled = true
```

Untuk Windows:

```
aws elasticache modify-replication-group ^
   --replication-group-id myReplGroup ^
   --multi-az-enabled
```

Mengubah mode klaster dari dinonaktifkan menjadi diaktifkan:

Untuk mengubah mode klaster dari dinonaktifkan ke diaktifkan, Anda harus terlebih dahulu mengatur mode klaster ke kompatibel. Mode yang kompatibel memungkinkan klien Redis Anda untuk terhubung baik menggunakan mode klaster diaktifkan maupun mode klaster dinonaktifkan. Setelah Anda memigrasikan semua klien Redis untuk menggunakan mode klaster diaktifkan, Anda kemudian dapat menyelesaikan konfigurasi mode klaster dan mengatur mode klaster ke diaktifkan.

Untuk Linux, macOS, atau Unix:

Atur ke mode klaster agar kompatibel.

```
aws elasticache modify-replication-group \
   --replication-group-id myReplGroup \
   --cache-parameter-group-name myParameterGroupName \
   --cluster-mode compatible
```

Atur ke mode klaster untuk diaktifkan.

```
aws elasticache modify-replication-group \
   --replication-group-id myReplGroup \
   --cluster-mode enabled
```

Untuk Windows:

Atur ke mode klaster agar kompatibel.

```
aws elasticache modify-replication-group ^
   --replication-group-id myReplGroup ^
   --cache-parameter-group-name myParameterGroupName ^
   --cluster-mode compatible
```

Atur ke mode klaster untuk diaktifkan.

```
aws elasticache modify-replication-group ^
   --replication-group-id myReplGroup ^
   --cluster-mode enabled
```

Untuk informasi selengkapnya tentang perintah AWS CLI `modify-replication-group`, lihat modify-replication-group atau Mengubah mode klaster di Panduan Pengguna ElastiCache for Redis.

Menggunakan API ElastiCache

Operasi API ElastiCache berikut mengaktifkan Multi-AZ pada grup replikasi Redis yang telah ada. Anda dapat menggunakan operasi yang sama untuk membuat perubahan lain pada grup replikasi.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=ModifyReplicationGroup
   &AutomaticFailoverEnabled=true
   &Mutli-AZEnabled=true
   &ReplicationGroupId=myReplGroup
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20141201T220302Z
   &Version=2014-12-01
   &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
   &X-Amz-Date=20141201T220302Z
   &X-Amz-SignedHeaders=Host
```

```
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

Untuk informasi lain tentang operasi `ModifyReplicationGroup` API ElastiCache, see ModifyReplicationGroup.

## Menghapus grup replikasi

Jika Anda tidak lagi memerlukan salah satu klaster dengan replika (disebut grup replikasi pada API/CLI), Anda dapat menghapusnya. Saat Anda menghapus grup replikasi, ElastiCache menghapus semua simpul di grup itu.

Setelah Anda memulainya, operasi ini tidak dapat diganggu atau dibatalkan.

> **Warning**
>
> Saat Anda menghapus klaster ElastiCache for Redis, snapshot manual Anda akan dipertahankan. Anda juga memiliki pilihan untuk membuat snapshot terakhir sebelum klaster dihapus. Snapshot cache otomatis tidak dipertahankan.

### Menghapus Grup Replikasi (Konsol)

Untuk menghapus klaster yang memiliki replika, lihat Menghapus klaster.

### Menghapus Grup Replikasi (AWS CLI)

Gunakan perintah delete-replication-group untuk menghapus grup replikasi.

```
aws elasticache delete-replication-group --replication-group-id my-repgroup
```

Prompt tampil meminta konfirmasi keputusan Anda. Masukkan y (ya) untuk memulai operasi itu dengan segera. Setelah dimulai, proses ini tidak dapat dipulihkan.

```
  After you begin deleting this replication group, all of its nodes will be deleted as
well.
  Are you sure you want to delete this replication group? [Ny]y

REPLICATIONGROUP  my-repgroup  My replication group  deleting
```

### Menghapus grup replikasi (API ElastiCache)

Memanggil DeleteReplicationGroup dengan parameter `ReplicationGroup`.

Example

```
https://elasticache.us-west-2.amazonaws.com/
```

```
?Action=DeleteReplicationGroup
&ReplicationGroupId=my-repgroup
&Version=2014-12-01
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20141201T220302Z
&X-Amz-Algorithm=&AWS;4-HMAC-SHA256
&X-Amz-Date=20141201T220302Z
&X-Amz-SignedHeaders=Host
&X-Amz-Expires=20141201T220302Z
&X-Amz-Credential=<credential>
&X-Amz-Signature=<signature>
```

**Note**

Jika Anda menetapkan parameter `RetainPrimaryCluster` menjadi `true`, maka semua replika baca akan dihapus, tetapi klaster primer akan dipertahankan.

## Mengubah jumlah replika

Anda dapat secara dinamis menambah atau mengurangi jumlah replika baca dalam grup replikasi Redis Anda menggunakan AWS Management Console, AWS CLI, atau API ElastiCache. Jika grup replikasi Anda adalah grup replikasi Redis (mode klaster diaktifkan), Anda dapat memilih serpihan (grup simpul) yang akan ditambah atau dikurangi jumlah replikanya.

Untuk secara dinamis mengubah jumlah replika dalam grup replikasi Redis Anda, pilih operasi dari tabel berikut yang sesuai dengan situasi Anda.

| Untuk Melakukannya | Untuk Redis (mode klaster diaktifkan) | Untuk Redis (mode klaster dinonaktifkan) |
|---|---|---|
| Menambah replika | Menambah jumlah replika dalam serpihan | Menambah jumlah replika dalam serpihan<br><br>Menambahkan replika baca, untuk grup replikasi Redis (Mode Klaster Dinonaktifkan) |
| Menghapus replika | Mengurangi jumlah replika dalam serpihan | Mengurangi jumlah replika dalam serpihan<br><br>Menghapus replika baca, untuk grup replikasi Redis (Mode Klaster Dinonaktifkan) |

## Menambah jumlah replika dalam serpihan

Anda dapat menambah jumlah replika dalam serpihan Redis (mode klaster diaktifkan) atau grup replikasi Redis (mode klaster dinonaktifkan) hingga maksimum lima. Anda dapat melakukannya dengan menggunakan AWS Management Console, AWS CLI, atau API ElastiCache.

Topik

- Menggunakan AWS Management Console
- Menggunakan AWS CLI
- Menggunakan API ElastiCache

### Menggunakan AWS Management Console

Prosedur berikut menggunakan konsol untuk menambah jumlah replika di dalam grup replikasi Redis (mode klaster diaktifkan).

Untuk menambah jumlah replika dalam serpihan Redis

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Pada panel navigasi, pilih Redis, lalu pilih nama grup replikasi yang ingin ditambah replikanya.
3. Pilih kotak untuk setiap serpihan yang replikanya akan ditambah.
4. Pilih Tambah replika.
5. Lengkapi halaman Tambahkan Replika ke Serpihan:
   - Untuk Jumlah replikasi/serpihan baru, masukkan jumlah replika yang Anda inginkan untuk semua serpihan yang dipilih. Nilai ini harus lebih besar atau sama dengan Jumlah Replika Sekarang per serpihan dan kurang dari atau sama dengan lima. Kami menyarankan setidaknya dua replika sebagai syarat minimum.
   - Untuk Zona Ketersediaan, pilih salah satu dari Tidak ada pilihan agar ElastiCache yang memilih Zona Ketersediaan untuk setiap replika baru, atau Tentukan Zona Ketersediaan untuk memilih Zona Ketersediaan untuk setiap replika baru.

     Jika Anda memilih Tentukan Zona Ketersediaan, tentukan satu Zona Ketersediaan untuk setiap replika baru menggunakan daftar.
6. Pilih Tambahkan untuk menambahkan replika atau Batalkan untuk membatalkan operasi.

### Menggunakan AWS CLI

Untuk menambah jumlah replika dalam serpihan Redis, gunakan perintah `increase-replica-count` dengan parameter berikut:

- `--replication-group-id` – Diperlukan. Mengidentifikasi grup replikasi yang ingin ditambah jumlah replikanya.
- `--apply-immediately` atau `--no-apply-immediately` – Diperlukan. Menentukan apakah akan menambah jumlah replika segera (`--apply-immediately`) atau pada jendela pemeliharaan berikutnya (`--no-apply-immediately`). Saat ini, `--no-apply-immediately` belum didukung.
- `--new-replica-count` – Opsional. Menentukan jumlah simpul replika yang Anda inginkan saat selesai, maksimum hingga lima. Gunakan parameter ini untuk grup replikasi Redis (mode klaster dinonaktifkan) yang memiliki hanya satu grup simpul atau grup Redis (mode klaster diaktifkan), atau di tempat yang semua grup simpulnya diinginkan memiliki jumlah replika yang sama. Jika nilai ini tidak lebih besar dari jumlah replika saat ini di dalam grup simpul, panggilan akan gagal dengan pengecualian.
- `--replica-configuration` – Opsional. Memungkinkan Anda untuk menentukan jumlah replika dan Zona Ketersediaan untuk setiap grup simpul secara bebas. Gunakan parameter ini untuk grup Redis (mode klaster diaktifkan) tempat Anda ingin membuat konfigurasi setiap grup simpul secara terpisah.

  `--replica-configuration` memiliki tiga anggota opsional:

  - `NodeGroupId` – ID empat digit untuk grup simpul yang sedang dikonfigurasi. Untuk grup replikasi Redis (mode klaster dinonaktifkan), ID serpihan selalu `0001`. Untuk menemukan ID grup simpul (serpihan) Redis (mode klaster diaktifkan), lihat Menemukan ID serpihan.
  - `NewReplicaCount` – Jumlah replika yang Anda inginkan di dalam grup simpul ini pada akhir operasi ini. Nilai ini harus lebih besar dari jumlah replika saat ini, maksimum hingga lima. Jika nilai ini tidak lebih besar dari jumlah replika saat ini di dalam grup simpul, panggilan akan gagal dengan pengecualian.
  - `PreferredAvailabilityZones` – Daftar string `PreferredAvailabilityZone` yang menentukan Zona Ketersediaan yang akan menjadi tempat simpul dari grup replikasi berada. Jumlah nilai `PreferredAvailabilityZone` harus sama dengan nilai `NewReplicaCount` ditambah 1 untuk memperhitungkan simpul primer. Jika anggota `--replica-configuration` ini dihilangkan, ElastiCache for Redis memilih Zona Ketersediaan untuk setiap replika baru.

> **Important**
>
> Anda harus menyertakan parameter `--new-replica-count` atau `--replica-configuration`, tapi tidak keduanya sekaligus, dalam panggilan Anda.

Example

Contoh berikut menambah jumlah replika di dalam grup replikasi `sample-repl-group` menjadi tiga. Ketika contoh ini selesai, terdapat tiga replika di setiap grup simpul. Jumlah ini berlaku terlepas dari apakah ini adalah grup Redis (mode klaster dinonaktifkan) dengan grup simpul tunggal atau grup Redis (mode klaster diaktifkan) dengan beberapa grup simpul.

Untuk Linux, macOS, atau Unix:

```
aws elasticache increase-replica-count \
    --replication-group-id sample-repl-group \
    --new-replica-count 3 \
    --apply-immediately
```

Untuk Windows:

```
aws elasticache increase-replica-count ^
    --replication-group-id sample-repl-group ^
    --new-replica-count 3 ^
    --apply-immediately
```

Contoh berikut menambah jumlah replika di dalam grup replikasi `sample-repl-group` ke nilai yang ditentukan untuk kedua grup simpul yang ditentukan tersebut. Mengingat adanya beberapa grup simpul, ini adalah grup replikasi Redis (mode klaster diaktifkan). Saat menentukan `PreferredAvailabilityZones` opsional, jumlah Zona Ketersediaan yang tercantum harus sama dengan nilai dari `NewReplicaCount` ditambah 1. Pendekatan ini memperhitungkan simpul primer untuk grup yang diidentifikasi oleh `NodeGroupId`.

Untuk Linux, macOS, atau Unix:

```
aws elasticache increase-replica-count \
    --replication-group-id sample-repl-group \
    --replica-configuration \
```

```
        NodeGroupId=0001,NewReplicaCount=2,PreferredAvailabilityZones=us-east-1a,us-
east-1c,us-east-1b \
        NodeGroupId=0003,NewReplicaCount=3,PreferredAvailabilityZones=us-east-1a,us-
east-1b,us-east-1c,us-east-1c \
    --apply-immediately
```

Untuk Windows:

```
aws elasticache increase-replica-count ^
    --replication-group-id sample-repl-group ^
    --replica-configuration ^
        NodeGroupId=0001,NewReplicaCount=2,PreferredAvailabilityZones=us-east-1a,us-
east-1c,us-east-1b ^
        NodeGroupId=0003,NewReplicaCount=3,PreferredAvailabilityZones=us-east-1a,us-
east-1b,us-east-1c,us-east-1c \
    --apply-immediately
```

Untuk informasi selengkapnya tentang penambahan jumlah replika menggunakan CLI, lihat increase-replica-count di Referensi Baris Perintah Amazon ElastiCache.

Menggunakan API ElastiCache

Untuk menambah jumlah replika dalam serpihan Redis, gunakan perintah `IncreaseReplicaCount` dengan parameter berikut:

- `ReplicationGroupId` – Diperlukan. Mengidentifikasi grup replikasi yang ingin ditambah jumlah replikanya.
- `ApplyImmediately` – Diperlukan. Menentukan apakah akan menambah jumlah replika segera (`ApplyImmediately=True`) atau pada jendela pemeliharaan berikutnya (`ApplyImmediately=False`). Saat ini, `ApplyImmediately=False` belum didukung.
- `NewReplicaCount` – Opsional. Menentukan jumlah simpul replika yang Anda inginkan saat selesai, maksimum hingga lima. Gunakan parameter ini untuk grup replikasi Redis (mode klaster dinonaktifkan) yang memiliki hanya satu grup simpul, atau grup Redis (mode klaster diaktifkan) yang ingin disamakan jumlah replikanya. Jika nilai ini tidak lebih besar dari jumlah replika saat ini di dalam grup simpul, panggilan akan gagal dengan pengecualian.
- `ReplicaConfiguration` – Opsional. Memungkinkan Anda untuk menentukan jumlah replika dan Zona Ketersediaan untuk setiap grup simpul secara bebas. Gunakan parameter ini untuk grup Redis (mode klaster diaktifkan) tempat Anda ingin membuat konfigurasi setiap grup simpul secara terpisah.

`ReplicaConfiguraion` memiliki tiga anggota opsional:

- `NodeGroupId` – ID empat digit untuk grup simpul yang sedang dikonfigurasi. Untuk grup replikasi Redis (mode klaster dinonaktifkan), ID grup simpul (serpihan) selalu `0001`. Untuk menemukan ID grup simpul (serpihan) Redis (mode klaster diaktifkan), lihat Menemukan ID serpihan.
- `NewReplicaCount` – Jumlah replika yang Anda inginkan di dalam grup simpul ini pada akhir operasi ini. Nilai tersebut harus lebih dari jumlah replika saat ini dan maksimum sebanyak lima. Jika nilai ini tidak lebih besar dari jumlah replika saat ini di dalam grup simpul, panggilan akan gagal dengan pengecualian.
- `PreferredAvailabilityZones` – Daftar string `PreferredAvailabilityZone` yang menentukan Zona Ketersediaan yang akan menjadi tempat simpul dari grup replikasi berada. Jumlah nilai `PreferredAvailabilityZone` harus sama dengan nilai `NewReplicaCount` ditambah 1 untuk memperhitungkan simpul primer. Jika anggota `ReplicaConfiguration` ini dihilangkan, ElastiCache for Redis memilih Zona Ketersediaan untuk setiap replika baru.

> **Important**
>
> Anda harus menyertakan parameter `NewReplicaCount` atau `ReplicaConfiguration`, tapi tidak keduanya sekaligus, dalam panggilan Anda.

Example

Contoh berikut menambah jumlah replika di dalam grup replikasi `sample-repl-group` menjadi tiga. Ketika contoh ini selesai, terdapat tiga replika di setiap grup simpul. Jumlah ini berlaku terlepas dari apakah ini adalah grup Redis (mode klaster dinonaktifkan) dengan grup simpul tunggal atau grup Redis (mode klaster diaktifkan) dengan beberapa grup simpul.

```
https://elasticache.us-west-2.amazonaws.com/
      ?Action=IncreaseReplicaCount
      &ApplyImmediately=True
      &NewReplicaCount=3
      &ReplicationGroupId=sample-repl-group
      &Version=2015-02-02
      &SignatureVersion=4
      &SignatureMethod=HmacSHA256
      &Timestamp=20150202T192317Z
```

```
      &X-Amz-Credential=<credential>
```

Contoh berikut menambah jumlah replika di dalam grup replikasi `sample-repl-group` ke nilai yang ditentukan untuk kedua grup simpul yang ditentukan tersebut. Mengingat adanya beberapa grup simpul, ini adalah grup replikasi Redis (mode klaster diaktifkan). Saat menentukan `PreferredAvailabilityZones` opsional, jumlah Zona Ketersediaan yang tercantum harus sama dengan nilai dari `NewReplicaCount` ditambah 1. Pendekatan ini memperhitungkan simpul primer, untuk grup yang diidentifikasi dengan `NodeGroupId`.

```
https://elasticache.us-west-2.amazonaws.com/
      ?Action=IncreaseReplicaCount
      &ApplyImmediately=True
      &ReplicaConfiguration.ConfigureShard.1.NodeGroupId=0001
      &ReplicaConfiguration.ConfigureShard.1.NewReplicaCount=2

 &ReplicaConfiguration.ConfigureShard.1.PreferredAvailabilityZones.PreferredAvailabilityZone.1=
east-1a

 &ReplicaConfiguration.ConfigureShard.1.PreferredAvailabilityZones.PreferredAvailabilityZone.2=
east-1c

 &ReplicaConfiguration.ConfigureShard.1.PreferredAvailabilityZones.PreferredAvailabilityZone.3=
east-1b
      &ReplicaConfiguration.ConfigureShard.2.NodeGroupId=0003
      &ReplicaConfiguration.ConfigureShard.2.NewReplicaCount=3

 &ReplicaConfiguration.ConfigureShard.2.PreferredAvailabilityZones.PreferredAvailabilityZone.1=
east-1a

 &ReplicaConfiguration.ConfigureShard.2.PreferredAvailabilityZones.PreferredAvailabilityZone.2=
east-1b

 &ReplicaConfiguration.ConfigureShard.2.PreferredAvailabilityZones.PreferredAvailabilityZone.3=
east-1c

 &ReplicaConfiguration.ConfigureShard.2.PreferredAvailabilityZones.PreferredAvailabilityZone.4=
east-1c
      &ReplicationGroupId=sample-repl-group
      &Version=2015-02-02
      &SignatureVersion=4
      &SignatureMethod=HmacSHA256
      &Timestamp=20150202T192317Z
```

```
    &X-Amz-Credential=<credential>
```

Untuk informasi selengkapnya tentang penambahan jumlah replika menggunakan API, lihat IncreaseReplicaCount di Referensi API Amazon ElastiCache.

## Mengurangi jumlah replika dalam serpihan

Anda dapat mengurangi jumlah replika dalam serpihan untuk Redis (mode klaster diaktifkan), atau dalam grup replikasi untuk Redis (mode klaster dinonaktifkan):

- Untuk Redis (mode klaster dinonaktifkan), Anda dapat mengurangi jumlah replika hingga satu jika Multi-AZ diaktifkan, dan hingga nol jika tidak diaktifkan.
- Untuk Redis (mode klaster diaktifkan), Anda dapat mengurangi jumlah replika hingga nol. Namun, Anda tidak dapat melakukan fail over ke replika jika simpul primer Anda gagal.

Anda dapat menggunakan AWS Management Console, AWS CLI atau API ElastiCache untuk mengurangi jumlah replika di dalam grup simpul (serpihan) atau grup replikasi.

Topik

- Menggunakan AWS Management Console
- Menggunakan AWS CLI
- Menggunakan API ElastiCache

### Menggunakan AWS Management Console

Prosedur berikut menggunakan konsol tersebut untuk menambah jumlah replika di dalam grup replikasi Redis (mode klaster diaktifkan).

Mengurangi jumlah replika dalam serpihan Redis

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Pada panel navigasi, pilih Redis, lalu pilih nama grup replikasi yang ingin dihapus replika di dalamnya.
3. Pilih kotak untuk setiap serpihan yang ingin dihapus simpul replika di dalamnya.
4. Pilih Hapus replika.
5. Lengkapi halaman Hapus Replika dari Serpihan:

    a. Untuk Jumlah replikasi/serpihan baru, masukkan jumlah replika yang Anda inginkan untuk serpihan yang dipilih. Nilai ini harus lebih besar atau sama dengan 1. Kami menyarankan setidaknya dua replika per serpihan sebagai syarat minimum.

b. Pilih Hapus untuk menghapus replika atau Batalkan untuk membatalkan operasi.

> **Important**
>
> - Jika Anda tidak menentukan simpul replika yang akan dihapus, ElastiCache secara otomatis memilih simpul replika untuk dihapus. Pada saat melakukannya, ElastiCache mencoba untuk mempertahankan arsitektur Multi-AZ untuk grup replikasi Anda diikuti dengan mempertahankan replika dengan ketertinggalan replikasi terkecil dari primer.
> - Anda tidak dapat menghapus primer atau simpul primer di dalam grup replikasi. Jika Anda menentukan simpul primer untuk dihapus, operasi akan gagal dengan peristiwa kesalahan yang menunjukkan bahwa simpul primer telah dipilih untuk penghapusan.

Menggunakan AWS CLI

Untuk mengurangi jumlah replika dalam serpihan Redis, gunakan perintah `decrease-replica-count` dengan parameter berikut:

- `--replication-group-id` – Wajib. Mengidentifikasi grup replikasi yang ingin dikurangi jumlah replika di dalamnya.
- `--apply-immediately` atau `--no-apply-immediately` – Diperlukan. Menentukan apakah akan mengurangi jumlah replika dengan segera (`--apply-immediately`) atau pada jendela pemeliharaan berikutnya (`--no-apply-immediately`). Saat ini, `--no-apply-immediately` belum didukung.
- `--new-replica-count` – Opsional. Menentukan jumlah simpul replika yang Anda inginkan. Nilai dari `--new-replica-count` harus nilai yang valid dan kurang dari jumlah replika saat ini di dalam grup simpul. Untuk nilai minimum yang diizinkan, lihat Mengurangi jumlah replika dalam serpihan. Jika nilai `--new-replica-count` tidak memenuhi persyaratan ini, panggilan akan gagal.
- `--replicas-to-remove` – Opsional. Berisi daftar ID simpul yang menentukan simpul replika yang akan dihapus.
- `--replica-configuration` – Opsional. Memungkinkan Anda untuk menentukan jumlah replika dan Zona Ketersediaan untuk setiap grup simpul secara bebas. Gunakan parameter ini untuk grup Redis (mode klaster diaktifkan) tempat Anda ingin membuat konfigurasi setiap grup simpul secara terpisah.

`--replica-configuration` memiliki tiga anggota opsional:

- `NodeGroupId` – ID empat digit untuk grup simpul yang sedang dikonfigurasi. Untuk grup replikasi Redis (mode klaster dinonaktifkan), ID serpihan selalu `0001`. Untuk menemukan ID grup simpul (serpihan) Redis (mode klaster diaktifkan), lihat Menemukan ID serpihan.
- `NewReplicaCount` – Parameter opsional yang menentukan jumlah simpul replika yang Anda inginkan. Nilai dari `NewReplicaCount` harus nilai yang valid dan kurang dari jumlah replika saat ini di dalam grup simpul. Untuk nilai minimum yang diizinkan, lihat Mengurangi jumlah replika dalam serpihan. Jika nilai `NewReplicaCount` tidak memenuhi persyaratan ini, panggilan akan gagal.
- `PreferredAvailabilityZones` – Daftar string `PreferredAvailabilityZone` yang menentukan Availability Zone yang menjadi tempat simpul dari grup replikasi berada. Jumlah nilai `PreferredAvailabilityZone` harus sama dengan nilai `NewReplicaCount` ditambah 1 untuk memperhitungkan simpul primer. Jika anggota `--replica-configuration` ini dihilangkan, ElastiCache for Redis memilih Zona Ketersediaan untuk setiap replika baru.

> **Important**
>
> Anda harus menyertakan satu dan hanya satu dari parameter `--new-replica-count`, `--replicas-to-remove`, atau `--replica-configuration`.

Example

Contoh berikut menggunakan `--new-replica-count` untuk mengurangi jumlah replika di dalam grup replikasi `sample-repl-group` menjadi satu. Saat contoh ini selesai, terdapat satu replika di setiap grup simpul. Jumlah ini berlaku terlepas dari apakah ini adalah grup Redis (mode klaster dinonaktifkan) dengan grup simpul tunggal atau grup Redis (mode klaster diaktifkan) dengan beberapa grup simpul.

Untuk Linux, macOS, atau Unix:

```
aws elasticache decrease-replica-count
    --replication-group-id sample-repl-group \
    --new-replica-count 1 \
    --apply-immediately
```

Untuk Windows:

```
aws elasticache decrease-replica-count ^
    --replication-group-id sample-repl-group ^
    --new-replica-count 1 ^
    --apply-immediately
```

Contoh berikut mengurangi jumlah replika di dalam grup replikasi `sample-repl-group` dengan menghapus dua replika yang ditentukan (`0001` dan `0003`) dari grup simpul.

Untuk Linux, macOS, atau Unix:

```
aws elasticache decrease-replica-count \
    --replication-group-id sample-repl-group \
    --replicas-to-remove 0001,0003 \
    --apply-immediately
```

Untuk Windows:

```
aws elasticache decrease-replica-count ^
    --replication-group-id sample-repl-group ^
    --replicas-to-remove 0001,0003 \
    --apply-immediately
```

Contoh berikut menggunakan `--replica-configuration` untuk mengurangi jumlah replika dalam grup replikasi `sample-repl-group` dengan nilai yang ditentukan untuk dua grup simpul yang sudah ditentukan. Mengingat adanya beberapa grup simpul, ini adalah grup replikasi Redis (mode klaster diaktifkan). Saat menentukan `PreferredAvailabilityZones` opsional, jumlah Zona Ketersediaan yang tercantum harus sama dengan nilai dari `NewReplicaCount` ditambah 1. Pendekatan ini memperhitungkan simpul primer untuk grup yang diidentifikasi oleh `NodeGroupId`.

Untuk Linux, macOS, atau Unix:

```
aws elasticache decrease-replica-count \
    --replication-group-id sample-repl-group \
    --replica-configuration \
        NodeGroupId=0001,NewReplicaCount=1,PreferredAvailabilityZones=us-east-1a,us-
east-1c \
        NodeGroupId=0003,NewReplicaCount=2,PreferredAvailabilityZones=us-east-1a,us-
east-1b,us-east-1c \
    --apply-immediately
```

Untuk Windows:

```
aws elasticache decrease-replica-count ^
    --replication-group-id sample-repl-group ^
    --replica-configuration ^
        NodeGroupId=0001,NewReplicaCount=2,PreferredAvailabilityZones=us-east-1a,us-
east-1c ^
        NodeGroupId=0003,NewReplicaCount=3,PreferredAvailabilityZones=us-east-1a,us-
east-1b,us-east-1c \
    --apply-immediately
```

Untuk informasi lebih lanjut tentang menambah jumlah replika menggunakan CLI, lihat decrease-replica-count di Referensi Baris Perintah Amazon ElastiCache.

Menggunakan API ElastiCache

Untuk mengurangi jumlah replika di dalam serpihan Redis, gunakan tindakan `DecreaseReplicaCount` dengan parameter berikut:

- `ReplicationGroupId` – Wajib. Mengidentifikasi grup replikasi yang ingin dikurangi jumlah replika di dalamnya.
- `ApplyImmediately` – Wajib. Menentukan apakah akan mengurangi jumlah replika dengan segera (`ApplyImmediately=True`) atau pada jendela pemeliharaan berikutnya (`ApplyImmediately=False`). Saat ini, `ApplyImmediately=False` belum didukung.
- `NewReplicaCount` – Opsional. Menentukan jumlah simpul replika yang Anda inginkan. Nilai dari `NewReplicaCount` harus nilai yang valid dan kurang dari jumlah replika saat ini di dalam grup simpul. Untuk nilai minimum yang diizinkan, lihat Mengurangi jumlah replika dalam serpihan. Jika nilai `--new-replica-count` tidak memenuhi persyaratan ini, panggilan akan gagal.
- `ReplicasToRemove` – Opsional. Berisi daftar ID simpul yang menentukan simpul replika yang akan dihapus.
- `ReplicaConfiguration` – Opsional. Berisi daftar grup simpul yang memungkinkan Anda untuk menentukan jumlah replika dan Availability Zones untuk setiap grup simpul secara bebas. Gunakan parameter ini untuk grup Redis (mode klaster diaktifkan) tempat Anda ingin membuat konfigurasi setiap grup simpul secara terpisah.

  `ReplicaConfiguraion` memiliki tiga anggota opsional:

  - `NodeGroupId` – ID empat digit untuk grup simpul yang sedang dikonfigurasi. Untuk grup replikasi Redis (mode klaster dinonaktifkan), ID grup simpul selalu `0001`. Untuk menemukan ID grup simpul (serpihan) Redis (mode klaster diaktifkan), lihat Menemukan ID serpihan.

- `NewReplicaCount` – Jumlah replika yang Anda inginkan di dalam grup simpul ini pada akhir operasi ini. Nilai ini harus kurang dari jumlah replika saat ini hingga minimum 1 jika Multi-AZ diaktifkan atau hingga 0 jika Multi-AZ dengan Failover Otomatis tidak diaktifkan. Jika nilai ini tidak lebih kecil dari jumlah replika saat ini di dalam grup simpul, maka panggilan akan gagal dengan pengecualian.
- `PreferredAvailabilityZones` – Daftar string `PreferredAvailabilityZone` yang menentukan Availability Zone yang menjadi tempat simpul dari grup replikasi berada. Jumlah nilai `PreferredAvailabilityZone` harus sama dengan nilai `NewReplicaCount` ditambah 1 untuk memperhitungkan simpul primer. Jika anggota `ReplicaConfiguration` ini dihilangkan, ElastiCache for Redis memilih Zona Ketersediaan untuk setiap replika baru.

> **Important**
>
> Anda harus menyertakan satu dan hanya satu dari parameter `NewReplicaCount`, `ReplicasToRemove`, atau `ReplicaConfiguration`.

Example

Contoh berikut menggunakan `NewReplicaCount` untuk mengurangi jumlah replika di dalam grup replikasi `sample-repl-group` menjadi satu. Saat contoh ini selesai, terdapat satu replika di setiap grup simpul. Jumlah ini berlaku terlepas dari apakah ini adalah grup Redis (mode klaster dinonaktifkan) dengan grup simpul tunggal atau grup Redis (mode klaster diaktifkan) dengan beberapa grup simpul.

```
https://elasticache.us-west-2.amazonaws.com/
      ?Action=DecreaseReplicaCount
      &ApplyImmediately=True
      &NewReplicaCount=1
      &ReplicationGroupId=sample-repl-group
      &Version=2015-02-02
      &SignatureVersion=4
      &SignatureMethod=HmacSHA256
      &Timestamp=20150202T192317Z
      &X-Amz-Credential=<credential>
```

Contoh berikut mengurangi jumlah replika di dalam grup replikasi `sample-repl-group` dengan menghapus dua replika yang ditentukan (`0001` dan `0003`) dari grup simpul.

```
https://elasticache.us-west-2.amazonaws.com/
      ?Action=DecreaseReplicaCount
      &ApplyImmediately=True
      &ReplicasToRemove.ReplicaToRemove.1=0001
      &ReplicasToRemove.ReplicaToRemove.2=0003
      &ReplicationGroupId=sample-repl-group
      &Version=2015-02-02
      &SignatureVersion=4
      &SignatureMethod=HmacSHA256
      &Timestamp=20150202T192317Z
      &X-Amz-Credential=<credential>
```

Contoh berikut menggunakan `ReplicaConfiguration` untuk mengurangi jumlah replika dalam grup replikasi `sample-repl-group` dengan nilai yang ditentukan untuk dua grup simpul yang sudah ditentukan. Mengingat adanya beberapa grup simpul, ini adalah grup replikasi Redis (mode klaster diaktifkan). Saat menentukan `PreferredAvailabilityZones` opsional, jumlah Zona Ketersediaan yang tercantum harus sama dengan nilai dari `NewReplicaCount` ditambah 1. Pendekatan ini memperhitungkan simpul primer untuk grup yang diidentifikasi oleh `NodeGroupId`.

```
https://elasticache.us-west-2.amazonaws.com/
      ?Action=DecreaseReplicaCount
      &ApplyImmediately=True
      &ReplicaConfiguration.ConfigureShard.1.NodeGroupId=0001
      &ReplicaConfiguration.ConfigureShard.1.NewReplicaCount=1

 &ReplicaConfiguration.ConfigureShard.1.PreferredAvailabilityZones.PreferredAvailabilityZone.1=
east-1a

 &ReplicaConfiguration.ConfigureShard.1.PreferredAvailabilityZones.PreferredAvailabilityZone.2=
east-1c
      &ReplicaConfiguration.ConfigureShard.2.NodeGroupId=0003
      &ReplicaConfiguration.ConfigureShard.2.NewReplicaCount=2

 &ReplicaConfiguration.ConfigureShard.2.PreferredAvailabilityZones.PreferredAvailabilityZone.1=
east-1a

 &ReplicaConfiguration.ConfigureShard.2.PreferredAvailabilityZones.PreferredAvailabilityZone.2=
east-1b

 &ReplicaConfiguration.ConfigureShard.2.PreferredAvailabilityZones.PreferredAvailabilityZone.4=
east-1c
      &ReplicationGroupId=sample-repl-group
```

```
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &X-Amz-Credential=<credential>
```

Untuk informasi selengkapnya tentang cara menurunkan jumlah replika menggunakan API, lihat DecreaseReplicaCount di Referensi API Amazon ElastiCache.

Menambahkan replika baca, untuk grup replikasi Redis (Mode Klaster Dinonaktifkan)

Informasi pada topik berikut berlaku hanya untuk grup replikasi Redis (mode klaster dinonaktifkan).

Saat lalu lintas baca Anda meningkat, Anda mungkin ingin menyebarkan tugas baca itu ke lebih banyak simpul dan mengurangi kepadatan proses baca pada salah satu simpul. Pada topik ini, Anda dapat menemukan cara menambahkan replika baca ke klaster Redis (mode klaster dinonaktifkan).

Sebuah grup replikasi Redis (mode klaster dinonaktifkan) dapat memiliki maksimal lima replika baca. Jika Anda mencoba menambahkan replika baca ke grup replikasi yang sudah memiliki lima replika baca, maka operasi ini akan gagal.

Untuk informasi tentang menambahkan replika ke grup replikasi Redis (mode klaster diaktifkan), lihat hal berikut:

- Penskalaan klaster di Redis (Mode Klaster Diaktifkan)
- Menambah jumlah replika dalam serpihan

Anda dapat menambahkan replika baca ke klaster Redis (mode klaster dinonaktifkan) menggunakan konsol ElastiCache, AWS CLI, atau API ElastiCache.

Topik terkait

- Menambahkan simpul ke klaster
- Menambahkan replika baca ke grup replikasi (AWS CLI)
- Menambahkan replika baca ke grup replikasi menggunakan API

### Menambahkan replika baca ke grup replikasi (AWS CLI)

Untuk menambahkan replika baca ke grup replikasi Redis (mode klaster dinonaktifkan), gunakan perintah AWS CLI `create-cache-cluster`, dengan parameter `--replication-group-id` untuk menentukan grup replikasi yang akan ditambahkan klaster (simpul) tersebut.

Contoh berikut membuat klaster `my-read replica` dan menambahkannya ke grup replikasi `my-replication-group`. Jenis simpul, grup parameter, grup keamanan, jendela pemeliharaan, dan pengaturan lain untuk replika baca adalah sama seperti untuk simpul lain di `my-replication-group`.

Untuk Linux, macOS, atau Unix:

```
aws elasticache create-cache-cluster \
      --cache-cluster-id my-read-replica \
      --replication-group-id my-replication-group
```

Untuk Windows:

```
aws elasticache create-cache-cluster ^
      --cache-cluster-id my-read-replica ^
      --replication-group-id my-replication-group
```

Untuk informasi lain tentang menambahkan replika baca menggunakan CLI, lihat create-cache-cluster di Referensi Baris Perintah Amazon ElastiCache.

### Menambahkan replika baca ke grup replikasi menggunakan API

Untuk menambahkan replika baca ke grup replikasi Redis (mode klaster dinonaktifkan), gunakan operasi `CreateCacheCluster` ElastiCache, dengan parameter `ReplicationGroupId` untuk menentukan grup replikasi yang akan ditambahkan klaster (simpul) tersebut.

Contoh berikut membuat klaster `myReadReplica` dan menambahkannya ke grup replikasi `myReplicationGroup`. Jenis simpul, grup parameter, grup keamanan, jendela pemeliharaan, dan pengaturan lain untuk replika baca adalah sama seperti untuk simpul lain di `myReplicationGroup`.

```
https://elasticache.us-west-2.amazonaws.com/
      ?Action=CreateCacheCluster
      &CacheClusterId=myReadReplica
      &ReplicationGroupId=myReplicationGroup
```

```
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &X-Amz-Credential=<credential>
```

Untuk informasi lain tentang menambahkan replika baca menggunakan API, lihat CreateCacheCluster di Referensi API Amazon ElastiCache.

Menghapus replika baca, untuk grup replikasi Redis (Mode Klaster Dinonaktifkan)

Informasi pada topik berikut berlaku untuk grup replikasi Redis (mode klaster dinonaktifkan) saja.

Saat lalu lintas baca pada grup replikasi Redis Anda berubah, Anda mungkin ingin menambahkan atau menghapus replika baca. Menghapus simpul dari grup replikasi Redis (mode klaster dinonaktifkan) adalah sama seperti menghapus klaster saja, meskipun terdapat pembatasan:

- Anda tidak dapat menghapus primer dari grup replikasi. Jika Anda ingin menghapus primer, lakukan hal berikut:

  1. Naikkan replika baca menjadi primer. Untuk informasi tentang menaikkan replika baca menjadi primer, lihat Menaikkan replika baca menjadi primer, untuk grup replikasi Redis (mode klaster dinonaktifkan).
  2. Hapus primer lama. Untuk pembatasan dari metode ini, lihat poin berikutnya.
- Jika Multi-AZ diaktifkan pada grup replikasi, Anda tidak dapat menghapus replika baca terakhir dari grup replikasi tersebut. Dalam kasus ini, lakukan hal berikut:

  1. Ubah grup replikasi dengan menonaktifkan Multi-AZ. Untuk informasi selengkapnya, lihat Mengubah grup replikasi.
  2. Hapus replika baca.

Anda dapat menghapus replika baca dari grup replikasi Redis (mode klaster dinonaktifkan) menggunakan konsol ElastiCache, AWS CLI untuk ElastiCache, atau API ElastiCache.

Untuk petunjuk tentang menghapus klaster dari grup replikasi Redis, lihat yang berikut ini:

- Menggunakan AWS Management Console
- Menggunakan AWS CLI
- Menggunakan API ElastiCache

- Penskalaan klaster di Redis (Mode Klaster Diaktifkan)
- Mengurangi jumlah replika dalam serpihan

## Menaikkan replika baca menjadi primer, untuk grup replikasi Redis (mode klaster dinonaktifkan)

Informasi pada topik berikut berlaku untuk grup-grup replikasi Redis (mode klaster dinonaktifkan) saja.

Anda dapat menaikkan replika baca klaster Redis (mode klaster dinonaktifkan) ke primer menggunakan AWS Management Console, AWS CLI, atau API ElastiCache. Anda tidak dapat menaikkan replika baca ke primer jika Multi-AZ dengan Failover Otomatis diaktifkan pada grup replikasi Redis (mode klaster dinonaktifkan). Untuk menaikkan replika Redis (mode klaster dinonaktifkan) menjadi primer pada grup replikasi yang mengaktifkan Multi-AZ, lakukan hal berikut:

1. Ubah grup replikasi untuk menonaktifkan Multi-AZ (melakukan hal ini tidak mensyaratkan semua klaster Anda berada dalam Zona Ketersediaan yang sama). Untuk informasi selengkapnya, lihat Mengubah grup replikasi.
2. Naikkan replika baca menjadi primer.
3. Ubah grup replikasi untuk mengaktifkan kembali Multi-AZ.

Multi-AZ tidak tersedia pada grup replikasi yang menjalankan Redis 2.6.13 atau sebelumnya.

Menggunakan AWS Management Console

Prosedur berikut menggunakan konsol untuk menaikkan simpul replika menjadi primer.

Untuk menaikkan replika baca menjadi primer (konsol)

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Jika replika yang ingin dinaikkan adalah anggota grup replikasi Redis (mode klaster dinonaktifkan) yang mengaktifkan Multi-AZ, ubah grup replikasi untuk menonaktifkan Multi-AZ sebelum Anda melanjutkan. Untuk informasi selengkapnya, lihat Mengubah grup replikasi.
3. Pilih Redis, kemudian dari daftar klaster, pilih grup replikasi yang ingin diubah. Grup replikasi ini harus menjalankan mesin "Redis", bukan mesin "Redis Berklaster", dan harus memiliki dua atau lebih simpul.
4. Dari daftar simpul, pilih simpul replika yang ingin dinaikkan menjadi primer, kemudian untuk Tindakan, pilih Naikkan.
5. Pada kotak dialog Menaikkan Replika Baca, lakukan hal berikut:

a. Untuk Terapkan Segera, pilih Ya untuk menaikkan replika baca dengan segera, atau Tidak untuk menaikkannya pada jendela pemeliharaan klaster berikutnya.

b. Pilih Naikkan untuk menaikkan replika baca atau Batalkan untuk membatalkan operasi.

6. Jika klaster Multi-AZ diaktifkan sebelum Anda memulai proses kenaikan, tunggu hingga status grup replikasi menjadi tersedia, dan kemudian ubah klaster untuk mengaktifkan kembali Multi-AZ. Untuk informasi selengkapnya, lihat Mengubah grup replikasi.

Menggunakan AWS CLI

Anda tidak dapat menaikkan replika baca menjadi primer jika grup replikasi mengaktifkan Multi-AZ. Dalam beberapa kasus, replika yang ingin dinaikkan mungkin adalah anggota grup replikasi yang mengaktifkan Multi-AZ. Dalam kasus ini, Anda harus mengubah grup replikasi untuk menonaktifkan Multi-AZ sebelum Anda melanjutkan. Melakukan hal ini tidak mensyaratkan semua klaster Anda berada di dalam Zona Ketersediaan yang sama. Untuk informasi lain tentang mengubah grup replikasi, lihat Mengubah grup replikasi.

Perintah AWS CLI berikut mengubah grup replikasi `sample-repl-group`, membuat replika baca `my-replica-1` menjadi primer dalam grup replikasi.

Untuk Linux, macOS, atau Unix:

```
aws elasticache modify-replication-group \
   --replication-group-id sample-repl-group \
   --primary-cluster-id my-replica-1
```

Untuk Windows:

```
aws elasticache modify-replication-group ^
   --replication-group-id sample-repl-group ^
   --primary-cluster-id my-replica-1
```

Untuk informasi lain tentang mengubah grup replikasi, lihat modify-replication-group di Referensi Baris Perintah Amazon ElastiCache.

Menggunakan API ElastiCache

Anda tidak dapat menaikkan replika baca menjadi primer jika grup replikasi mengaktifkan Multi-AZ. Dalam beberapa kasus, replika yang ingin dinaikkan mungkin adalah anggota grup replikasi yang mengaktifkan Multi-AZ. Dalam kasus ini, Anda harus mengubah grup replikasi untuk menonaktifkan

Multi-AZ sebelum Anda melanjutkan. Melakukan hal ini tidak mensyaratkan semua klaster Anda berada di dalam Zona Ketersediaan yang sama. Untuk informasi lain tentang mengubah grup replikasi, lihat Mengubah grup replikasi.

Tindakan API ElastiCache berikut mengubah grup replikasi `myReplGroup`, membuat replika baca `myReplica-1` menjadi primer dalam grup replikasi.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=ModifyReplicationGroup
   &ReplicationGroupId=myReplGroup
   &PrimaryClusterId=myReplica-1
   &Version=2014-12-01
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20141201T220302Z
   &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
   &X-Amz-Date=20141201T220302Z
   &X-Amz-SignedHeaders=Host
   &X-Amz-Expires=20141201T220302Z
   &X-Amz-Credential=<credential>
   &X-Amz-Signature=<signature>
```

Untuk informasi lain tentang mengubah grup replikasi, lihat ModifyReplicationGroup di Referensi API Amazon ElastiCache.

## Mengelola pemeliharaan

Setiap klaster dan grup replikasi memiliki periode pemeliharaan mingguan yang perubahan sistemnya diterapkan. Jika Anda tidak menentukan jendela pemeliharaan yang diinginkan saat Anda membuat atau mengubah klaster atau grup replikasi, maka ElastiCache menetapkan jendela pemeliharaan 60 menit pada hari yang dipilih secara acak dalam seminggu.

Periode pemeliharaan 60 menit dipilih secara acak dari blok 8 jam waktu per wilayah. Tabel berikut mencantumkan blok waktu untuk tiap wilayah dari jendela waktu default yang ditetapkan. Anda dapat memilih jendela perawatan yang disukai di luar blok jendela pemeliharaan wilayah.

| Kode Wilayah | Nama wilayah | Jendela Pemeliharaan Wilayah |
|---|---|---|
| `ap-northeast-1` | Wilayah Asia Pasifik (Tokyo) | 13.00–21.00 UTC |
| `ap-northeast-2` | Wilayah Asia Pasifik (Seoul) | 12.00–20.00 UTC |

| Kode Wilayah | Nama wilayah | Jendela Pemeliharaan Wilayah |
|---|---|---|
| ap-northeast-3 | Wilayah Asia Pasifik (Osaka) | 12.00–20.00 UTC |
| ap-southeast-3 | Wilayah Asia Pasifik (Jakarta) | 14.00–22.00 UTC |
| ap-south-1 | Wilayah Asia Pasifik (Mumbai) | 17.30–01.30 UTC |
| ap-southeast-1 | Wilayah Asia Pasifik (Singapura) | 14.00–22.00 UTC |
| cn-north-1 | Wilayah Tiongkok (Beijing) | 14.00–22.00 UTC |
| cn-northwest-1 | Wilayah Tiongkok (Ningxia) | 14.00–22.00 UTC |
| ap-east-1 | Wilayah Asia Pacific (Hong Kong) | 13.00–21.00 UTC |
| ap-southeast-2 | Wilayah Asia Pasifik (Sydney) | 12.00–20.00 UTC |
| eu-west-3 | Wilayah EU (Paris) | 23.59–07.29 UTC |
| af-south-1 | Wilayah Afrika (Cape Town) | 13.00–21.00 UTC |
| eu-central-1 | Wilayah Eropa (Frankfurt) | 23.00–07.00 UTC |
| eu-west-1 | Wilayah Eropa (Irlandia) | 22.00–06.00 UTC |
| eu-west-2 | Wilayah Eropa (London) | 23.00–07.00 UTC |
| me-south-1 | Wilayah Timur Tengah (Bahrain) | 13.00–21.00 UTC |
| me-central-1 | Wilayah Timur Tengah (UEA) | 13.00–21.00 UTC |
| eu-south-1 | Wilayah Eropa (Milan) | 21.00–05.00 UTC |
| sa-east-1 | Wilayah Amerika Selatan (Sao Paulo) | 01.00–09.00 UTC |
| us-east-1 | Wilayah AS Timur (Virginia Utara) | 03.00–11.00 UTC |
| us-east-2 | Wilayah AS Timur (Ohio) | 04.00–12.00 UTC |
| us-gov-west-1 | Wilayah AWS GovCloud (US) | 06.00–14.00 UTC |

| Kode Wilayah | Nama wilayah | Jendela Pemeliharaan Wilayah |
|---|---|---|
| us-west-1 | Wilayah AS Barat (California Utara) | 06.00–14.00 UTC |
| us-west-2 | Wilayah AS Barat (Oregon) | 06.00–14.00 UTC |

Mengubah Jendela Pemeliharaan dari Klaster atau Grup Replikasi Anda

Periode pemeliharaan harus berada dalam waktu penggunaan terendah, sehingga kemungkinan memerlukan modifikasi dari waktu ke waktu. Anda dapat mengubah klaster atau grup replikasi Anda untuk menentukan rentang waktu hingga durasi 24 jam yang dalam waktu tersebut kegiatan pemeliharaan yang Anda minta akan dilakukan. Setiap perubahan klaster modifikasi yang Anda minta yang ditangguhkan atau ditunda terjadi dalam kurun waktu ini.

> **Note**
>
> Jika Anda ingin segera menerapkan modifikasi jenis simpul dan/atau peningkatan mesin menggunakan AWS Management Console pilih kotak Terapkan Segera. Jika tidak, modifikasi ini akan diterapkan selama masa pemeliharaan terjadwal berikutnya. Untuk menggunakan API, lihat modify-replication-group atau modify-cache-cluster.

Informasi selengkapnya

Untuk informasi tentang jendela pemeliharaan dan penggantian simpul, lihat yang berikut ini:

- Pemeliharaan ElastiCache—FAQ tentang pemeliharaan dan penggantian simpul
- Mengganti simpul—Mengelola penggantian simpul
- Mengubah grup replikasi—Mengubah jendela pemeliharaan grup replikasi

## Mengonfigurasi parameter mesin menggunakan grup parameter

Amazon ElastiCache menggunakan parameter untuk mengontrol properti runtime simpul dan klaster Anda. Secara umum, versi mesin yang lebih baru mencakup parameter tambahan untuk mendukung fungsionalitas yang lebih baru. Untuk tabel parameter, lihat Parameter spesifik Redis.

Seperti yang Anda harapkan, beberapa nilai parameter, seperti `maxmemory`, ditentukan oleh mesin dan jenis simpul. Untuk tabel nilai parameter berdasarkan jenis simpul, lihat Parameter khusus jenis simpul Redis.

Topik

- Manajemen parameter
- Tingkat grup parameter cache
- Membuat grup parameter
- Membuat daftar grup parameter berdasarkan nama
- Membuat daftar nilai grup parameter
- Mengubah grup parameter
- Menghapus grup parameter
- Parameter spesifik Memcached
- Parameter spesifik Redis

## Manajemen parameter

Parameter dikelompokkan bersama ke grup parameter bernama untuk manajemen parameter yang lebih mudah. Grup parameter mewakili kombinasi nilai tertentu untuk parameter yang diteruskan ke perangkat lunak mesin selama penyalaan mesin. Nilai ini menentukan cara mesin berproses pada setiap simpul berperilaku pada saat runtime. Nilai parameter pada grup parameter tertentu berlaku untuk semua simpul yang terkait dengan grup tersebut, terlepas dari asal klaster tersebut.

Untuk menyempurnakan kinerja klaster, Anda dapat mengubah beberapa nilai parameter atau mengubah grup parameter klaster.

- Anda tidak dapat mengubah atau menghapus grup parameter default. Jika membutuhkan nilai parameter khusus, Anda harus membuat grup parameter khusus.
- Keluarga grup parameter dan klaster yang menjadi destinasi penetapan harus kompatibel. Misalnya, jika klaster menjalankan Redis versi 3.2.10, Anda hanya dapat menggunakan grup parameter, default atau khusus, dari keluarga Redis3.2.
- Jika Anda mengubah grup parameter klaster, nilai untuk setiap parameter yang dapat diubah bersyarat harus sama di kedua grup parameter baru dan saat ini.
- Saat Anda mengubah parameter klaster, perubahan diterapkan pada klaster baik dengan segera atau, dengan pengecualian yang dicatat berikut, setelah simpul klaster di-boot ulang. Ini berlaku terlepas dari apakah Anda mengubah grup parameter klaster itu sendiri atau nilai parameter di dalam grup parameter klaster. Untuk menentukan waktu penerapan perubahan parameter tertentu, lihat kolom Perubahan Berlaku dalam tabel untuk Parameter spesifik Redis.

  Untuk informasi selengkapnya, lihat Boot ulang simpul.

> **Perubahan parameter Redis (Mode Klaster Diaktifkan)**
>
> Jika Anda membuat perubahan parameter berikut pada klaster Redis (mode klaster diaktifkan), ikuti langkah-langkah berikutnya.
>
> - activerehashing
> - databases
>
> 1. Buat cadangan manual klaster Anda. Lihat Mengambil cadangan manual.
> 2. Menghapus klaster Redis (mode klaster diaktifkan). Lihat Menghapus klaster.

3. Pulihkan klaster menggunakan grup parameter yang sudah diubah dan cadangkan untuk menyemai klaster baru. Lihat Melakukan pemulihan dari cadangan ke dalam cache baru.

Perubahan parameter lain tidak memerlukan ini.

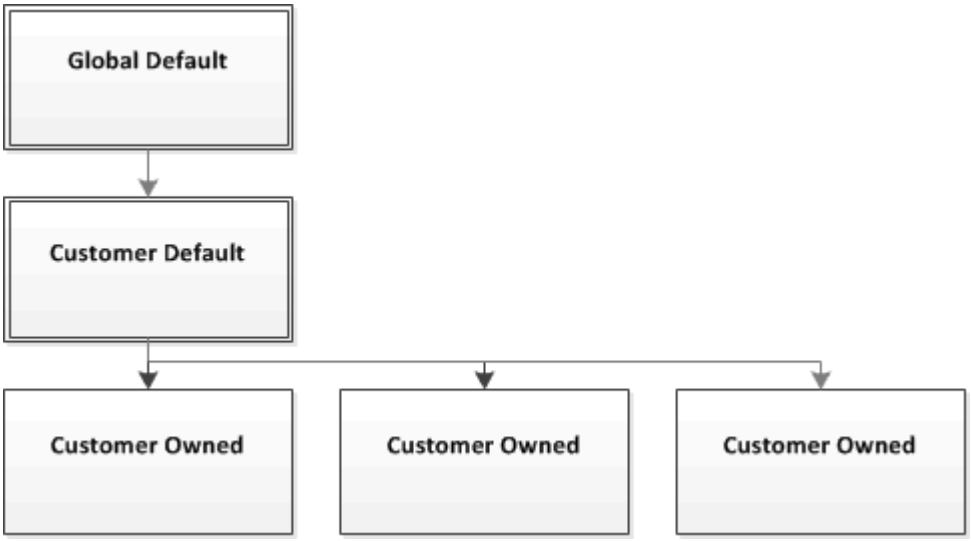
- Anda dapat mengaitkan grup parameter dengan penyimpanan data global Redis. Penyimpanan data global adalah kumpulan dari satu atau beberapa klaster yang menjangkau Wilayah AWS. Dalam kasus ini, grup parameter dibagi oleh semua klaster yang membentuk penyimpanan data global. Perubahan apa pun ke grup parameter klaster primer akan direplikasi ke semua klaster lain di penyimpanan data global. Untuk informasi selengkapnya, lihat Replikasi di seluruh Wilayah AWS menggunakan penyimpanan data global.

  Anda dapat memeriksa apakah grup parameter bagian dari penyimpanan data global dengan melihat di lokasi berikut:

  - Pada konsol ElastiCache di halaman Grup Parameter, atribut Global ya/tidak
  - Ya/tidak pada properti `IsGlobal` operasi API CacheParameterGroup

## Tingkat grup parameter cache

Amazon ElastiCache memiliki tiga tingkat grup parameter cache seperti yang ditunjukkan berikut.




Tingkat grup parameter Amazon ElastiCache

Default Global

Grup parameter root tingkat atas untuk semua pelanggan Amazon ElastiCache di wilayah tersebut.

Grup parameter cache default global:

- Disimpan untuk ElastiCache dan tidak tersedia untuk pelanggan.

Default Pelanggan

Salinan grup parameter cache Default Global yang dibuat untuk penggunaan pelanggan.

Grup parameter cache Default Pelanggan:

- Dibuat dan dimiliki oleh ElastiCache.
- Tersedia bagi pelanggan untuk digunakan sebagai grup parameter cache untuk setiap klaster yang menjalankan versi mesin yang didukung oleh grup parameter cache ini.
- Tidak dapat diedit oleh pelanggan.

Milik Pelanggan

Salinan grup parameter cache Default Pelanggan. Grup parameter cache Milik Pelanggan dibuat setiap kali pelanggan membuat grup parameter cache.

Grup parameter cache Milik Pelanggan:

- Dibuat dan dimiliki oleh pelanggan.
- Dapat ditetapkan untuk semua klaster pelanggan yang kompatibel.
- Dapat diubah oleh pelanggan untuk membuat grup parameter cache khusus.

  Tidak semua nilai parameter dapat diubah. Untuk informasi selengkapnya, lihat Parameter spesifik Redis.

## Membuat grup parameter

Anda perlu membuat grup parameter baru jika ada satu atau beberapa nilai parameter yang ingin Anda ubah dari nilai default. Anda dapat membuat grup parameter menggunakan konsol ElastiCache, AWS CLI, atau API ElastiCache.

Membuat grup parameter (Konsol)

Prosedur berikut menunjukkan cara membuat grup parameter menggunakan konsol ElastiCache.

Untuk membuat grup parameter menggunakan konsol ElastiCache

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Untuk melihat daftar semua grup parameter yang tersedia, pilih Grup Parameter di panel navigasi sebelah kiri.
3. Untuk membuat grup parameter, pilih Buat Grup Parameter.

   Layar Buat Grup Parameter akan muncul.
4. Dari daftar Keluarga, pilih grup parameter yang akan menjadi templat untuk grup parameter Anda.

   Keluarga grup parameter, misalnya redis3.2, menentukan parameter aktual dalam grup parameter Anda dan nilai awalnya. Keluarga grup parameter harus serupa dengan mesin dan versi klaster.
5. Di kotak Nama, ketik nama unik untuk grup parameter ini.

Saat membuat klaster atau mengubah grup parameter klaster, Anda akan memilih grup parameter berdasarkan namanya. Oleh karena itu, kami merekomendasikan agar namanya informatif dan dapat mengidentifikasi keluarga grup parameter.

Batasan penamaan grup parameter adalah sebagai berikut:

- Harus dimulai dengan huruf ASCII.
- Hanya dapat berisi huruf ASCII, angka, dan tanda hubung.
- Memiliki panjang 1-255 karakter.
- Tidak dapat berisi dua tanda hubung berturut-turut.
- Tidak boleh diakhiri dengan sebuah tanda hubung.

6. Di kotak Deskripsi, masukkan deskripsi untuk grup parameter yang baru.
7. Untuk membuat grup parameter, pilih Buat.

   Untuk mengakhiri proses tanpa membuat grup parameter, pilih Batalkan.
8. Ketika grup parameter dibuat, ia akan memiliki nilai default keluarga. Untuk mengubah nilai default Anda harus mengubah grup parameter. Untuk informasi selengkapnya, lihat Mengubah grup parameter.

### Membuat grup parameter (AWS CLI)

Untuk membuat grup parameter menggunakan AWS CLI, gunakan perintah `create-cache-parameter-group` dengan parameter ini.

- `--cache-parameter-group-name` — Nama grup parameter.

  Batasan penamaan grup parameter adalah sebagai berikut:
  - Harus dimulai dengan huruf ASCII.
  - Hanya dapat berisi huruf ASCII, angka, dan tanda hubung.
  - Memiliki panjang 1-255 karakter.
  - Tidak dapat berisi dua tanda hubung berturut-turut.
  - Tidak boleh diakhiri dengan sebuah tanda hubung.
- `--cache-parameter-group-family` — Keluarga mesin dan versi untuk grup parameter.
- `--description` — Deskripsi yang diberikan pengguna untuk grup parameter.

Example

Contoh berikut membuat grup parameter bernama myRed28 menggunakan keluarga redis2.8 sebagai templat.

Untuk Linux, macOS, atau Unix:

```
aws elasticache create-cache-parameter-group \
    --cache-parameter-group-name myRed28  \
    --cache-parameter-group-family redis2.8 \
    --description "My first parameter group"
```

Untuk Windows:

```
aws elasticache create-cache-parameter-group ^
    --cache-parameter-group-name myRed28  ^
    --cache-parameter-group-family redis2.8 ^
    --description "My first parameter group"
```

Output dari perintah ini akan terlihat seperti ini.

```
{
    "CacheParameterGroup": {
        "CacheParameterGroupName": "myRed28",
        "CacheParameterGroupFamily": "redis2.8",
        "Description": "My first parameter group"
    }
}
```

Ketika grup parameter dibuat, ia akan memiliki nilai default keluarga. Untuk mengubah nilai default Anda harus mengubah grup parameter. Untuk informasi selengkapnya, lihat Mengubah grup parameter.

Untuk informasi selengkapnya, lihat `create-cache-parameter-group`.

Membuat grup parameter (API ElastiCache)

Untuk membuat grup parameter menggunakan API ElastiCache, gunakan tindakan `CreateCacheParameterGroup` dengan parameter ini.

- `ParameterGroupName` — Nama grup parameter.

Batasan penamaan grup parameter adalah sebagai berikut:

- Harus dimulai dengan huruf ASCII.
- Hanya dapat berisi huruf ASCII, angka, dan tanda hubung.
- Memiliki panjang 1-255 karakter.
- Tidak dapat berisi dua tanda hubung berturut-turut.
- Tidak boleh diakhiri dengan sebuah tanda hubung.

- `CacheParameterGroupFamily` — Keluarga mesin dan versi untuk grup parameter. Misalnya, `redis2.8`.
- `Description` — Deskripsi yang diberikan pengguna untuk grup parameter.

Example

Contoh berikut membuat grup parameter bernama myRed28 menggunakan keluarga redis2.8 sebagai templat.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=CreateCacheParameterGroup
   &CacheParameterGroupFamily=redis2.8
   &CacheParameterGroupName=myRed28
   &Description=My%20first%20parameter%20group
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &Version=2015-02-02
   &X-Amz-Credential=<credential>
```

Respons dari tindakan ini akan terlihat seperti ini.

```
<CreateCacheParameterGroupResponse xmlns="http://elasticache.amazonaws.com/
doc/2013-06-15/">
  <CreateCacheParameterGroupResult>
    <CacheParameterGroup>
      <CacheParameterGroupName>myRed28</CacheParameterGroupName>
      <CacheParameterGroupFamily>redis2.8</CacheParameterGroupFamily>
      <Description>My first parameter group</Description>
    </CacheParameterGroup>
  </CreateCacheParameterGroupResult>
  <ResponseMetadata>
```

```
    <RequestId>d8465952-af48-11e0-8d36-859edca6f4b8</RequestId>
  </ResponseMetadata>
</CreateCacheParameterGroupResponse>
```

Ketika grup parameter dibuat, ia akan memiliki nilai default keluarga. Untuk mengubah nilai default Anda harus mengubah grup parameter. Untuk informasi selengkapnya, lihat Mengubah grup parameter.

Untuk informasi selengkapnya, lihat `CreateCacheParameterGroup`.

## Membuat daftar grup parameter berdasarkan nama

Anda dapat membuat daftar grup parameter menggunakan konsol ElastiCache, AWS CLI, atau API ElastiCache.

### Membuat daftar grup parameter berdasarkan nama (Konsol)

Prosedur berikut menunjukkan cara membuat grup parameter menggunakan konsol ElastiCache.

Untuk membuat daftar grup parameter menggunakan konsol ElastiCache

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Untuk melihat daftar semua grup parameter yang tersedia, pilih Grup Parameter di panel navigasi sebelah kiri.

### Membuat daftar grup parameter berdasarkan nama (AWS CLI)

Untuk menghasilkan daftar grup parameter menggunakan AWS CLI, gunakan perintah `describe-cache-parameter-groups`. Jika Anda memberikan nama grup parameter, hanya grup parameter tersebut yang akan dicantumkan. Jika Anda memberikan nama grup parameter, hanya grup parameter hingga `--max-records` yang akan dicantumkan. Dalam kedua kasus, nama, keluarga, dan deskripsi grup parameter akan dicantumkan.

Example

Kode sampel berikut membuat daftar grup parameter myRed28.

Untuk Linux, macOS, atau Unix:

```
aws elasticache describe-cache-parameter-groups \
    --cache-parameter-group-name myRed28
```

Untuk Windows:

```
aws elasticache describe-cache-parameter-groups ^
    --cache-parameter-group-name myRed28
```

Output dari perintah ini akan terlihat seperti ini, mencantumkan nama, keluarga, dan deskripsi untuk grup parameter.

```
{
    "CacheParameterGroups": [
     {
         "CacheParameterGroupName": "myRed28",
         "CacheParameterGroupFamily": "redis2.8",
         "Description": "My first parameter group"
     }
    ]
}
```

Example

Kode sampel berikut akan membuat daftar grup parameter myRed56 untuk grup parameter yang berjalan pada mesin Redis versi 5.0.6 ke atas. Jika grup parameter adalah bagian dari Replikasi di seluruh Wilayah AWS menggunakan penyimpanan data global, nilai properti `IsGlobal` yang dikembalikan dalam output akan berupa `Yes`.

Untuk Linux, macOS, atau Unix:

```
aws elasticache describe-cache-parameter-groups \
    --cache-parameter-group-name myRed56
```

Untuk Windows:

```
aws elasticache describe-cache-parameter-groups ^
    --cache-parameter-group-name myRed56
```

Output dari perintah ini akan terlihat seperti ini, mencantumkan nama, keluarga, isGlobal, dan deskripsi untuk grup parameter.

```
{
    "CacheParameterGroups": [
     {
         "CacheParameterGroupName": "myRed56",
         "CacheParameterGroupFamily": "redis5.0",
         "Description": "My first parameter group",
         "IsGlobal": "yes"
     }
    ]
}
```

Example

Kode sampel berikut membuat daftar hingga 10 grup parameter.

```
aws elasticache describe-cache-parameter-groups --max-records 10
```

Output JSON dari perintah ini akan terlihat seperti ini, mencantumkan nama, keluarga, deskripsi dan, pada redis5.6 menunjukkan apakah grup parameter bagian dari penyimpanan data global (isGlobal), untuk setiap grup parameter.

```
{
    "CacheParameterGroups": [
        {
            "CacheParameterGroupName": "custom-redis32",
            "CacheParameterGroupFamily": "redis3.2",
            "Description": "custom parameter group with reserved-memory > 0"
        },
        {
            "CacheParameterGroupName": "default.memcached1.4",
            "CacheParameterGroupFamily": "memcached1.4",
            "Description": "Default parameter group for memcached1.4"
        },
        {
            "CacheParameterGroupName": "default.redis2.6",
            "CacheParameterGroupFamily": "redis2.6",
            "Description": "Default parameter group for redis2.6"
        },
        {
            "CacheParameterGroupName": "default.redis2.8",
            "CacheParameterGroupFamily": "redis2.8",
            "Description": "Default parameter group for redis2.8"
        },
        {
            "CacheParameterGroupName": "default.redis3.2",
            "CacheParameterGroupFamily": "redis3.2",
            "Description": "Default parameter group for redis3.2"
        },
        {
            "CacheParameterGroupName": "default.redis3.2.cluster.on",
            "CacheParameterGroupFamily": "redis3.2",
            "Description": "Customized default parameter group for redis3.2 with
 cluster mode on"
        },
```

```
        {
            "CacheParameterGroupName": "default.redis5.6.cluster.on",
            "CacheParameterGroupFamily": "redis5.0",
            "Description": "Customized default parameter group for redis5.6 with
 cluster mode on",
            "isGlobal": "yes"
        },
    ]
}
```

Untuk informasi selengkapnya, lihat [describe-cache-parameter-groups](#).

Membuat daftar grup parameter berdasarkan nama (API ElastiCache)

Untuk membuat daftar grup parameter menggunakan API ElastiCache, gunakan tindakan `DescribeCacheParameterGroups`. Jika Anda memberikan nama grup parameter, hanya grup parameter tersebut yang akan dicantumkan. Jika Anda memberikan nama grup parameter, hanya grup parameter hingga `MaxRecords` yang akan dicantumkan. Dalam kedua kasus, nama, keluarga, dan deskripsi grup parameter akan dicantumkan.

Example

Kode sampel berikut membuat daftar hingga 10 grup parameter.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeCacheParameterGroups
   &MaxRecords=10
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &Version=2015-02-02
   &X-Amz-Credential=<credential>
```

Respons dari tindakan ini akan terlihat seperti ini, mencantumkan nama, keluarga, deskripsi dan, dalam kasus redis5.6 jika grup parameter milik penyimpanan data global (isGlobal), untuk setiap grup parameter.

```
<DescribeCacheParameterGroupsResponse xmlns="http://elasticache.amazonaws.com/
doc/2013-06-15/">
  <DescribeCacheParameterGroupsResult>
    <CacheParameterGroups>
      <CacheParameterGroup>
```

```
        <CacheParameterGroupName>myRedis28</CacheParameterGroupName>
        <CacheParameterGroupFamily>redis2.8</CacheParameterGroupFamily>
        <Description>My custom Redis 2.8 parameter group</Description>
      </CacheParameterGroup>
      <CacheParameterGroup>
        <CacheParameterGroupName>myMem14</CacheParameterGroupName>
        <CacheParameterGroupFamily>memcached1.4</CacheParameterGroupFamily>
        <Description>My custom Memcached 1.4 parameter group</Description>
      </CacheParameterGroup>
       <CacheParameterGroup>
        <CacheParameterGroupName>myRedis56</CacheParameterGroupName>
        <CacheParameterGroupFamily>redis5.0</CacheParameterGroupFamily>
        <Description>My custom redis 5.6 parameter group</Description>
        <isGlobal>yes</isGlobal>
      </CacheParameterGroup>
    </CacheParameterGroups>
  </DescribeCacheParameterGroupsResult>
  <ResponseMetadata>
    <RequestId>3540cc3d-af48-11e0-97f9-279771c4477e</RequestId>
  </ResponseMetadata>
</DescribeCacheParameterGroupsResponse>
```

Example

Kode sampel berikut membuat daftar grup parameter myRed28.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeCacheParameterGroups
   &CacheParameterGroupName=myRed28
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &Version=2015-02-02
   &X-Amz-Credential=<credential>
```

Respons dari tindakan ini akan terlihat seperti ini, mencantumkan nama, keluarga, dan deskripsi.

```
<DescribeCacheParameterGroupsResponse xmlns="http://elasticache.amazonaws.com/
doc/2013-06-15/">
  <DescribeCacheParameterGroupsResult>
    <CacheParameterGroups>
      <CacheParameterGroup>
        <CacheParameterGroupName>myRed28</CacheParameterGroupName>
```

```
        <CacheParameterGroupFamily>redis2.8</CacheParameterGroupFamily>
        <Description>My custom Redis 2.8 parameter group</Description>
      </CacheParameterGroup>
    </CacheParameterGroups>
  </DescribeCacheParameterGroupsResult>
  <ResponseMetadata>
    <RequestId>3540cc3d-af48-11e0-97f9-279771c4477e</RequestId>
  </ResponseMetadata>
</DescribeCacheParameterGroupsResponse>
```

Example

Kode sampel berikut membuat daftar grup parameter myRed56.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeCacheParameterGroups
   &CacheParameterGroupName=myRed56
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &Version=2015-02-02
   &X-Amz-Credential=<credential>
```

Respons dari tindakan ini akan terlihat seperti ini, mencantumkan nama, keluarga, deskripsi, dan apakah grup parameter adalah bagian dari penyimpanan data global (isGlobal).

```
<DescribeCacheParameterGroupsResponse xmlns="http://elasticache.amazonaws.com/
doc/2013-06-15/">
  <DescribeCacheParameterGroupsResult>
    <CacheParameterGroups>
      <CacheParameterGroup>
        <CacheParameterGroupName>myRed56</CacheParameterGroupName>
        <CacheParameterGroupFamily>redis5.0</CacheParameterGroupFamily>
        <Description>My custom Redis 5.6 parameter group</Description>
        <isGlobal>yes</isGlobal>
      </CacheParameterGroup>
    </CacheParameterGroups>
  </DescribeCacheParameterGroupsResult>
  <ResponseMetadata>
    <RequestId>3540cc3d-af48-11e0-97f9-279771c4477e</RequestId>
  </ResponseMetadata>
</DescribeCacheParameterGroupsResponse>
```

Untuk informasi selengkapnya, lihat `DescribeCacheParameterGroups`.

## Membuat daftar nilai grup parameter

Anda dapat membuat daftar parameter dan nilai grup parameter menggunakan konsol ElastiCache, AWS CLI, atau API ElastiCache.

### Membuat daftar nilai grup parameter (Konsol)

Prosedur berikut menunjukkan cara membuat daftar parameter dan nilainya untuk grup parameter menggunakan konsol ElastiCache.

Untuk membuat daftar parameter dan nilai grup parameter menggunakan konsol ElastiCache

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Untuk melihat daftar semua grup parameter yang tersedia, pilih Grup Parameter di panel navigasi sebelah kiri.
3. Pilih grup parameter yang ingin Anda cantumkan parameter dan nilainya dengan memilih kotak di sebelah kiri nama grup parameter.

   Parameter dan nilainya akan tercantum di bagian bawah layar. Karena jumlah parameter, Anda mungkin harus menggulir ke atas/bawah untuk menemukan parameter yang diinginkan.

### Membuat daftar nilai grup parameter (AWS CLI)

Untuk membuat daftar parameter dan nilai grup parameter menggunakan AWS CLI, gunakan perintah `describe-cache-parameters`.

Example

Kode sampel berikut mencantumkan semua parameter dan nilai untuk grup parameter myRedis28.

Untuk Linux, macOS, atau Unix:

```
aws elasticache describe-cache-parameters \
    --cache-parameter-group-name myRedis28
```

Untuk Windows:

```
aws elasticache describe-cache-parameters ^
```

```
    --cache-parameter-group-name myRed28
```

Untuk informasi selengkapnya, lihat describe-cache-parameters.

Membuat daftar nilai grup parameter (API ElastiCache)

Untuk membuat daftar parameter dan nilai grup parameter menggunakan API ElastiCache, gunakan tindakan `DescribeCacheParameters`.

Example

Kode sampel berikut mencantumkan semua parameter untuk grup parameter myRed28.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeCacheParameters
   &CacheParameterGroupName=myRed28
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &Version=2015-02-02
   &X-Amz-Credential=<credential>
```

Respons dari tindakan ini akan terlihat seperti ini. Respons ini telah terpotong.

```
<DescribeCacheParametersResponse xmlns="http://elasticache.amazonaws.com/
doc/2013-06-15/">
  <DescribeCacheParametersResult>
    <CacheClusterClassSpecificParameters>
      <CacheNodeTypeSpecificParameter>
        <DataType>integer</DataType>
        <Source>system</Source>
        <IsModifiable>false</IsModifiable>
        <Description>The maximum configurable amount of memory to use to store items,
 in megabytes.</Description>
        <CacheNodeTypeSpecificValues>
          <CacheNodeTypeSpecificValue>
            <Value>1000</Value>
            <CacheClusterClass>cache.c1.medium</CacheClusterClass>
          </CacheNodeTypeSpecificValue>
          <CacheNodeTypeSpecificValue>
            <Value>6000</Value>
            <CacheClusterClass>cache.c1.xlarge</CacheClusterClass>
```

```
          </CacheNodeTypeSpecificValue>
          <CacheNodeTypeSpecificValue>
            <Value>7100</Value>
            <CacheClusterClass>cache.m1.large</CacheClusterClass>
          </CacheNodeTypeSpecificValue>
          <CacheNodeTypeSpecificValue>
            <Value>1300</Value>
            <CacheClusterClass>cache.m1.small</CacheClusterClass>
          </CacheNodeTypeSpecificValue>

...output omitted...

    </CacheClusterClassSpecificParameters>
  </DescribeCacheParametersResult>
  <ResponseMetadata>
    <RequestId>6d355589-af49-11e0-97f9-279771c4477e</RequestId>
  </ResponseMetadata>
</DescribeCacheParametersResponse>
```

Untuk informasi selengkapnya, lihat `DescribeCacheParameters`.

## Mengubah grup parameter

> **Important**
>
> Anda tidak dapat mengubah grup parameter default.

Anda dapat mengubah beberapa nilai parameter di grup parameter. Nilai parameter ini diterapkan ke klaster yang terkait dengan grup parameter. Untuk informasi selengkapnya tentang kapan perubahan nilai parameter diterapkan ke grup parameter, lihat Parameter spesifik Redis.

Mengubah grup parameter (Konsol)

Prosedur berikut menunjukkan cara mengubah nilai `cluster-enabled` parameter menggunakan konsol ElastiCache. Anda akan menggunakan prosedur yang sama untuk mengubah nilai parameter apa pun.

Untuk mengubah nilai parameter menggunakan konsol ElastiCache

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.

2. Untuk melihat daftar semua grup parameter yang tersedia, pilih Grup Parameter di panel navigasi sebelah kiri.
3. Pilih grup parameter yang ingin Anda ubah dengan memilih kotak di sebelah kiri nama grup parameter.

   Parameter dari grup parameter akan tercantum di bagian bawah layar. Anda mungkin perlu menelusuri daftar untuk melihat semua parameter.
4. Untuk mengubah satu atau beberapa parameter, pilih Edit Parameter.
5. Pilih Simpan Perubahan.
6. Untuk menemukan nama parameter yang Anda ubah, lihat Parameter spesifik Redis. Jika Anda memiliki klaster Redis (mode klaster dinonaktifkan) dan membuat perubahan pada parameter berikut, Anda harus melakukan boot ulang simpul pada klaster:

   - activerehashing
   - databases

   Untuk informasi selengkapnya, lihat Boot ulang simpul.

   > **Perubahan parameter Redis (Mode Klaster Diaktifkan)**
   >
   > Jika Anda membuat perubahan parameter berikut pada klaster Redis (mode klaster diaktifkan), ikuti langkah-langkah berikutnya.
   >
   > - activerehashing
   > - databases
   >
   > 1. Buat cadangan manual klaster Anda. Lihat Mengambil cadangan manual.
   > 2. Menghapus klaster Redis (mode klaster diaktifkan). Lihat Menghapus klaster.
   > 3. Pulihkan klaster menggunakan grup parameter yang sudah diubah dan cadangkan untuk menyemai klaster baru. Lihat Melakukan pemulihan dari cadangan ke dalam cache baru.
   >
   > Perubahan parameter lain tidak memerlukan ini.

### Mengubah grup parameter (AWS CLI)

Untuk mengubah nilai parameter menggunakan AWS CLI, gunakan perintah `modify-cache-parameter-group`.

#### Example

Untuk menemukan nama dan nilai yang diizinkan dari parameter yang ingin Anda ubah, lihat Parameter spesifik Redis

Kode sampel berikut menetapkan nilai dua parameter, reserved-memory-percent dan cluster-enabled pada grup parameter `myredis32-on-30`. Kita mengatur reserved-memory-percent ke `30` (30 persen) dan cluster-enabled ke `yes` sehingga grup parameter dapat digunakan dengan klaster Redis (mode klaster diaktifkan) (grup replikasi).

Untuk Linux, macOS, atau Unix:

```
aws elasticache modify-cache-parameter-group \
    --cache-parameter-group-name myredis32-on-30 \
    --parameter-name-values \
        ParameterName=reserved-memory-percent,ParameterValue=30 \
        ParameterName=cluster-enabled,ParameterValue=yes
```

Untuk Windows:

```
aws elasticache modify-cache-parameter-group ^
    --cache-parameter-group-name myredis32-on-30 ^
    --parameter-name-values ^
        ParameterName=reserved-memory-percent,ParameterValue=30 ^
        ParameterName=cluster-enabled,ParameterValue=yes
```

Output dari perintah ini akan terlihat seperti ini.

```
{
    "CacheParameterGroupName": "my-redis32-on-30"
}
```

Untuk informasi selengkapnya, lihat `modify-cache-parameter-group`.

Untuk menemukan nama parameter yang Anda ubah, lihat Parameter spesifik Redis.

Jika Anda memiliki klaster Redis (mode klaster dinonaktifkan) dan membuat perubahan pada parameter berikut, Anda harus melakukan boot ulang simpul pada klaster:

- activerehashing
- databases

Untuk informasi selengkapnya, lihat Boot ulang simpul.

> **Perubahan parameter Redis (Mode Klaster Diaktifkan)**
>
> Jika Anda membuat perubahan parameter berikut pada klaster Redis (mode klaster diaktifkan), ikuti langkah-langkah berikutnya.
>
> - activerehashing
> - databases
>
> 1. Buat cadangan manual klaster Anda. Lihat Mengambil cadangan manual.
> 2. Menghapus klaster Redis (mode klaster diaktifkan). Lihat Menghapus klaster.
> 3. Pulihkan klaster menggunakan grup parameter yang sudah diubah dan cadangkan untuk menyemai klaster baru. Lihat Melakukan pemulihan dari cadangan ke dalam cache baru.
>
> Perubahan parameter lain tidak memerlukan ini.

Mengubah grup parameter (API ElastiCache)

Untuk mengubah nilai parameter grup parameter menggunakan API ElastiCache, gunakan tindakan `ModifyCacheParameterGroup`.

Example

Untuk menemukan nama dan nilai yang diizinkan dari parameter yang ingin Anda ubah, lihat Parameter spesifik Redis

Kode sampel berikut menetapkan nilai dua parameter, reserved-memory-percent dan cluster-enabled pada grup parameter `myredis32-on-30`. Kita mengatur reserved-memory-percent ke `30` (30 persen) dan cluster-enabled ke `yes` sehingga grup parameter dapat digunakan dengan klaster Redis (mode klaster diaktifkan) (grup replikasi).

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=ModifyCacheParameterGroup
   &CacheParameterGroupName=myredis32-on-30
   &ParameterNameValues.member.1.ParameterName=reserved-memory-percent
   &ParameterNameValues.member.1.ParameterValue=30
   &ParameterNameValues.member.2.ParameterName=cluster-enabled
   &ParameterNameValues.member.2.ParameterValue=yes
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &Version=2015-02-02
   &X-Amz-Credential=<credential>
```

Untuk informasi selengkapnya, lihat `ModifyCacheParameterGroup`.

Jika Anda memiliki klaster Redis (mode klaster dinonaktifkan) dan membuat perubahan pada parameter berikut, Anda harus melakukan boot ulang simpul pada klaster:

- activerehashing
- databases

Untuk informasi selengkapnya, lihat Boot ulang simpul.

**Perubahan parameter Redis (Mode Klaster Diaktifkan)**

Jika Anda membuat perubahan parameter berikut pada klaster Redis (mode klaster diaktifkan), ikuti langkah-langkah berikutnya.

- activerehashing
- databases

1. Buat cadangan manual klaster Anda. Lihat Mengambil cadangan manual.
2. Hapus klaster Redis (mode klaster diaktifkan). Lihat Menghapus klaster.
3. Pulihkan klaster menggunakan grup parameter dan cadangan yang sudah diubah untuk melakukan seeding klaster baru. Lihat Melakukan pemulihan dari cadangan ke dalam cache baru.

Perubahan parameter lain tidak memerlukan ini.

## Menghapus grup parameter

Anda dapat membuat grup parameter menggunakan konsol ElastiCache, AWS CLI, atau API ElastiCache.

Anda tidak dapat menghapus grup parameter jika dikaitkan dengan setiap klaster. Anda juga tidak dapat menghapus grup parameter default.

### Menghapus grup parameter (Konsol)

Prosedur berikut menunjukkan cara membuat grup parameter menggunakan konsol ElastiCache.

Untuk membuat grup parameter menggunakan konsol ElastiCache

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Untuk melihat daftar semua grup parameter yang tersedia, pilih Grup Parameter di panel navigasi sebelah kiri.
3. Pilih grup parameter yang ingin Anda hapus dengan memilih kotak di sebelah kiri nama grup parameter.

   Tombol Hapus akan menjadi aktif.
4. Pilih Hapus.

   Layar konfirmasi Hapus Grup Parameter akan muncul.
5. Untuk menghapus grup parameter, pada layar konfirmasi Hapus Grup Parameter, pilih Hapus.

   Untuk mempertahankan grup parameter, pilih Batalkan.

### Menghapus grup parameter (AWS CLI)

Untuk menghapus grup parameter menggunakan AWS CLI, gunakan perintah `delete-cache-parameter-group`. Untuk grup parameter yang akan dihapus, grup parameter yang ditentukan oleh `--cache-parameter-group-name` tidak boleh memiliki klaster yang terkait dengannya, juga tidak dapat berupa grup parameter default.

Kode sampel berikut menghapus grup parameter myMem14.

Example

Untuk Linux, macOS, atau Unix:

```
aws elasticache delete-cache-parameter-group \
    --cache-parameter-group-name myRed28
```

Untuk Windows:

```
aws elasticache delete-cache-parameter-group ^
    --cache-parameter-group-name myRed28
```

Untuk informasi selengkapnya, lihat delete-cache-parameter-group.

Menghapus grup parameter (API ElastiCache)

Untuk menghapus grup parameter menggunakan API ElastiCache, gunakan tindakan `DeleteCacheParameterGroup`. Untuk grup parameter yang akan dihapus, grup parameter yang ditentukan oleh `CacheParameterGroupName` tidak boleh memiliki klaster yang terkait dengannya, juga tidak dapat berupa grup parameter default.

Example

Kode sampel berikut menghapus grup parameter myRed28.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DeleteCacheParameterGroup
   &CacheParameterGroupName=myRed28
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &Version=2015-02-02
   &X-Amz-Credential=<credential>
```

Untuk informasi selengkapnya, lihat DeleteCacheParameterGroup.

## Parameter spesifik Memcached

Jika Anda tidak menentukan grup parameter untuk klaster Redis, maka grup parameter default yang sesuai dengan versi mesin Anda akan digunakan. Anda tidak dapat mengubah nilai parameter dalam grup parameter default. Namun, Anda dapat membuat grup parameter kustom dan menetapkannya ke klaster Anda kapan saja. Untuk informasi selengkapnya, lihat Membuat grup parameter.

Topik



### Perubahan Memcached 1.6.17

Mulai Memcached 1.6.17, kami tidak lagi mendukung perintah administratif ini: `lru_crawler`, `lru`, dan `slabs`. Dengan perubahan ini, Anda tidak akan dapat mengaktifkan/menonaktifkan `lru_crawler` saat runtime melalui perintah. Aktifkan/nonaktifkan `lru_crawler` dengan memodifikasi grup parameter kustom Anda.

### Parameter yang ditambahkan dalam Memcached 1.6.6

Untuk Memcached 1.6.6, tidak ada parameter tambahan yang didukung.

Rangkaian grup parameter: memcached1.6

### Perubahan parameter Memcached 1.5.10

Untuk Memcached 1.5.10, parameter tambahan berikut didukung.

Rangkaian grup parameter: memcached1.5

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `no_modern` | Default: 1<br><br>Tipe: boolean<br><br>Dapat diubah: Ya<br><br>Nilai yang Diizinkan: 0,1<br><br>Perubahan Berlaku: Saat peluncuran | Alias untuk menonaktifkan perintah `slab_reassign` , `slab_automove` , `lru_crawler` , `lru_maintainer` , `maxconns_fast` . `No modern` juga menetapkan hash_algorithm untuk `jenkins` dan memungkinkan inlining ASCII VALUE. Berlaku untuk memcached 1.5 dan lebih tinggi. Untuk mengembalikan ke modern, Anda harus menonaktifkan parameter ini dan meluncurkannya kembali, yang secara otomatis akan mengaktifkan `slab_reassign` , `slab_automove` , `lru_crawler` , `lru_maintainer` , dan `maxconns_fast` .<br><br>**Note**<br>Nilai konfigurasi default untuk parameter ini telah diubah dari 0 ke 1 mulai 20 Agustus 2021. Nilai default yang diperbarui akan diterapkan secara otomatis oleh pengguna ElastiCache baru untuk setiap wilayah setelah 20 Agustus 2021. Pengguna ElastiCache yang sudah ada di wilayah sebelum |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| | | 20 Agustus 2021 perlu memodifikasi grup parameter kustom mereka secara manual untuk menerapkan perubahan baru ini. |
| `inline_ascii_resp` | Default: 0<br>Tipe: boolean<br>Dapat diubah: Ya<br>Nilai yang Diizinkan: 0,1<br>Perubahan Berlaku: Saat peluncuran | Menyimpan angka dari respons `VALUE`, di dalam item, menggunakan hingga 24 byte. Perlambatan kecil untuk `get` ASCII, `faster` ditetapkan. |

Untuk Memcached 1.5.10, parameter berikut dihapus.

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `expirezero_does_not_evict` | Default: 0<br>Tipe: boolean<br>Dapat diubah: Ya<br>Nilai yang Diizinkan: 0,1<br>Perubahan Berlaku: Saat peluncuran | Tidak lagi didukung di versi ini. |
| `modern` | Default: 1 | |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| | Tipe: boolean<br><br>Dapat Diubah: Ya (memerlukan peluncuran ulang jika diatur ke `no_modern`)<br><br>Nilai yang Diizinkan: 0,1<br><br>Perubahan Berlaku: Saat peluncuran | Tidak lagi didukung di versi ini. Dimulai dari versi ini, `no-modern` diaktifkan secara default pada setiap peluncuran atau peluncuran ulang. |

Parameter yang ditambahkan dalam Memcached 1.4.34

Untuk Memcached 1.4.34, tidak ada parameter tambahan yang didukung.

Rangkaian grup parameter: memcached1.4

Parameter yang ditambahkan dalam Memcached 1.4.33

Untuk Memcached 1.4.33, parameter tambahan berikut didukung.

Rangkaian grup parameter: memcached1.4

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `modern` | Default: diaktifkan<br><br>Tipe: boolean<br><br>Dapat diubah: Ya<br><br>Perubahan Berlaku: Saat peluncuran | Alias untuk beberapa fitur. Pengaktifan `modern` setara dengan mengaktifkan perintah berikut dan menggunakan algoritma hash murmur3: `slab_reassign` , `slab_automove` , `lru_crawler` , `lru_maintainer` , `maxconns_` |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| | | `fast` , dan `hash_algorithm=murmur3` . |
| `watch` | Default: diaktifkan<br><br>Tipe: boolean<br><br>Dapat diubah: Ya<br><br>Perubahan Berlaku: Segera<br><br>Log dapat dihapus jika pengguna mencapai batas `watcher_logbuf_size` dan `worker_logbuf_size` mereka. | Pengambilan, pengosongan, dan mutasi log. Ketika, misalnya, pengguna mengaktifkan `watch`, mereka dapat melihat log saat `get`, `set`, `delete`, atau `update` terjadi. |
| `idle_timeout` | Default: 0 (dinonaktifkan)<br><br>Tipe: integer<br><br>Dapat diubah: Ya<br><br>Perubahan Berlaku: Saat Peluncuran | Jumlah minimum detik saat klien akan diizinkan untuk idle sebelum diminta untuk tutup. Rentang nilai: 0 hingga 86400. |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `track_sizes` | Default: dinonaktifkan<br><br>Tipe: boolean<br><br>Dapat diubah: Ya<br><br>Perubahan Berlaku: Saat Peluncuran | Menunjukkan ukuran setiap grup slab yang telah dikonsumsi.<br><br>Pengaktifan `track_sizes` memungkinkan Anda menjalankan `stats sizes` tanpa perlu menjalankan `stats sizes_enable` . |
| `watcher_logbuf_size` | Default: 256 (KB)<br><br>Tipe: integer<br><br>Dapat diubah: Ya<br><br>Perubahan Berlaku: Saat Peluncuran | Perintah `watch` mengaktifkan logging stream untuk Memcached . Namun `watch` dapat menghapus log jika tingkat pengosongan, mutasi, atau pengambilan cukup tinggi untuk menyebabkan buffer logging menjadi penuh. Dalam situasi tersebut, pengguna dapat meningkatkan ukuran buffer untuk mengurangi kemungkinan kehilangan log. |
| `worker_logbuf_size` | Default: 64 (KB)<br><br>Tipe: integer<br><br>Dapat diubah: Ya<br><br>Perubahan Berlaku: Saat Peluncuran | Perintah `watch` mengaktifkan logging stream untuk Memcached . Namun `watch` dapat menghapus log jika tingkat pengosongan, mutasi, atau pengambilan cukup tinggi untuk menyebabkan logging buffer menjadi penuh. Dalam situasi tersebut, pengguna dapat meningkatkan ukuran buffer untuk mengurangi kemungkinan kehilangan log. |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `slab_chunk_max` | Default: 524288 (byte)<br>Tipe: integer<br>Dapat diubah: Ya<br>Perubahan Berlaku: Saat Peluncuran | Menentukan ukuran maksimum slab. Pengaturan ukuran slab lebih kecil membuat penggunaan memori lebih efisien. Item yang lebih besar dari `slab_chunk_max` akan dibagi menjadi beberapa slab. |
| `lru_crawler metadump [all\|1\|2\|3]` | Default: dinonaktifkan<br>Tipe: boolean<br>Dapat diubah: Ya<br>Perubahan Berlaku: Segera | jika lru_crawler diaktifkan, perintah ini menghapus semua kunci.<br>`all\|1\|2\|3` - semua slab, atau tentukan nomor slab tertentu |

Parameter yang ditambahkan dalam Memcached 1.4.24

Untuk Memcached 1.4.24, parameter tambahan berikut didukung.

Rangkaian grup parameter: memcached1.4

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `disable_flush_all` | Default: 0 (dinonaktifkan)<br>Tipe: boolean<br>Dapat diubah: Ya<br>Perubahan Berlaku: Saat peluncuran | Tambahkan parameter (`-F`) untuk menonaktifkan flush_all. Berguna jika Anda tidak ingin dapat menjalankan flush penuh pada instans produksi.<br>Nilai: 0, 1 (pengguna dapat melakukan `flush_all` jika nilai adalah 0). |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `hash_algorithm` | Default: jenkins<br>Tipe: string<br>Dapat diubah: Ya<br>Perubahan Berlaku: Saat peluncuran | Algoritma hash yang akan digunakan. Nilai yang diizinkan: murmur3 dan jenkins. |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `lru_crawler` | Default: 0 (dinonaktifkan)<br><br>Tipe: boolean<br><br>Dapat diubah: Ya<br><br>Perubahan Berlaku: Setelah pengaktifan ulang<br><br>**Note**<br>Anda dapat mengaktifkan `lru_crawler` untuk sementara pada saat runtime dari baris perintah. Untuk informasi selengkapnya, lihat kolom Deskripsi. | Membersihkan kelas slab item yang telah kedaluwarsa. Ini adalah proses berdampak rendah yang berjalan di latar belakang. Saat ini memerlukan inisiasi perayapan menggunakan perintah manual.<br><br>Untuk mengaktifkan sementara, jalankan `lru_crawler enable` di baris perintah.<br><br>`lru_crawler 1,3,5` merayapi kelas slab 1, 3, 5 dengan mencari item yang kedaluwarsa untuk ditambahkan ke daftar bebas.<br><br>Nilai: 0,1<br><br>**Note**<br>Pengaktifan `lru_crawler` pada baris perintah akan mengaktifkan perayap hingga dinonaktifkan pada baris perintah atau boot ulang berikutnya. Untuk mengaktifkan secara permanen, Anda harus mengubah nilai parameter. Untuk informasi selengkapnya, lihat Mengubah grup parameter . |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `lru_maintainer` | Default: 0 (dinonaktifkan)<br>Tipe: boolean<br>Dapat diubah: Ya<br>Perubahan Berlaku: Saat peluncuran | Sebuah thread latar belakang yang mengacak item di antara LRU seiring kapasitas tercapai. Nilai: 0, 1. |
| `expirezero_does_not_evict` | Default: 0 (dinonaktifkan)<br>Tipe: boolean<br>Dapat diubah: Ya<br>Perubahan Berlaku: Saat peluncuran | Ketika digunakan dengan `lru_maintainer` , menjadikan item yang memiliki waktu kedaluwarsa 0 tidak dapat dikosongkan.<br>**Warning**<br>Hal ini dapat memenuhi memori yang tersedia untuk item lainnya yang dapat dikosongkan.<br>Dapat diatur untuk mengabaikan `lru_maintainer` . |

Parameter yang ditambahkan dalam Memcached 1.4.14

Untuk Memcached 1.4.14, parameter tambahan berikut didukung.

Rangkaian grup parameter: memcached1.4

Parameter yang ditambahkan dalam Memcached 1.4.14

| Nama | Deskripsi |
| --- | --- |
| `config_max` | Jumlah maksimum entri konfigurasi ElastiCache. |
| `config_size_max` | Ukuran maksimum entri konfigurasi, dalam byte. |
| `hashpower_init` | Ukuran awal tabel hash ElastiCache, dinyatakan sebagai pangkat dua. Default-nya adalah 16 (2^16), atau 65536 kunci. |
| `maxconns_fast` | Mengubah cara permintaan koneksi baru ditangani ketika batas koneksi maksimum tercapai. Jika parameter ini diatur ke 0 (nol), koneksi baru ditambahkan ke antrean backlog dan akan menunggu sampai koneksi lain ditutup. Jika parameter diatur ke 1, ElastiCache mengirimkan kesalahan ke klien dan segera menutup koneksi. |

| Nama | Deskripsi |
| --- | --- |
| | |
| `slab_automove` | Menyesuaikan algoritma automove slab: Jika parameter ini diatur ke 0 (nol), algoritma automove dinonaktifkan. Jika diatur ke 1, ElastiCache membutuhkan pendekatan konservatif yang lambat untuk memindahk an slab secara otomatis. Jika diatur ke 2, ElastiCache akan memindahk an slab secara agresif setiap kali ada pengosongan. (Mode ini tidak direkomendasikan kecuali untuk tujuan pengujian.) |

| Nama | Deskripsi |
| --- | --- |
| `slab_reassign` | Mengaktifkan atau menonaktifkan penetapan ulang slab. Jika parameter ini diatur ke 1, Anda dapat menggunakan perintah "slab reassign" untuk secara manual menetapkan ulang memori. |

Parameter yang didukung Memcached 1.4.5

Rangkaian grup parameter: memcached1.4

Untuk Memcached 1.4.5, parameter tambahan berikut didukung.

Parameter yang ditambahkan dalam Memcached 1.4.5

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `backlog_queue_limit` | Default: 1024<br>Tipe: integer<br>Dapat diubah: Tidak | Batas antrean backlog. |
| `binding_protocol` | Default: otomatis<br>Tipe: string | Protokol pengikatan.<br>Nilai yang diizinkan adalah `ascii` dan `auto`. |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| | Dapat diubah: Ya<br><br>Perubahan Berlaku: Setelah pengaktifan ulang | Untuk panduan dalam mengubah nilai `binding_protocol` , lihat Mengubah grup parameter. |
| `cas_disabled` | Default: 0 (salah)<br><br>Jenis: Boolean<br><br>Dapat diubah: Ya<br><br>Perubahan Berlaku: Setelah pengaktifan ulang | Jika 1 (benar), operasi periksa dan atur (CAS) akan dinonaktifkan, dan item yang disimpan akan menggunakan 8 byte lebih sedikit dibandingkan dengan CAS diaktifkan. |
| `chunk_size` | Default: 48<br><br>Tipe: integer<br><br>Dapat diubah: Ya<br><br>Perubahan Berlaku: Setelah pengaktifan ulang | Jumlah minimum, dalam byte, ruang untuk mengalokasikan kunci, nilai, dan bendera item terkecil. |
| `chunk_size_growth_factor` | Default: 1,25<br><br>Tipe: Float<br><br>Dapat diubah: Ya<br><br>Perubahan Berlaku: Setelah pengaktifan ulang | Faktor pertumbuhan yang mengontrol ukuran setiap potongan Memcached berturut-turut; setiap potongan akan `chunk_size_growth_factor` kali lebih besar dari potongan sebelumnya. |
| `error_on_memory_exhausted` | Default: 0 (salah)<br><br>Jenis: Boolean<br><br>Dapat diubah: Ya<br><br>Perubahan Berlaku: Setelah pengaktifan ulang | Jika 1 (benar), ketika tidak ada lagi memori untuk menyimpan item, Memcached akan menampilkan kesalahan dan bukan mengosongkan item. |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| large_memory_pages | Default: 0 (salah)<br>Jenis: Boolean<br>Dapat diubah: Tidak | Jika 1 (benar), ElastiCache akan mencoba menggunakan halaman memori yang besar. |
| lock_down_paged_memory | Default: 0 (salah)<br>Jenis: Boolean<br>Dapat diubah: Tidak | Jika 1 (benar), ElastiCache akan mengunci semua memori yang di-paging. |
| max_item_size | Default: 1048576<br>Tipe: integer<br>Dapat diubah: Ya<br>Perubahan Berlaku: Setelah pengaktifan ulang | Ukuran, dalam byte, item terbesar yang dapat disimpan dalam klaster. |
| max_simultaneous_connections | Default: 65000<br>Tipe: integer<br>Dapat diubah: Tidak | Jumlah maksimum koneksi bersamaan. |
| maximize_core_file_limit | Default: 0 (salah)<br>Jenis: Boolean<br>Dapat diubah:<br>Perubahan Berlaku: Setelah pengaktifan ulang | Jika 1 (benar), ElastiCache akan memaksimalkan batas file inti. |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `memcached_connections_overhead` | Default: 100<br>Tipe: integer<br>Dapat diubah: Ya<br>Perubahan Berlaku: Setelah pengaktifan ulang | Jumlah memori yang akan disimpan untuk koneksi Memcached dan berbagai overhead lainnya. Untuk informasi tentang parameter ini, lihat Overhead koneksi Memcached. |
| `requests_per_event` | Default: 20<br>Tipe: integer<br>Dapat diubah: Tidak | Jumlah maksimum permintaan per peristiwa untuk koneksi tertentu. Batas ini diperlukan untuk mencegah kekurangan sumber daya. |

Overhead koneksi Memcached

Pada setiap simpul, memori yang tersedia untuk menyimpan item adalah total memori yang tersedia pada simpul tersebut (yang disimpan dalam parameter `max_cache_memory`) dikurangi memori yang digunakan untuk koneksi dan overhead lainnya (yang disimpan dalam parameter `memcached_connections_overhead`). Misalnya, sebuah simpul jenis `cache.m1.small` memiliki `max_cache_memory` sebesar 1.300 MB. Dengan nilai `memcached_connections_overhead` default 100 MB, proses Memcached akan memiliki 1.200 MB yang tersedia untuk menyimpan item.

Nilai default untuk parameter `memcached_connections_overhead` memenuhi sebagian besar kasus penggunaan; namun, jumlah alokasi yang diperlukan untuk overhead koneksi dapat bervariasi bergantung pada beberapa faktor, termasuk tingkat permintaan, ukuran muatan, dan jumlah koneksi.

Anda dapat mengubah nilai `memcached_connections_overhead` agar lebih sesuai dengan kebutuhan aplikasi Anda. Misalnya, peningkatan nilai parameter `memcached_connections_overhead` akan mengurangi jumlah memori yang tersedia untuk menyimpan item dan memberikan buffer yang lebih besar untuk overhead koneksi. Pengurangan nilai parameter `memcached_connections_overhead` akan memberi Anda lebih banyak memori untuk menyimpan item, tetapi dapat meningkatkan risiko penggunaan swap dan penurunan performa. Jika Anda melihat penggunaan swap dan penurunan performa, coba tingkatkan nilai parameter `memcached_connections_overhead`.

**Important**

Untuk jenis simpul `cache.t1.micro`, nilai untuk `memcached_connections_overhead` ditentukan sebagai berikut:

- Jika klaster Anda menggunakan grup parameter default, ElastiCache akan mengatur nilai untuk `memcached_connections_overhead` ke 13 MB.
- Jika klaster menggunakan grup parameter yang telah Anda buat sendiri, nilai `memcached_connections_overhead` dapat diatur ke nilai pilihan Anda.

Parameter khusus jenis simpul Memcached

Meskipun sebagian besar parameter memiliki nilai tunggal, beberapa parameter memiliki nilai yang berbeda-beda bergantung pada jenis simpul yang digunakan. Tabel berikut menunjukkan nilai default untuk parameter `max_cache_memory` dan `num_threads` untuk setiap jenis simpul. Nilai pada parameter ini tidak dapat diubah.

| Jenis simpul | max_cache_memory (dalam megabyte) | num_threads |
|---|---|---|
| cache.t1.micro | 213 | 1 |
| cache.t2.micro | 555 | 1 |
| cache.t2.small | 1588 | 1 |
| cache.t2.medium | 3301 | 2 |
| cache.t3.micro | 512 | 2 |
| cache.t3.small | 1402 | 2 |
| cache.t3.medium | 3364 | 2 |
| cache.t4g.micro | 512 | 2 |
| cache.t4g.small | 1402 | 2 |
| cache.t4g.medium | 3164 | 2 |

| Jenis simpul | max_cache_memory (dalam megabyte) | num_threads |
|---|---|---|
| cache.m1.small | 1301 | 1 |
| cache.m1.medium | 3350 | 1 |
| cache.m1.large | 7100 | 2 |
| cache.m1.xlarge | 14600 | 4 |
| cache.m2.xlarge | 33800 | 2 |
| cache.m2.2xlarge | 30412 | 4 |
| cache.m2.4xlarge | 68000 | 16 |
| cache.m3.medium | 2850 | 1 |
| cache.m3.large | 6200 | 2 |
| cache.m3.xlarge | 13600 | 4 |
| cache.m3.2xlarge | 28600 | 8 |
| cache.m4.large | 6573 | 2 |
| cache.m4.xlarge | 11496 | 4 |
| cache.m4.2xlarge | 30412 | 8 |
| cache.m4.4xlarge | 62234 | 16 |
| cache.m4.10xlarge | 158355 | 40 |
| cache.m5.large | 6537 | 2 |
| cache.m5.xlarge | 13248 | 4 |
| cache.m5.2xlarge | 26671 | 8 |
| cache.m5.4xlarge | 53516 | 16 |

| Jenis simpul | max_cache_memory (dalam megabyte) | num_threads |
|---|---|---|
| cache.m5.12xlarge | 160900 | 48 |
| cache.m5.24xlarge | 321865 | 96 |
| cache.m6g.large | 6537 | 2 |
| cache.m6g.xlarge | 13248 | 4 |
| cache.m6g.2xlarge | 26671 | 8 |
| cache.m6g.4xlarge | 53516 | 16 |
| cache.m6g.8xlarge | 107000 | 32 |
| cache.m6g.12xlarge | 160900 | 48 |
| cache.m6g.16xlarge | 214577 | 64 |
| cache.c1.xlarge | 6600 | 8 |
| cache.r3.large | 13800 | 2 |
| cache.r3.xlarge | 29100 | 4 |
| cache.r3.2xlarge | 59600 | 8 |
| cache.r3.4xlarge | 120600 | 16 |
| cache.r3.8xlarge | 120600 | 32 |
| cache.r4.large | 12590 | 2 |
| cache.r4.xlarge | 25652 | 4 |
| cache.r4.2xlarge | 51686 | 8 |
| cache.r4.4xlarge | 103815 | 16 |
| cache.r4.8xlarge | 208144 | 32 |

| Jenis simpul | max_cache_memory (dalam megabyte) | num_threads |
|---|---|---|
| cache.r4.16xlarge | 416776 | 64 |
| cache.r5.large | 13387 | 2 |
| cache.r5.xlarge | 26953 | 4 |
| cache.r5.2xlarge | 54084 | 8 |
| cache.r5.4xlarge | 108347 | 16 |
| cache.r5.12xlarge | 325400 | 48 |
| cache.r5.24xlarge | 650869 | 96 |
| cache.r6g.large | 13387 | 2 |
| cache.r6g.xlarge | 26953 | 4 |
| cache.r6g.2xlarge | 54084 | 8 |
| cache.r6g.4xlarge | 108347 | 16 |
| cache.r6g.8xlarge | 214577 | 32 |
| cache.r6g.12xlarge | 325400 | 48 |
| cache.r6g.16xlarge | 429154 | 64 |
| cache.c7gn.large | 3164 | 2 |
| cache.c7gn.xlarge | 6537 | 4 |
| cache.c7gn.2xlarge | 13248 | 8 |
| cache.c7gn.4xlarge | 26671 | 16 |
| cache.c7gn.8xlarge | 53516 | 32 |
| cache.c7gn.12xlarge | 325400 | 48 |

| Jenis simpul | max_cache_memory (dalam megabyte) | num_threads |
|---|---|---|
| cache.c7gn.16xlarge | 108347 | 64 |

> **Note**
>
> Semua instans T2 dibuat di Amazon Virtual Private Cloud (Amazon VPC).

## Parameter spesifik Redis

Jika Anda tidak menentukan grup parameter untuk klaster Redis, maka grup parameter default yang sesuai dengan versi mesin Anda akan digunakan. Anda tidak dapat mengubah nilai parameter dalam grup parameter default. Namun, Anda dapat membuat grup parameter kustom dan menetapkannya ke klaster Anda setiap saat asalkan nilai parameter yang dapat diubah secara bersyarat di kedua grup parameter sama. Untuk informasi selengkapnya, lihat Membuat grup parameter.

Topik



### Perubahan parameter Redis 7

Rangkaian grup parameter: redis7

Grup parameter default Redis 7 adalah sebagai berikut:

- `default.redis7` – Gunakan grup parameter ini, atau turunannya, untuk grup replikasi dan klaster Redis (mode klaster dinonaktifkan).

- `default.redis7.cluster.on` – Gunakan grup parameter ini, atau turunannya, untuk grup replikasi dan klaster Redis (mode klaster diaktifkan).

Parameter yang ditambahkan dalam Redis 7 adalah sebagai berikut.

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `cluster-allow-pubsubshard-when-down` | Nilai yang diizinkan: `yes`, `no`<br>Default: `yes`<br>Tipe: string<br>Dapat diubah: Ya<br>Perubahan berlaku: Segera di semua simpul dalam klaster. | Ketika diatur ke nilai default ya, memungkinkan simpul melayani lalu lintas serpihan pubsub saat klaster dalam keadaan nonaktif, asalkan klaster ini mengetahui bahwa dirinya memiliki slot. |
| `cluster-preferred-endpoint-type` | Nilai yang diizinkan: `ip`, `tls-dynamic`<br>Default: `tls-dynamic`<br>Tipe: string<br>Dapat diubah: Ya<br>Perubahan berlaku: Segera di semua simpul dalam klaster. | Nilai ini mengontrol titik akhir apa yang ditampilkan untuk permintaan MOVED/ASKING serta bidang titik akhir untuk `CLUSTER SLOTS` dan `CLUSTER SHARDS`. Ketika nilai diatur ke ip, simpul akan menyatakan alamat ip-nya. Ketika nilai diatur ke tls-dynamic, simpul akan menyatakan nama host saat encryption-in-transit diaktifkan dan jika tidak, alamat ip. |
| `latency-tracking` | Nilai yang diizinkan: `yes`, `no`<br>Default: no<br>Tipe: string<br>Dapat diubah: Ya | Ketika diatur ke ya, akan melacak latensi per perintah dan memungkinkan ekspor distribusi persentil melalui perintah statistik latensi `INFO`, dan distribusi latensi kumulatif (histogram) melalui perintah `LATENCY`. |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| | Perubahan berlaku: Segera di semua simpul dalam klaster. | |
| `hash-max-listpack-entries` | Nilai yang diizinkan: `0+`<br>Default: `512`<br>Tipe: integer<br>Dapat diubah: Ya<br>Perubahan berlaku: Segera di semua simpul dalam klaster. | Jumlah maksimum entri hash agar set data dikompresi. |
| `hash-max-listpack-value` | Nilai yang diizinkan: `0+`<br>Default: `64`<br>Tipe: integer<br>Dapat diubah: Ya<br>Perubahan berlaku: Segera di semua simpul dalam klaster. | Ambang batas entri hash terbesar agar set data dikompresi. |
| `zset-max-listpack-entries` | Nilai yang diizinkan: `0+`<br>Default: `128`<br>Tipe: integer<br>Dapat diubah: Ya<br>Perubahan berlaku: Segera di semua simpul dalam klaster. | Jumlah maksimum entri set yang diurutkan agar set data dikompresi. |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `zset-max-listpack-value` | Nilai yang diizinkan: `0+`<br>Default: `64`<br>Tipe: integer<br>Dapat diubah: Ya<br>Perubahan berlaku: Segera di semua simpul dalam klaster. | Ambang batas entri set terbesar yang diurutkan agar set data dikompresi. |

Parameter yang diubah dalam Redis 7 adalah sebagai berikut.

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `activerehashing` | Dapat diubah: no. Di Redis 7, parameter ini disembunyikan dan diaktifkan secara default. Untuk menonaktifkannya, Anda perlu membuat kasus dukungan. | Dapat dimodifikasi sebelumnya ya. |

Parameter yang dihapus dalam Redis 7 adalah sebagai berikut.

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `hash-max-ziplist-entries` | Nilai yang diizinkan: `0+`<br>Default: `512`<br>Tipe: integer<br>Dapat diubah: Ya | Gunakan `listpack` bukan `ziplist` untuk merepresentasikan pengenkodean hash kecil |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| | Perubahan berlaku: Segera di semua simpul dalam klaster. | |
| `hash-max-ziplist-value` | Nilai yang diizinkan: `0+`<br>Default: `64`<br>Tipe: integer<br>Dapat diubah: Ya<br>Perubahan berlaku: Segera di semua simpul dalam klaster. | Gunakan `listpack` bukan `ziplist` untuk merepresentasikan pengenkodean hash kecil |
| `zset-max-ziplist-entries` | Nilai yang diizinkan: `0+`<br>Default: `128`<br>Tipe: integer<br>Dapat diubah: Ya<br>Perubahan berlaku: Segera di semua simpul dalam klaster. | Gunakan `listpack` bukan `ziplist` untuk merepresentasikan pengenkodean hash kecil. |
| `zset-max-ziplist-value` | Nilai yang diizinkan: `0+`<br>Default: `64`<br>Tipe: integer<br>Dapat diubah: Ya<br>Perubahan berlaku: Segera di semua simpul dalam klaster. | Gunakan `listpack` bukan `ziplist` untuk merepresentasikan pengenkodean hash kecil. |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `list-max-ziplist-size` | Nilai yang diizinkan:<br>Default: `-2`<br>Tipe: integer<br>Dapat diubah: Ya<br>Perubahan berlaku: Segera di semua simpul dalam klaster. | Jumlah entri yang diizinkan per simpul daftar internal. |

Perubahan parameter Redis 6.x

Rangkaian grup parameter: redis6.x

Grup parameter default Redis 6.x adalah sebagai berikut:

- `default.redis6.x` – Gunakan grup parameter ini, atau turunannya, untuk grup replikasi dan klaster Redis (mode klaster dinonaktifkan).
- `default.redis6.x.cluster.on` – Gunakan grup parameter ini, atau turunannya, untuk grup replikasi dan klaster Redis (mode klaster diaktifkan).

> **Note**
>
> Dalam mesin Redis versi 6.2, ketika rangkaian simpul r6gd diperkenalkan untuk digunakan dengan Tingkatan data, hanya kebijakan max-memory noeviction, volatile-lru, dan allkeys-lru yang didukung dengan jenis simpul r6gd.

Untuk informasi selengkapnya, lihat ElastiCache for Redis versi 6.2 (ditingkatkan) dan ElastiCache for Redis versi 6.0 (ditingkatkan).

Parameter yang ditambahkan dalam Redis 6.x adalah sebagai berikut.

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `acl-pubsub-default (added in 6.2)` | Nilai yang diizinkan : `resetchannels` , `allchannels`<br><br>Default: `allchannels`<br><br>Tipe: string<br><br>Dapat diubah: Ya<br><br>Perubahan berlaku: Pengguna Redis yang ada yang terkait dengan klaster akan terus memiliki izin yang ada. Perbarui pengguna atau boot ulang klaster untuk memperbarui pengguna Redis yang ada. | Izin saluran pubsub default untuk pengguna ACL yang di-deploy ke klaster ini. |
| `cluster-allow-reads-when-down (added in 6.0)` | Default: no<br><br>Tipe: string<br><br>Dapat diubah: Ya<br><br>Perubahan berlaku: Segera di semua simpul dalam klaster | Ketika diatur ke ya, grup replikasi Redis (mode klaster diaktifkan) terus memproses perintah baca bahkan saat simpul tidak mampu mencapai kuorum primer.<br><br>Jika diatur ke default tidak, grup replikasi menolak semua perintah. Kami merekomendasikan untuk mengatur nilai ini ke ya jika Anda menggunakan klaster dengan kurang dari tiga grup simpul atau aplikasi Anda dapat dengan aman menangani pembacaan yang usang dengan aman. |
| `tracking-table-max-keys` | Default: 1.000.000<br><br>Tipe: angka | Untuk membantu caching sisi klien, Redis mendukung pelacakan kunci mana yang diakses klien tertentu. |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `(added in 6.0)` | Dapat diubah: Ya<br><br>Perubahan berlaku: Segera di semua simpul dalam klaster | Ketika kunci yang dilacak diubah, pesan invalidasi dikirim ke semua klien untuk memberitahukan bahwa nilai cache-nya tidak valid lagi. Nilai ini memungkinkan Anda menentukan batas atas tabel ini. Setelah nilai parameter ini terlampaui, klien mendapatk an pesan invalidasi secara acak. Nilai ini harus disetel untuk membatasi penggunaan memori sambil masih melacak kunci. Kunci juga diinvalidasi dalam kondisi memori rendah. |
| `acllog-max-len (added in 6.0)` | Default: 128<br><br>Tipe: angka<br><br>Dapat diubah: Ya<br><br>Perubahan berlaku: Segera di semua simpul dalam klaster | Nilai ini sesuai dengan jumlah maks entri di log ACL. |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `active-expire-effort (added in 6.0)` | Default: 1<br><br>Tipe: angka<br><br>Dapat diubah: Ya<br><br>Perubahan berlaku: Segera di semua simpul dalam klaster | Redis menghapus kunci yang telah melampaui time-to-live-nya dengan dua mekanisme. Di satu sisi, kunci diakses dan ditemukan akan kedaluwarsa. Di sisi lain, pekerjaan berkala mengambil sampel kunci dan membuat kunci yang telah melebihi time-to-live (TTL)-nya menjadi kedaluwarsa. Parameter ini menentukan jumlah upaya yang digunakan Redis untuk membuat item menjadi kedaluwarsa dalam pekerjaan berkala.<br><br>Nilai default 1 akan mencoba mencegah adanya lebih dari 10 persen kunci kedaluwarsa yang masih berada dalam memori. Hal ini juga akan mencoba mencegah konsumsi lebih dari 25 persen dari total memori dan menambahkan latensi ke sistem. Anda dapat meningkatkan nilai ini hingga 10 untuk meningkatkan jumlah upaya yang digunakan untuk kunci kedaluwarsa. Sebagai gantinya, CPU akan beroperasi lebih keras dan latensi berpotensi lebih tinggi. Kami merekomendasikan nilai 1 kecuali jika Anda melihat penggunaan memori tinggi dan dapat menoleransi peningkatan utilisasi CPU. |
| `lazyfree-lazy-user-del (added in 6.0)` | Default: no<br><br>Tipe: string<br><br>Dapat diubah: Ya<br><br>Perubahan berlaku: Segera di semua simpul dalam klaster | Ketika nilai diatur ke ya, perintah `DEL` bertindak sama seperti `UNLINK`. |

Parameter yang dihapus dalam Redis 6.x adalah sebagai berikut.

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `lua-replicate-commands` | Nilai yang diizinkan: yes/no<br>Default: yes<br>Tipe: boolean<br>Dapat diubah: Ya<br>Perubahan berlaku: Segera. | Selalu mengaktifkan replikasi efek Lua atau tidak dalam skrip Lua |

Perubahan parameter Redis 5.0.3

Rangkaian grup parameter: redis5.0

Grup parameter default Redis 5.0

- `default.redis5.0` – Gunakan grup parameter ini, atau turunannya, untuk grup replikasi dan klaster Redis (mode klaster dinonaktifkan).
- `default.redis5.0.cluster.on` – Gunakan grup parameter ini, atau turunannya, untuk grup replikasi dan klaster Redis (mode klaster diaktifkan).

Parameter yang ditambahkan dalam Redis 5.0.3

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `rename-commands` | Default: tidak ada<br>Tipe: string<br>Dapat diubah: Ya<br>Perubahan berlaku: Segera di semua simpul dalam klaster | Daftar yang dipisahkan oleh spasi dari perintah Redis yang diubah namanya. Berikut adalah daftar terbatas perintah yang tersedia untuk diubah namanya:<br>`APPEND AUTH BITCOUNT BITFIELD BITOP BITPOS BLPOP BRPOP BRPOPLPUSH BZPOPMIN BZPOPMAX CLIENT CLUSTER COMMAND DBSIZE DECR DECRBY DEL DISCARD DUMP ECHO EVAL EVALSHA EXEC EXISTS EXPIRE` |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| | | EXPIREAT FLUSHALL FLUSHDB GEOADD GEOHASH GEOPOS GEODIST GEORADIUS GEORADIUSBYMEMBER GET GETBIT GETRANGE GETSET HDEL HEXISTS HGET HGETALL HINCRBY HINCRBYFL OAT HKEYS HLEN HMGET HMSET HSET HSETNX HSTRLEN HVALS INCR INCRBY INCRBYFLOAT INFO KEYS LASTSAVE LINDEX LINSERT LLEN LPOP LPU SH LPUSHX LRANGE LREM LSET LTRIM MEMORY MGET MONITOR MOVE MSET MSETNX MULTI OBJECT PERSIST PEXPIRE PEXPIREAT PFADD PFCOUNT PFMERGE PING PSETEX PSUBSCRIBE PUBSUB PTTL PUBLISH PUNSUBSCRIBE RANDOMKEY READONLY READWRITE RENAME RENAMENX RESTORE ROLE RPOP RPOPLPUSH RPUSH RPUSHX SADD SCARD SCRIPT SDIFF SDIFFSTORE SELECT SET SETBIT SETEX SETNX SETRANGE SINTER SINTERSTORE SISMEMBER SLOWLOG SMEMBERS SMOVE SORT SPOP SRANDMEMBER SREM STRLEN SUBSCRIBE SUNION SUNIONSTORE SWAPDB TIME TOUCH TTL TYPE UNSUBSCRIBE UNLINK UNWATCH WAIT WATCH ZADD ZCARD ZCOUNT ZINCRBY ZINTERSTO RE ZLEXCOUNT ZPOPMAX ZPOPMIN ZRANGE ZRANGEBYLEX ZREVRANGE BYLEX ZRANGEBYSCORE ZRANK ZREM ZREMRANGEBYLEX ZREMRANGEBYRANK ZREMRANGEBYSCORE ZREVRANGE ZREVRANGEBYSCORE ZREVRANK ZSCORE ZUNIONSTORE SCAN SSCAN HSCAN |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| | | `ZSCAN XINFO XADD XTRIM XDEL XRANGE XREVRANGE XLEN XREAD XGROUP XREADGROUP XACK XCLAIM XPENDING GEORADIUS_RO GEORADIUSBYMEMBER_RO LOLWUT XSETID SUBSTR` |

Untuk informasi selengkapnya, lihat ElastiCache for Redis versi 5.0.6 (Ditingkatkan).

Perubahan parameter Redis 5.0.0

Rangkaian grup parameter: redis5.0

Grup parameter default Redis 5.0

- `default.redis5.0` – Gunakan grup parameter ini, atau turunannya, untuk grup replikasi dan klaster Redis (mode klaster dinonaktifkan).
- `default.redis5.0.cluster.on` – Gunakan grup parameter ini, atau turunannya, untuk grup replikasi dan klaster Redis (mode klaster diaktifkan).

Parameter yang ditambahkan dalam Redis 5.0

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `stream-node-max-bytes` | Nilai yang diizinkan: 0+<br>Default: 4096<br>Tipe: integer<br>Dapat diubah: Ya<br>Perubahan berlaku: Segera. | Struktur data stream adalah pohon radix simpul yang mengenkode beberapa item di dalamnya. Gunakan konfigurasi ini untuk menentukan ukuran maksimum simpul tunggal dalam pohon radix dalam Byte. Jika diatur ke 0, ukuran simpul pohon tidak terbatas. |
| `stream-node-max-entries` | Nilai yang diizinkan: 0+<br>Default: 100 | Struktur data stream adalah pohon radix simpul yang mengenkode beberapa item di dalamnya. Gunakan konfigurasi ini untuk menentukan jumlah maksimum item yang dapat ditampung |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| | Tipe: integer<br>Dapat diubah: Ya<br>Perubahan berlaku: Segera. | simpul tunggal sebelum beralih ke simpul baru saat menambahkan entri stream baru. Jika diatur ke 0, jumlah item di simpul pohon adalah tidak terbatas |
| `active-defrag-max-scan-fields` | Nilai yang diizinkan: 1 hingga 1000000<br>Default: 1000<br>Tipe: integer<br>Dapat diubah: Ya<br>Perubahan berlaku: Segera. | Jumlah maksimum bidang set/hash/zset/list yang akan diproses dari pemindaian kamus utama |
| `lua-replicate-commands` | Nilai yang diizinkan: yes/no<br>Default: yes<br>Tipe: boolean<br>Dapat diubah: Ya<br>Perubahan berlaku: Segera. | Selalu mengaktifkan replikasi efek Lua atau tidak dalam skrip Lua |
| `replica-ignore-maxmemory` | Default: yes<br>Tipe: boolean<br>Dapat diubah: Tidak | Menentukan apakah replika mengabaikan pengaturan `maxmemory` dengan tidak mengosongkan item yang independen dari primer |

Redis telah mengubah nama sejumlah parameter di versi mesin 5.0 dalam menanggapi umpan balik komunitas. Untuk informasi selengkapnya, lihat Apa yang Baru di Redis 5?. Tabel berikut mencantumkan nama baru dan pemetaannya ke versi sebelumnya.

Parameter yang diubah namanya dalam Redis 5.0

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `replica-lazy-flush` | Default: yes<br>Tipe: boolean<br>Dapat diubah: Tidak<br>Nama sebelumnya: slave-lazy-flush | Melakukan flushDB asinkron selama sinkronisasi replika. |
| `client-output-buffer-limit-replica-hard-limit` | Default: Untuk nilai, lihat Parameter khusus jenis simpul Redis<br>Tipe: integer<br>Dapat diubah: Tidak<br>Nama sebelumnya: client-output-buffer-limit-slave-hard-limit | Untuk replika baca Redis: Jika buffer output klien mencapai jumlah byte tertentu, klien akan terputus. |
| `client-output-buffer-limit-replica-soft-limit` | Default: Untuk nilai, lihat Parameter khusus jenis simpul Redis<br>Tipe: integer<br>Dapat diubah: Tidak<br>Nama sebelumnya: client-output-buffer-limit-slave-soft-limit | Untuk replika baca Redis: Jika buffer output klien mencapai jumlah byte tertentu, klien akan terputus, tetapi hanya jika kondisi ini tetap ada untuk `client-output-buffer-limit-replica-soft-seconds` . |
| `client-output-buffer-limit-replica-s` | Default: 60<br>Tipe: integer<br>Dapat diubah: Tidak | Untuk replika baca Redis: Jika buffer output klien tetap sebesar `client-output-buffer-limit-replica-soft-limit` byte |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `oft-seconds` | Nama sebelumnya: client-output-buffer-limit-slave-soft-seconds | selama lebih dari jumlah detik ini, klien akan terputus. |
| `replica-allow-chaining` | Default: no<br>Tipe: string<br>Dapat diubah: Tidak<br>Nama sebelumnya: slave-allow-chaining | Menentukan apakah replika baca di Redis dapat membaca replika sendiri. |
| `min-replicas-to-write` | Default: 0<br>Tipe: integer<br>Dapat diubah: Ya<br>Nama sebelumnya: min-slave-to-write<br>Perubahan Berlaku: Segera | Jumlah minimum replika baca yang harus tersedia agar simpul primer dapat menerima penulisan dari klien. Jika jumlah replika yang tersedia di bawah jumlah ini, maka simpul primer tidak akan lagi menerima permintaan tulis.<br>Jika parameter ini atau min-replicas-max-lag adalah 0, maka simpul primer akan selalu menerima permintaan tulis, meskipun tidak ada replika yang tersedia. |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `min-replicas-max-lag` | Default: 10<br><br>Tipe: integer<br><br>Dapat diubah: Ya<br><br>Nama sebelumnya: min-slaves-max-lag<br><br>Perubahan Berlaku: Segera | Jumlah detik saat simpul primer harus menerima permintaan ping dari replika baca. Jika jumlah waktu ini berlalu dan primer tidak menerima ping, maka replika tidak lagi dianggap tersedia. Jika jumlah replika yang tersedia di bawah min-replicas-to-write, maka primer akan berhenti menerima penulisan pada saat itu.<br><br>Jika parameter ini atau min-replicas-to-write adalah 0, maka simpul primer akan selalu menerima permintaan tulis, meskipun tidak ada replika yang tersedia. |
| `close-on-replica-write` | Default: yes<br><br>Tipe: boolean<br><br>Dapat diubah: Ya<br><br>Nama sebelumnya: close-on-slave-write<br><br>Perubahan Berlaku: Segera | Jika diaktifkan, klien yang mencoba menulis ke replika hanya baca akan terputus. |

Parameter yang dihapus dalam Redis 5.0

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `repl-timeout` | Default: 60<br><br>Dapat diubah: Tidak | Parameter tidak tersedia dalam versi ini. |

Perubahan parameter Redis 4.0.10

Rangkaian grup parameter: redis4.0

Grup parameter default Redis 4.0.x

- `default.redis4.0` – Gunakan grup parameter ini, atau turunannya, untuk grup replikasi dan klaster Redis (mode klaster dinonaktifkan).
- `default.redis4.0.cluster.on` – Gunakan grup parameter ini, atau turunannya, untuk grup replikasi dan klaster Redis (mode klaster diaktifkan).

Parameter yang diubah dalam Redis 4.0.10

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `maxmemory-policy` | Nilai yang diizinkan: `allkeys-lru` , `volatile-lru` , **`allkeys-lfu`** , **`volatile-lfu`** , `allkeys-random` , `volatile-random` , `volatile-ttl` , `noeviction`<br><br>Default: volatile-lru<br><br>Tipe: string<br><br>Dapat diubah: Ya<br><br>Perubahan berlaku: segera | `maxmemory-policy` telah ditambahkan dalam versi 2.6.13. Dalam versi 4.0.10, ditambahkan dua nilai baru yang diizinkan : `allkeys-lfu` , yang akan mengosongkan setiap kunci menggunakan LFU yang diperkirakan, dan `volatile-lfu` , yang akan mengosongkan kunci dengan set yang kedaluwarsa menggunakan LFU yang diperkirakan. Dalam versi 6.2, ketika rangkaian simpul r6gd diperkenalkan untuk digunakan dengan tingkatan data, hanya kebijakan max-memory `noeviction` , `volatile-lru` dan `allkeys-lru` yang didukung dengan tipe simpul r6gd. |

Parameter yang ditambahkan dalam Redis 4.0.10

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| Parameter penghapusan asinkron | | |
| `lazyfree-lazy-eviction` | Nilai yang diizinkan: yes/no<br><br>Default: no<br><br>Tipe: boolean | Performs an asynchronous delete on evictions. |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| | Dapat diubah: Ya<br>Perubahan berlaku: segera | |
| `lazyfree-lazy-expire` | Nilai yang diizinkan: yes/no<br>Default: no<br>Tipe: boolean<br>Dapat diubah: Ya<br>Perubahan berlaku: segera | Performs an asynchronous delete on expired keys. |
| `lazyfree-lazy-server-del` | Nilai yang diizinkan: yes/no<br>Default: no<br>Tipe: boolean<br>Dapat diubah: Ya<br>Perubahan berlaku: segera | Performs an asynchronous delete for commands which update values. |
| `slave-lazy-flush` | Nilai yang diizinkan: Tidak Ada<br>Default: no<br>Tipe: boolean<br>Dapat diubah: Tidak<br>Changes take place: N/A | Performs an asynchronous flushDB during slave sync. |
| Parameter LFU | | |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `lfu-log-factor` | Nilai yang diizinkan: semua integer > 0<br>Default: 10<br>Tipe: integer<br>Dapat diubah: Ya<br>Perubahan berlaku: segera | Set the log factor, which determines the number of key hits to saturate the key counter. |
| `lfu-decay-time` | Nilai yang diizinkan: semua integer<br>Default: 1<br>Tipe: integer<br>Dapat diubah: Ya<br>Perubahan berlaku: segera | The amount of time in minutes to decrement the key counter. |
| Parameter defragmentasi aktif | | |
| `activedefrag` | Nilai yang diizinkan: yes/no<br>Default: no<br>Tipe: boolean<br>Dapat diubah: Ya<br>Perubahan berlaku: segera | Enabled active defragmentation. |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `active-defrag-ignore-bytes` | Nilai yang diizinkan: 10485760-104857600<br>Default: 104857600<br>Tipe: integer<br>Dapat diubah: Ya<br>Perubahan berlaku: segera | Minimum amount of fragmentation waste to start active defrag. |
| `active-defrag-threshold-lower` | Nilai yang diizinkan: 1-100<br>Default: 10<br>Tipe: integer<br>Dapat diubah: Ya<br>Perubahan berlaku: segera | Minimum percentage of fragmentation to start active defrag. |
| `active-defrag-threshold-upper` | Nilai yang diizinkan: 1-100<br>Default: 100<br>Tipe: integer<br>Dapat diubah: Ya<br>Perubahan berlaku: segera | Maximum percentage of fragmentation at which we use maximum effort. |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `active-defrag-cycle-min` | Nilai yang diizinkan: 1-75<br>Default: 25<br>Tipe: integer<br>Dapat diubah: Ya<br>Perubahan berlaku: segera | Minimal effort for defrag in CPU percentage. |
| `active-defrag-cycle-max` | Nilai yang diizinkan: 1-75<br>Default: 75<br>Tipe: integer<br>Dapat diubah: Ya<br>Perubahan berlaku: segera | Maximal effort for defrag in CPU percentage. |
| Parameter buffer output klien | | |
| `client-query-buffer-limit` | Nilai yang diizinkan: 1048576-1073741824<br>Default: 1073741824<br>Tipe: integer<br>Dapat diubah: Ya<br>Perubahan berlaku: segera | Max size of a single client query buffer. |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `proto-max-bulk-len` | Nilai yang diizinkan: 1048576-536870912<br><br>Default: 536870912<br><br>Tipe: integer<br><br>Dapat diubah: Ya<br><br>Perubahan berlaku: segera | Max size of a single element request. |

Perubahan parameter Redis 3.2.10

Rangkaian grup parameter: redis3.2

Untuk ElastiCache for Redis 3.2.10, tidak ada parameter tambahan yang didukung.

Perubahan parameter Redis 3.2.6

Rangkaian grup parameter: redis3.2

Untuk Redis 3.2.6, tidak ada parameter tambahan yang didukung.

Perubahan parameter Redis 3.2.4

Rangkaian grup parameter: redis3.2

Dimulai dari Redis 3.2.4, terdapat dua grup parameter default.

- `default.redis3.2` – Ketika menjalankan Redis 3.2.4, tentukan grup parameter ini atau turunannya, jika Anda ingin membuat grup replikasi Redis (mode klaster dinonaktifkan) dan masih menggunakan fitur tambahan Redis 3.2.4.
- `default.redis3.2.cluster.on` – Tentukan grup parameter ini atau turunannya, jika Anda ingin membuat grup replikasi Redis (mode klaster diaktifkan).

Topik

- Parameter baru untuk Redis 3.2.4
- Parameter yang diubah dalam Redis 3.2.4 (ditingkatkan)

Parameter baru untuk Redis 3.2.4

Rangkaian grup parameter: redis3.2

Untuk Redis 3.2.4, parameter tambahan berikut didukung.

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `list-max-ziplist-size` | Default: -2<br>Tipe: integer<br>Dapat diubah: Tidak | Daftar dienkode dengan cara khusus untuk menghemat ruang. Jumlah entri yang diizinkan per simpul daftar internal dapat ditentukan sebagai ukuran maksimum tetap atau jumlah maksimum elemen. Untuk ukuran maksimum tetap, gunakan -5 hingga -1, yang berarti:<br>• -5: ukuran maks: 64 Kb - tidak direkomendasikan untuk beban kerja normal<br>• -4: ukuran maks: 32 Kb - tidak direkomendasikan<br>• -3: ukuran maks: 16 Kb - tidak direkomendasikan<br>• -2: ukuran maks: 8 Kb - direkomendasikan<br>• -1: ukuran maks: 4 Kb - direkomendasikan<br>• Angka positif berarti menyimpan hingga persis jumlah elemen per simpul daftar tersebut. |
| `list-compress-depth` | Default: 0<br>Tipe: integer<br>Dapat diubah: Ya<br>Perubahan Berlaku: Segera | Daftar juga dapat dikompresi. Kedalaman kompresi adalah jumlah simpul quicklist ziplist dari setiap sisi daftar yang akan dikecualikan dari kompresi. Kepala dan ekor dari daftar selalu tidak dikompresi untuk operasi fast push dan pop Pengaturannya adalah: |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| | | • 0: Menonaktifkan semua kompresi.<br>• 1: Mulai mengompresi dengan simpul pertama masuk dari kepala dan ekor.<br>[head]->simpul->simpul->...->simpul->[tail]<br>Semua simpul kecuali jika [head] dan [tail] dikompresi.<br>• 2: Mulai mengompresi dengan simpul kedua masuk dari kepala dan ekor.<br>[head]->[next]->simpul->simpul->...->simpul->[prev]->[tail]<br>[head], [next], [prev], [tail] tidak dikompresi. Semua simpul lainnya dikompresi.<br>• Dll. |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `cluster-enabled` | Default: tidak/ya *<br><br>Tipe: string<br><br>Dapat diubah: Tidak | Menunjukkan apakah ini adalah grup replikasi Redis (mode klaster diaktifkan) dalam mode klaster (ya) atau grup replikasi Redis (klaster diaktifkan) dalam mode non-klaster (tidak). Grup replikasi Redis (mode klaster diaktifkan) dalam mode klaster dapat mempartisi datanya di maksimal 500 grup simpul.<br><br>* Redis 3.2.x memiliki dua grup parameter default.<br><br>• `default.redis3.2` – Nilai default no.<br>• `default.redis3.2.cluster.on` – Nilai default `yes`.<br><br>. |
| `cluster-require-full-coverage` | Default: no<br><br>Tipe: boolean<br><br>Dapat diubah: ya<br><br>Perubahan Berlaku: Segera | Ketika diatur ke `yes`, simpul Redis (mode klaster diaktifkan) dalam mode klaster berhenti menerima kueri jika mendeteksi setidakny a satu slot hash yang tidak tercakup (tidak ada yang simpul untuk melayaninya). Dengan cara ini, jika sebagian klaster berhenti, klaster menjadi tidak tersedia. Klaster secara otomatis menjadi tersedia lagi begitu semua slot tercakup lagi.<br><br>Namun, terkadang Anda ingin subset klaster yang berfungsi terus menerima permintaa n untuk bagian dari ruang kunci yang masih tercakup. Untuk melakukannya, cukup atur opsi `cluster-require-full-coverage` ke no. |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `hll-sparse-max-bytes` | Default: 3000<br><br>Tipe: integer<br><br>Dapat diubah: Ya<br><br>Perubahan Berlaku: Segera | Batas byte sparse representation HyperLogLog. Batas termasuk header 16 byte. Ketika HyperLogLog yang menggunakan sparse representation melewati batas ini, HyperLogLog ini dikonversi menjadi dense representation.<br><br>Nilai yang lebih besar dari 16000 tidak disarankan karena pada titik tersebut dense representation lebih hemat memori.<br><br>Kami merekomendasikan nilai sekitar 3000 untuk mendapatkan manfaat dari pengenkodean hemat ruang tanpa terlalu memperlambat PFADD, yaitu O(N) dengan sparse encoding. Nilai dapat dinaikkan ke ~10000 jika ruang, bukan CPU, dapat menjadi masalah, dan set data terdiri dari banyak HyperLogLog dengan kardinalitas dalam rentang 0 - 15000. |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `reserved-memory-percent` | Default: 25<br><br>Tipe: integer<br><br>Dapat diubah: Ya<br><br>Perubahan Berlaku: Segera | Persentase memori simpul yang dicadangkan untuk penggunaan nondata. Secara default, jejak data Redis akan bertambah sampai mengonsumsi semua memori simpul. Jika ini terjadi, maka performa simpul kemungkinan akan terdampak negatif karena memory paging yang berlebihan. Dengan mencadangkan memori, Anda dapat menyisihkan sebagian memori yang tersedia untuk tujuan non-Redis agar membantu mengurangi jumlah paging.<br><br>Parameter ini spesifik untuk ElastiCache, dan bukan bagian dari distribusi Redis standar.<br><br>Untuk informasi selengkapnya, lihat `reserved-memory` dan Mengelola Memori Terpesan. |

Parameter yang diubah dalam Redis 3.2.4 (ditingkatkan)

Rangkaian grup parameter: redis3.2

Untuk Redis 3.2.4, parameter berikut telah diubah.

| Nama | Detail | Perubahan |
| --- | --- | --- |
| `activerehashing` | Dapat diubah: Ya jika grup parameter tidak terkait dengan klaster cache. Jika tidak, tidak. | Dapat diubah adalah Tidak. |
| `databases` | Dapat diubah: Ya jika grup parameter tidak terkait dengan klaster cache. Jika tidak, tidak. | Dapat diubah adalah Tidak. |

| Nama | Detail | Perubahan |
| --- | --- | --- |
| `appendonly` | Default: nonaktif<br>Dapat diubah: Tidak | Jika ingin meng-upgrade dari versi Redis sebelumnya, Anda harus terlebih dahulu menonaktifkan `appendonly` . |
| `appendfsync` | Default: nonaktif<br>Dapat diubah: Tidak | Jika ingin meng-upgrade dari versi Redis sebelumnya, Anda harus terlebih dahulu menonaktifkan `appendfsync` . |
| `repl-timeout` | Default: 60<br>Dapat diubah: Tidak | Sekarang tidak dapat diubah dengan default 60. |
| `tcp-keepalive` | Default: 300 | Default adalah 0. |
| `list-max-ziplist-entries` | | Parameter tidak lagi tersedia. |
| `list-max-ziplist-value` | | Parameter tidak lagi tersedia. |

Parameter yang ditambahkan dalam Redis 2.8.24 (ditingkatkan)

Rangkaian grup parameter: redis2.8

Untuk Redis 2.8.24, tidak ada parameter tambahan yang didukung.

Parameter yang ditambahkan dalam Redis 2.8.23 (ditingkatkan)

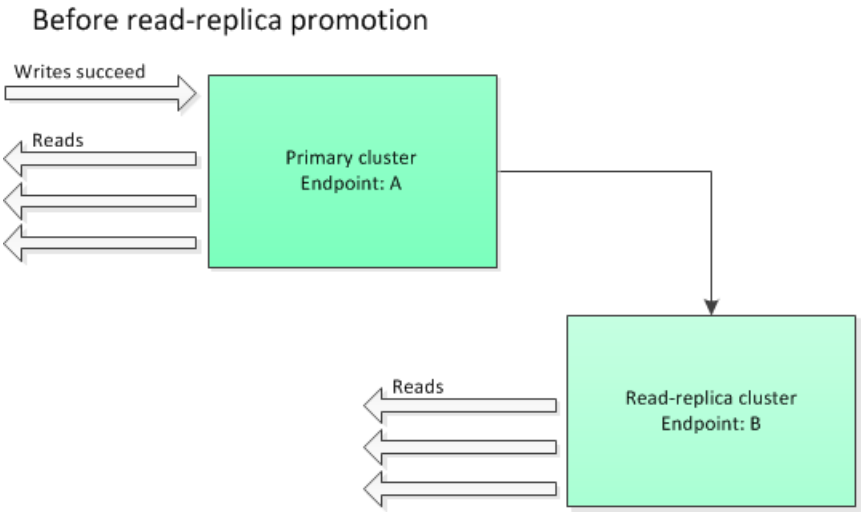
Rangkaian grup parameter: redis2.8

Untuk Redis 2.8.23, parameter tambahan berikut didukung.

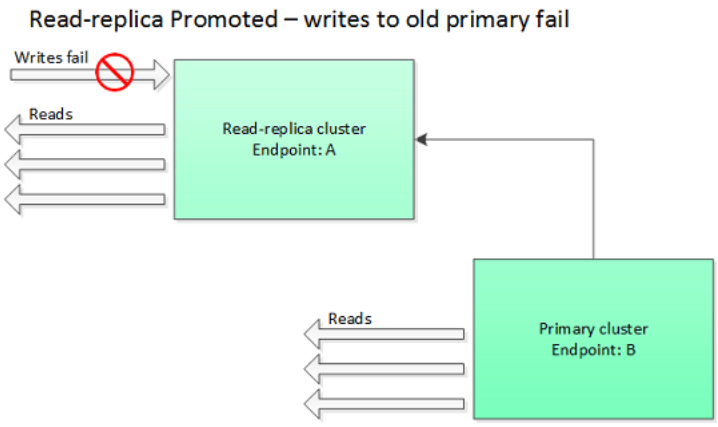
| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `close-on-slave-write` | Default: yes<br>Tipe: string (ya/tidak)<br>Dapat diubah: Ya<br>Perubahan Berlaku: Segera | Jika diaktifkan, klien yang mencoba menulis ke replika hanya baca akan terputus. |

Cara kerja close-on-slave-write

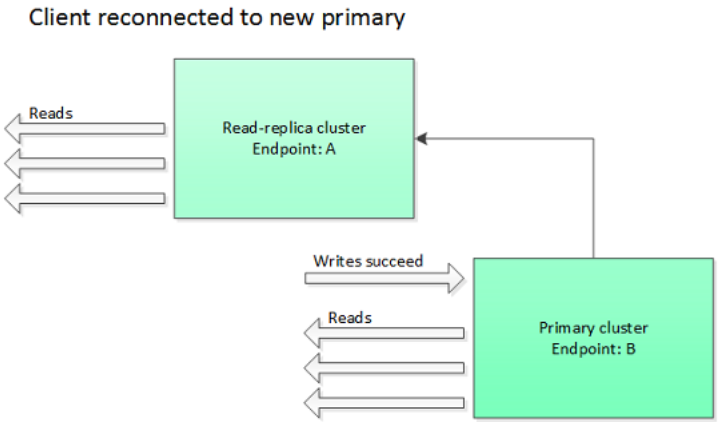
Parameter `close-on-slave-write` diperkenalkan oleh Amazon ElastiCache untuk memberikan lebih banyak kontrol pada cara klaster Anda merespons ketika simpul primer dan simpul replika baca bertukar peran karena mempromosikan replika baca ke primer.




Jika klaster replika baca dipromosikan ke primer untuk alasan apa pun selain failover grup replikasi dengan Multi-AZ diaktifkan, klien akan terus mencoba menulis ke titik akhir A. Karena titik akhir A sekarang adalah titik akhir untuk replika baca, penulisan ini akan gagal. Ini adalah perilaku untuk Redis sebelum ElastiCache memperkenalkan `close-on-replica-write` dan perilaku jika Anda menonaktifkan `close-on-replica-write`.




Dengan `close-on-replica-write` diaktifkan, setiap kali klien mencoba menulis ke replika baca, koneksi klien ke klaster ditutup. Logika aplikasi Anda harus mendeteksi pemutusan koneksi, memeriksa tabel DNS, dan menghubungkan kembali ke titik akhir primer, yang sekarang akan menjadi titik akhir B.




Kapan Anda dapat menonaktifkan close-on-replica-write

Jika penonaktifan `close-on-replica-write` mengakibatkan gagalnya penulisan ke klaster, mengapa `close-on-replica-write` dinonaktifkan?

Seperti yang telah disebutkan, dengan mengaktifkan `close-on-replica-write`, setiap kali klien mencoba menulis ke replika baca, koneksi klien ke klaster akan ditutup. Pembuatan koneksi baru ke simpul membutuhkan waktu. Jadi, pemutusan koneksi dan pembuatan koneksi kembali sebagai akibat dari permintaan tulis ke replika juga memengaruhi latensi permintaan baca yang dilayani melalui koneksi yang sama. Efek ini tetap ada sampai dibuatnya koneksi baru. Jika aplikasi Anda sangat sarat pembacaan atau sangat sensitif terhadap latensi, Anda dapat menjaga klien tetap terhubung untuk menghindari performa baca yang menurun.

Parameter yang ditambahkan dalam Redis 2.8.22 (ditingkatkan)

Rangkaian grup parameter: redis2.8

Untuk Redis 2.8.22, tidak ada parameter tambahan yang didukung.

> **Important**
>
> - Dimulai dengan Redis versi 2.8.22, `repl-backlog-size` berlaku untuk klaster primer serta klaster replika.
> - Dimulai dengan Redis versi 2.8.22, parameter `repl-timeout` tidak didukung. Jika parameter ini diubah, ElastiCache akan menimpa dengan nilai default (60s), seperti pada `appendonly`.

Parameter berikut tidak lagi didukung.

- appendonly
- appendfsync
- repl-timeout

Parameter yang ditambahkan dalam Redis 2.8.21

Rangkaian grup parameter: redis2.8

Untuk Redis 2.8.21, tidak ada parameter tambahan yang didukung.

Parameter yang ditambahkan dalam Redis 2.8.19

Rangkaian grup parameter: redis2.8

Untuk Redis 2.8.19, tidak ada parameter tambahan yang didukung.

Parameter yang ditambahkan dalam Redis 2.8.6

Rangkaian grup parameter: redis2.8

Untuk Redis 2.8.6, parameter tambahan berikut didukung.

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `min-slaves-max-lag` | Default: 10<br><br>Tipe: integer<br><br>Dapat diubah: Ya<br><br>Perubahan Berlaku: Segera | Jumlah detik saat simpul primer harus menerima permintaan ping dari replika baca. Jika jumlah waktu ini berlalu dan primer tidak menerima ping, maka replika tidak lagi dianggap tersedia. Jika jumlah replika yang tersedia turun di bawah min-slaves-to-write, maka primer akan berhenti menerima permintaan tulis pada tahap tersebut.<br><br>Jika parameter ini atau min-slaves-to-write adalah 0, maka simpul primer akan selalu menerima permintaan tulis, meskipun tidak ada replika yang tersedia. |
| `min-slaves-to-write` | Default: 0<br><br>Tipe: integer<br><br>Dapat diubah: Ya<br><br>Perubahan Berlaku: Segera | Jumlah minimum replika baca yang harus tersedia agar simpul primer dapat menerima penulisan dari klien. Jika jumlah replika yang tersedia di bawah jumlah ini, maka simpul primer tidak akan lagi menerima permintaan tulis.<br><br>Jika parameter ini atau min-slaves-max-lag adalah 0, maka simpul primer akan selalu menerima permintaan tulis, meskipun tidak ada replika yang tersedia. |
| `notify-keyspace-events` | Default: (string kosong)<br><br>Tipe: string | Jenis peristiwa ruang kunci yang dapat diberitahukan oleh Redis |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| | Dapat diubah: Ya<br><br>Perubahan Berlaku: Segera | ke klien. Setiap jenis peristiwa direpresentasikan oleh satu huruf:<br>• K – Peristiwa ruang kunci, diterbitkan dengan awalan __keyspace@<db>__<br>• E – Peristiwa key-event, diterbitkan dengan awalan __keyevent@<db>__<br>• g – Perintah generik non-spesifik seperti DEL, EXPIRE, RENAME, dll.<br>• $ – Perintah String<br>• l – Perintah Daftar<br>• s – Perintah Set<br>• h – Perintah Hash<br>• z – Perintah Set berurutan<br>• x – Peristiwa kedaluwarsa (peristiwa yang dihasilkan setiap kali kunci kedaluwarsa)<br>• e – Peristiwa pengosongan (peristiwa yang dihasilkan ketika kunci dikosongkan untuk maxmemory)<br>• A – Alias untuk g$lshzxe |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| | | Anda dapat memiliki kombinasi dari semua jenis peristiwa ini. Misalnya, AKE berarti bahwa Redis dapat menerbitkan notifikasi dari semua jenis peristiwa.<br><br>Jangan gunakan karakter selain yang tercantum di atas; jika tidak, pesan kesalahan akan dihasilkan.<br><br>Secara default, parameter ini diatur ke string kosong, yang berarti bahwa notifikasi peristiwa ruang kunci dinonaktifkan. |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `repl-backlog-size` | Default: 1048576<br><br>Tipe: integer<br><br>Dapat diubah: Ya<br><br>Perubahan Berlaku: Segera | Ukuran, dalam byte, buffer backlog simpul primer. Backlog digunakan untuk mencatat pembaruan data pada simpul primer. Ketika replika baca terhubung ke primer, replika ini mencoba melakukan sinkronisasi parsial (`psync`), yang menerapkan data dari backlog untuk mengejar simpul primer. Jika `psync` gagal, maka sinkronisasi penuh diperlukan.<br><br>Nilai minimum untuk parameter ini adalah 16384.<br><br>**Note**<br>Dimulai dengan Redis 2.8.22, parameter ini berlaku untuk klaster primer serta klaster replika. |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `repl-backlog-ttl` | Default: 3600<br><br>Tipe: integer<br><br>Dapat diubah: Ya<br><br>Perubahan Berlaku: Segera | Jumlah detik saat simpul primer mempertahankan buffer backlog. Mulai dari waktu simpul replika terakhir terputus, data dalam backlog akan tetap utuh sampai `repl-backlog-ttl` kedaluwarsa. Jika replika tidak terhubung ke primer dalam waktu ini, maka primer akan melepaskan buffer backlog. Ketika replika akhirnya terhubung kembali, replikasi ini harus melakukan sinkronisasi penuh dengan primer.<br><br>Jika parameter ini diatur ke 0, maka buffer backlog tidak akan pernah dilepas. |
| `repl-timeout` | Default: 60<br><br>Tipe: integer<br><br>Dapat diubah: Ya<br><br>Perubahan Berlaku: Segera | Merepresentasikan periode waktu habis, dalam detik, untuk:<br><br>• Transfer data massal selama sinkronisasi, dari perspektif replika baca<br>• Waktu habis simpul primer dari perspektif replika<br>• Waktu habis replika dari perspektif simpul primer |

Parameter Redis 2.6.13

Rangkaian grup parameter: redis2.6

Redis versi 2.6.13 adalah versi pertama Redis yang didukung oleh ElastiCache. Tabel berikut menunjukkan parameter Redis 2.6.13 yang didukung ElastiCache.

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `activerehashing` | Default: yes<br><br>Tipe: string (ya/tidak)<br><br>Dapat diubah: Ya<br><br>Perubahan berlaku: Saat Pembuatan | Menentukan apakah akan mengaktifkan fitur rehashing aktif Redis. Tabel hash utama di-rehash sepuluh kali per detik; setiap operasi rehash mengonsumsi 1 milidetik waktu CPU.<br><br>Nilai ini diatur saat Anda membuat grup parameter. Ketika menetapkan grup parameter baru untuk klaster, nilai ini harus sama dalam grup parameter lama dan baru. |
| `appendonly` | Default: no<br><br>Tipe: string<br><br>Dapat diubah: Ya<br><br>Perubahan Berlaku: Segera | Mengaktifkan atau menonaktifkan fitur append only file (AOF) Redis. AOF mencatat setiap perintah Redis yang mengubah data dalam cache, dan digunakan untuk memulihkan kegagalan simpul tertentu.<br><br>Nilai default adalah no, yang berarti AOF dinonaktifkan. Atur parameter ini ke yes untuk mengaktifkan AOF.<br><br>Untuk informasi selengkapnya, lihat Mitigasi Kegagalan.<br><br>**Note**<br>Append Only File (AOF) tidak didukung untuk simpul cache.t1.micro dan cache.t2.*. Untuk simpul jenis ini, nilai parameter `appendonly` akan diabaikan. |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| | | **Note**<br>Untuk grup replikasi Multi-AZ, AOF tidak diizinkan. |
| `appendfsync` | Default: everysec<br>Tipe: string<br>Dapat diubah: Ya<br>Perubahan Berlaku: Segera | Saat `appendonly` diatur ke yes, akan mengontrol seberapa sering buffer output AOF ditulis ke disk:<br>• no – buffer di-flushing ke disk sesuai kebutuhan.<br>• everysec – buffer di-flushing sekali per detik. Ini adalah opsi default.<br>• always – buffer di-flushing setiap kali data dalam klaster diubah.<br>• Appendfsync tidak didukung untuk versi 2.8.22 dan yang lebih baru. |
| `client-output-buffer-limit-normal-hard-limit` | Default: 0<br>Tipe: integer<br>Dapat diubah: Ya<br>Perubahan Berlaku: Segera | Jika buffer output klien mencapai jumlah byte tertentu, klien akan terputus. Default-nya adalah nol (tidak ada batas absolut). |
| `client-output-buffer-limit-normal-soft-limit` | Default: 0<br>Tipe: integer<br>Dapat diubah: Ya<br>Perubahan Berlaku: Segera | Jika buffer output klien mencapai jumlah byte tertentu, klien akan terputus, tetapi hanya jika kondisi ini bertahan selama `client-output-buffer-limit-normal-soft-seconds` . Default-nya adalah nol (tidak ada batas absolut). |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `client-output-buffer-limit-normal-soft-seconds` | Default: 0<br>Tipe: integer<br>Dapat diubah: Ya<br>Perubahan Berlaku: Segera | Jika buffer output klien tetap pada `client-output-buffer-limit-normal-soft-limit` byte lebih lama dari jumlah detik ini, klien akan terputus. Default-nya adalah nol (tidak ada batas absolut). |
| `client-output-buffer-limit-pubsub-hard-limit` | Default: 33554432<br>Tipe: integer<br>Dapat diubah: Ya<br>Perubahan Berlaku: Segera | Untuk klien publish/subscribe Redis: Jika buffer output klien mencapai jumlah byte tertentu, klien akan terputus. |
| `client-output-buffer-limit-pubsub-soft-limit` | Default: 8388608<br>Tipe: integer<br>Dapat diubah: Ya<br>Perubahan Berlaku: Segera | Untuk klien publish/subscribe Redis: Jika buffer output klien mencapai jumlah byte tertentu, klien akan terputus, tetapi hanya jika kondisi ini bertahan selama `client-output-buffer-limit-pubsub-soft-seconds` . |
| `client-output-buffer-limit-pubsub-soft-seconds` | Default: 60<br>Tipe: integer<br>Dapat diubah: Ya<br>Perubahan Berlaku: Segera | Untuk klien publish/subscribe Redis: Jika buffer output klien tetap di `client-output-buffer-limit-pubsub-soft-limit` byte lebih lama dari jumlah detik ini, klien akan terputus. |
| `client-output-buffer-limit-slave-hard-limit` | Default: Untuk nilai, lihat Parameter khusus jenis simpul Redis<br>Tipe: integer<br>Dapat diubah: Tidak | Untuk replika baca Redis: Jika buffer output klien mencapai jumlah byte tertentu, klien akan terputus. |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `client-output-buffer-limit-slave-soft-limit` | Default: Untuk nilai, lihat Parameter khusus jenis simpul Redis<br>Tipe: integer<br>Dapat diubah: Tidak | Untuk replika baca Redis: Jika buffer output klien mencapai jumlah byte tertentu, klien akan terputus, tetapi hanya jika kondisi ini tetap ada untuk `client-output-buffer-limit-slave-soft-seconds` . |
| `client-output-buffer-limit-slave-soft-seconds` | Default: 60<br>Tipe: integer<br>Dapat diubah: Tidak | Untuk replika baca Redis: Jika buffer output klien tetap sebesar `client-output-buffer-limit-slave-soft-limit` byte selama lebih dari jumlah detik ini, klien akan terputus. |
| `databases` | Default: 16<br>Tipe: integer<br>Dapat diubah: Tidak<br>Perubahan berlaku: Saat Pembuatan | Jumlah partisi logis yang membagi basis data. Kami merekomendasikan untuk menjaga nilai ini tetap rendah.<br>Nilai ini diatur saat Anda membuat grup parameter. Ketika menetapkan grup parameter baru untuk klaster, nilai ini harus sama dalam grup parameter lama dan baru. |
| `hash-max-ziplist-entries` | Default: 512<br>Tipe: integer<br>Dapat diubah: Ya<br>Perubahan Berlaku: Segera | Menentukan jumlah memori yang digunakan untuk hash. Hash dengan jumlah entri kurang dari yang ditentukan akan disimpan menggunakan pengenkodean khusus yang menghemat ruang. |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `hash-max-ziplist-value` | Default: 64<br>Tipe: integer<br>Dapat diubah: Ya<br>Perubahan Berlaku: Segera | Menentukan jumlah memori yang digunakan untuk hash. Hash dengan entri yang lebih kecil dari jumlah byte yang ditentukan akan disimpan menggunakan pengenkodean khusus yang menghemat ruang. |
| `list-max-ziplist-entries` | Default: 512<br>Tipe: integer<br>Dapat diubah: Ya<br>Perubahan Berlaku: Segera | Menentukan jumlah memori yang digunakan untuk daftar. Daftar dengan jumlah entri kurang dari yang ditentukan akan disimpan menggunakan pengenkodean khusus yang menghemat ruang. |
| `list-max-ziplist-value` | Default: 64<br>Tipe: integer<br>Dapat diubah: Ya<br>Perubahan Berlaku: Segera | Menentukan jumlah memori yang digunakan untuk daftar. Daftar dengan entri yang lebih kecil dari jumlah byte yang ditentukan akan disimpan menggunakan pengenkodean khusus yang menghemat ruang. |
| `lua-time-limit` | Default: 5000<br>Tipe: integer<br>Dapat diubah: Tidak | Waktu eksekusi maksimum untuk skrip Lua, dalam milidetik, sebelum ElastiCache mengambil tindakan untuk menghentikan skrip.<br>Jika `lua-time-limit` terlampaui, semua perintah Redis akan menampilkan kesalahan dalam bentuk ____-BUSY. Karena keadaan ini dapat menyebabkan gangguan terhadap banyak operasi penting Redis, ElastiCache akan terlebih dahulu mengeluarkan perintah SCRIPT KILL. Jika tindakan ini tidak berhasil, ElastiCache akan memulai ulang Redis secara paksa. |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `maxclients` Nilai ini berlaku untuk semua tipe instans kecuali yang ditentukan secara eksplisit | Default: 65000<br>Tipe: integer<br>Dapat diubah: Tidak | Jumlah maksimum klien yang dapat dihubungkan pada satu waktu. |
| | t2.medium Default: 20000<br>Tipe: integer<br>Dapat diubah: Tidak | |
| | t2.small Default: 20000<br>Tipe: integer<br>Dapat diubah: Tidak | |
| | t2.micro Default: 20000<br>Tipe: integer<br>Dapat diubah: Tidak | |
| | t4g.micro Default: 20000<br>Tipe: integer<br>Dapat diubah: Tidak | |
| | t3.medium Default: 65000<br>Tipe: integer<br>Dapat diubah: Tidak | |
| | t3.small Default: 65000<br>Tipe: integer<br>Dapat diubah: Tidak | |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| | t3.micro Default: 20000<br><br>Tipe: integer<br><br>Dapat diubah: Tidak | |
| `maxmemory-policy` | Default: volatile-lru<br><br>Tipe: string<br><br>Dapat diubah: Ya<br><br>Perubahan Berlaku: Segera | Kebijakan pengosongan untuk kunci saat penggunaan memori maksimum tercapai.<br><br>Nilai yang valid adalah: `volatile-lru \| allkeys-lru \| volatile-random \| allkeys-random \| volatile-ttl \| noeviction`<br><br>Untuk informasi selengkapnya, lihat Menggunakan Redis sebagai cache LRU. |
| `maxmemory-samples` | Default: 3<br><br>Tipe: integer<br><br>Dapat diubah: Ya<br><br>Perubahan Berlaku: Segera | Untuk penghitungan least-recently-used (LRU) dan time-to-live (TTL), parameter ini merepresentasikan ukuran sampel kunci yang akan diperiksa. Secara default, Redis memilih 3 kunci dan menggunakan kunci yang paling lama tidak digunakan. |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `reserved-memory` | Default: 0<br><br>Tipe: integer<br><br>Dapat diubah: Ya<br><br>Perubahan Berlaku: Segera | Total memori, dalam byte, yang disediakan untuk penggunaan non-data. Secara default, simpul Redis akan tumbuh sampai mengonsumsi `maxmemory` simpul (lihat Parameter khusus jenis simpul Redis). Jika ini terjadi, maka performa simpul kemungkinan akan terdampak negatif karena memory paging yang berlebihan. Dengan mencadangkan memori, Anda dapat menyisihkan sebagian memori yang tersedia untuk tujuan non-Redis agar membantu mengurangi jumlah paging.<br><br>Parameter ini spesifik untuk ElastiCache, dan bukan bagian dari distribusi Redis standar.<br><br>Untuk informasi selengkapnya, lihat `reserved-memory-percent` dan Mengelola Memori Terpesan. |
| `set-max-intset-entries` | Default: 512<br><br>Tipe: integer<br><br>Dapat diubah: Ya<br><br>Perubahan Berlaku: Segera | Menentukan jumlah memori yang digunakan untuk jenis tertentu dari set (string yang berupa integer dalam radix 10 pada rentang integer bertanda 64 bit). Set seperti itu dengan jumlah entri kurang dari yang ditentukan akan disimpan menggunakan pengenkodean khusus yang menghemat ruang. |
| `slave-allow-chaining` | Default: no<br><br>Tipe: string<br><br>Dapat diubah: Tidak | Menentukan apakah replika baca di Redis dapat membaca replika sendiri. |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `slowlog-log-slower-than` | Default: 10000<br><br>Tipe: integer<br><br>Dapat diubah: Ya<br><br>Perubahan Berlaku: Segera | Waktu eksekusi maksimum, dalam mikrodetik, untuk perintah yang akan dicatat lognya oleh fitur Redis Slow Log. |
| `slowlog-max-len` | Default: 128<br><br>Tipe: integer<br><br>Dapat diubah: Ya<br><br>Perubahan Berlaku: Segera | Panjang maksimum Redis Slow Log. |
| `tcp-keepalive` | Default: 0<br><br>Tipe: integer<br><br>Dapat diubah: Ya<br><br>Perubahan Berlaku: Segera | Jika parameter ini diatur ke nilai bukan nol (N), simpul klien akan di-polling setiap N detik untuk memastikan bahwa simpul ini masih terhubung. Dengan pengaturan default 0, tidak ada polling yang terjadi.<br><br>**Important**<br><br>Beberapa aspek dari parameter ini diubah dalam Redis versi 3.2.4. Lihat Parameter yang diubah dalam Redis 3.2.4 (ditingkatkan). |

| Nama | Detail | Deskripsi |
| --- | --- | --- |
| `timeout` | Default: 0<br>Tipe: integer<br>Dapat diubah: Ya<br>Perubahan Berlaku: Segera | Jumlah detik waktu tunggu simpul sebelum batas waktunya habis Nilainya adalah:<br>• `0` – tidak pernah memutus koneksi klien idle.<br>• `1-19` – nilai tidak valid.<br>• `>=20` – jumlah detik waktu tunggu simpul sebelum memutus koneksi klien idle. |
| `zset-max-ziplist-entries` | Default: 128<br>Tipe: integer<br>Dapat diubah: Ya<br>Perubahan Berlaku: Segera | Menentukan jumlah memori yang digunakan untuk set berurutan. Set berurutan dengan jumlah elemen kurang dari yang ditentukan akan disimpan menggunakan pengenkodean khusus yang menghemat ruang. |
| `zset-max-ziplist-value` | Default: 64<br>Tipe: integer<br>Dapat diubah: Ya<br>Perubahan Berlaku: Segera | Menentukan jumlah memori yang digunakan untuk set berurutan. Set berurutan dengan entri yang lebih kecil dari jumlah byte yang ditentukan akan disimpan menggunakan pengenkodean khusus yang menghemat ruang. |

**Note**

Jika Anda tidak menentukan grup parameter untuk klaster Redis 2.6.13, maka grup parameter default (`default.redis2.6`) akan digunakan. Anda tidak dapat mengubah nilai parameter dalam grup parameter default; namun, Anda selalu dapat membuat grup parameter kustom dan menetapkannya ke klaster Anda setiap saat.

### Parameter khusus jenis simpul Redis

Meskipun sebagian besar parameter memiliki nilai tunggal, beberapa parameter memiliki nilai yang berbeda-beda bergantung pada jenis simpul yang digunakan. Tabel berikut menunjukkan nilai default untuk parameter `maxmemory`, `client-output-buffer-limit-slave-hard-limit`, dan `client-output-buffer-limit-slave-soft-limit` untuk setiap jenis simpul. Nilai `maxmemory` adalah jumlah maksimum byte yang tersedia untuk Anda gunakan, untuk data, dan untuk penggunaan lainnya, pada simpul. Untuk informasi selengkapnya, lihat Memori yang tersedia.

> **Note**
>
> Parameter `maxmemory` tidak dapat diubah.

| Jenis simpul | Maxmemory | Client-output-buffer-limit-slave-hard-limit | Client-output-buffer-limit-slave-soft-limit |
|---|---|---|---|
| cache.t1.micro | 142606336 | 14260633 | 14260633 |
| cache.t2.micro | 581959680 | 58195968 | 58195968 |
| cache.t2.small | 1665138688 | 166513868 | 166513868 |
| cache.t2.medium | 3461349376 | 346134937 | 346134937 |
| cache.t3.micro | 536870912 | 53687091 | 53687091 |
| cache.t3.small | 1471026299 | 147102629 | 147102629 |
| cache.t3.medium | 3317862236 | 331786223 | 331786223 |
| cache.t4g.micro | 536870912 | 53687091 | 53687091 |
| cache.t4g.small | 1471026299 | 147102629 | 147102629 |
| cache.t4g.medium | 3317862236 | 331786223 | 331786223 |
| cache.m1.small | 943718400 | 94371840 | 94371840 |

| Jenis simpul | Maxmemory | Client-output-buffer-limit-slave-hard-limit | Client-output-buffer-limit-slave-soft-limit |
|---|---|---|---|
| cache.m1.medium | 3093299200 | 309329920 | 309329920 |
| cache.m1.large | 7025459200 | 702545920 | 702545920 |
| cache.m1.xlarge | 14889779200 | 1488977920 | 1488977920 |
| cache.m2.xlarge | 17091788800 | 1709178880 | 1709178880 |
| cache.m2.2xlarge | 35022438400 | 3502243840 | 3502243840 |
| cache.m2.4xlarge | 70883737600 | 7088373760 | 7088373760 |
| cache.m3.medium | 2988441600 | 309329920 | 309329920 |
| cache.m3.large | 6501171200 | 650117120 | 650117120 |
| cache.m3.xlarge | 14260633600 | 1426063360 | 1426063360 |
| cache.m3.2xlarge | 29989273600 | 2998927360 | 2998927360 |
| cache.m4.large | 6892593152 | 689259315 | 689259315 |
| cache.m4.xlarge | 15328501760 | 1532850176 | 1532850176 |
| cache.m4.2xlarge | 31889126359 | 3188912636 | 3188912636 |
| cache.m4.4xlarge | 65257290629 | 6525729063 | 6525729063 |
| cache.m4.10xlarge | 166047614239 | 16604761424 | 16604761424 |
| cache.m5.large | 6854542746 | 685454275 | 685454275 |
| cache.m5.xlarge | 13891921715 | 1389192172 | 1389192172 |
| cache.m5.2xlarge | 27966669210 | 2796666921 | 2796666921 |
| cache.m5.4xlarge | 56116178125 | 5611617812 | 5611617812 |

| Jenis simpul | Maxmemory | Client-output-buffer-limit-slave-hard-limit | Client-output-buffer-limit-slave-soft-limit |
|---|---|---|---|
| cache.m5.12xlarge | 168715971994 | 16871597199 | 16871597199 |
| cache.m5.24xlarge | 337500562842 | 33750056284 | 33750056284 |
| cache.m6g.large | 6854542746 | 685454275 | 685454275 |
| cache.m6g.xlarge | 13891921715 | 1389192172 | 1389192172 |
| cache.m6g.2xlarge | 27966669210 | 2796666921 | 2796666921 |
| cache.m6g.4xlarge | 56116178125 | 5611617812 | 5611617812 |
| cache.m6g.8xlarge | 111325552312 | 11132555231 | 11132555231 |
| cache.m6g.12xlarge | 168715971994 | 16871597199 | 16871597199 |
| cache.m6g.16xlarge | 225000375228 | 22500037523 | 22500037523 |
| cache.c1.xlarge | 6501171200 | 650117120 | 650117120 |
| cache.r3.large | 14470348800 | 1468006400 | 1468006400 |
| cache.r3.xlarge | 30513561600 | 3040870400 | 3040870400 |
| cache.r3.2xlarge | 62495129600 | 6081740800 | 6081740800 |
| cache.r3.4xlarge | 126458265600 | 12268339200 | 12268339200 |
| cache.r3.8xlarge | 254384537600 | 24536678400 | 24536678400 |
| cache.r4.large | 13201781556 | 1320178155 | 1320178155 |
| cache.r4.xlarge | 26898228839 | 2689822883 | 2689822883 |
| cache.r4.2xlarge | 54197537997 | 5419753799 | 5419753799 |
| cache.r4.4xlarge | 108858546586 | 10885854658 | 10885854658 |

| Jenis simpul | Maxmemory | Client-output-buffer-limit-slave-hard-limit | Client-output-buffer-limit-slave-soft-limit |
|---|---|---|---|
| cache.r4.8xlarge | 218255432090 | 21825543209 | 21825543209 |
| cache.r4.16xlarge | 437021573120 | 43702157312 | 43702157312 |
| cache.r5.large | 14037181030 | 1403718103 | 1403718103 |
| cache.r5.xlarge | 28261849702 | 2826184970 | 2826184970 |
| cache.r5.2xlarge | 56711183565 | 5671118356 | 5671118356 |
| cache.r5.4xlarge | 113609865216 | 11360986522 | 11360986522 |
| cache.r5.12xlarge | 341206346547 | 34120634655 | 34120634655 |
| cache.r5.24xlarge | 682485973811 | 68248597381 | 68248597381 |
| cache.r6g.large | 14037181030 | 1403718103 | 1403718103 |
| cache.r6g.xlarge | 28261849702 | 2826184970 | 2826184970 |
| cache.r6g.2xlarge | 56711183565 | 5671118356 | 5671118356 |
| cache.r6g.4xlarge | 113609865216 | 11360986522 | 11360986522 |
| cache.r6g.8xlarge | 225000375228 | 22500037523 | 22500037523 |
| cache.r6g.12xlarge | 341206346547 | 34120634655 | 34120634655 |
| cache.r6g.16xlarge | 450000750456 | 45000075046 | 45000075046 |
| cache.r6gd.xlarge | 28261849702 | 2826184970 | 2826184970 |
| cache.r6gd.2xlarge | 56711183565 | 5671118356 | 5671118356 |
| cache.r6gd.4xlarge | 113609865216 | 11360986522 | 11360986522 |
| cache.r6gd.8xlarge | 225000375228 | 22500037523 | 22500037523 |

| Jenis simpul | Maxmemory | Client-output-buffer-limit-slave-hard-limit | Client-output-buffer-limit-slave-soft-limit |
|---|---|---|---|
| cache.r6gd.12xlarge | 341206346547 | 34120634655 | 34120634655 |
| cache.r6gd.16xlarge | 450000750456 | 45000075046 | 45000075046 |
| cache.r7g.large | 14037181030 | 1403718103 | 1403718103 |
| cache.r7g.xlarge | 28261849702 | 2826184970 | 2826184970 |
| cache.r7g.2xlarge | 56711183565 | 5671118356 | 5671118356 |
| cache.r7g.4xlarge | 113609865216 | 11360986522 | 11360986522 |
| cache.r7g.8xlarge | 225000375228 | 22500037523 | 22500037523 |
| cache.r7g.12xlarge | 341206346547 | 34120634655 | 34120634655 |
| cache.r7g.16xlarge | 450000750456 | 45000075046 | 45000075046 |
| cache.m7g.large | 6854542746 | 685454275 | 685454275 |
| cache.m7g.xlarge | 13891921715 | 1389192172 | 1389192172 |
| cache.m7g.2xlarge | 27966669210 | 2796666921 | 2796666921 |
| cache.m7g.4xlarge | 56116178125 | 5611617812 | 5611617812 |
| cache.m7g.8xlarge | 111325552312 | 11132555231 | 11132555231 |
| cache.m7g.12xlarge | 168715971994 | 16871597199 | 16871597199 |
| cache.m7g.16xlarge | 225000375228 | 22500037523 | 22500037523 |
| cache.c7gn.large | 3317862236 | 1403718103 | 1403718103 |
| cache.c7gn.xlarge | 6854542746 | 2826184970 | 2826184970 |
| cache.c7gn.2xlarge | 13891921715 | 5671118356 | 5671118356 |

| Jenis simpul | Maxmemory | Client-output-buffer-limit-slave-hard-limit | Client-output-buffer-limit-slave-soft-limit |
|---|---|---|---|
| cache.c7gn.4xlarge | 27966669210 | 11360986522 | 11360986522 |
| cache.c7gn.8xlarge | 56116178125 | 22500037523 | 22500037523 |
| cache.c7gn.12xlarge | 84357985997 | 34120634655 | 34120634655 |
| cache.c7gn.16xlarge | 113609865216 | 45000075046 | 45000075046 |

> **Note**
>
> Semua jenis instans generasi saat ini dibuat di VPC Amazon Cloud Privat Virtual secara default.
> Instans T1 tidak mendukung Multi-AZ.
> Instans T1 dan T2 tidak mendukung AOF Redis.
> Variabel konfigurasi Redis `appendonly` dan `appendfsync` tidak didukung pada Redis versi 2.8.22 dan yang lebih baru.

# Penskalaan ElastiCache untuk Redis

## Penskalaan Tanpa Server ElastiCache

ElastiCache Tanpa server secara otomatis mengakomodasi lalu lintas beban kerja Anda saat naik atau turun. Untuk setiap cache ElastiCache Tanpa Server, ElastiCache terus melacak pemanfaatan sumber daya seperti CPU, memori, dan jaringan. Ketika salah satu sumber daya ini dibatasi, ElastiCache Serverless menskalakan dengan menambahkan pecahan baru dan mendistribusikan ulang data ke pecahan baru, tanpa waktu henti apa pun ke aplikasi Anda. Anda dapat memantau sumber daya yang dikonsumsi oleh cache Anda CloudWatch dengan memantau `BytesUsedForCache` metrik untuk penyimpanan data cache dan `ElastiCacheProcessingUnits` (ECPU) untuk penggunaan komputasi.

## Menetapkan batas penskalaan untuk mengelola biaya

Anda dapat memilih untuk mengonfigurasi penggunaan maksimum pada penyimpanan data cache dan ECPU/detik untuk cache Anda untuk mengontrol biaya cache. Melakukannya akan memastikan bahwa penggunaan cache Anda tidak pernah melebihi maksimum yang dikonfigurasi.

Jika Anda menetapkan skala maksimum, aplikasi Anda mungkin mengalami penurunan kinerja cache saat cache mencapai maksimum. Ketika Anda mengatur penyimpanan data cache maksimum dan penyimpanan data cache Anda mencapai maksimum, ElastiCache akan mulai mengusir data dalam cache Anda yang memiliki set Time-To-Live (TTL), menggunakan logika LRU. Jika tidak ada data yang dapat dikosongkan, maka permintaan untuk menulis data tambahan akan menerima pesan kesalahan Out Of Memory (OOM). Ketika Anda menetapkan maksimum ECPU/detik dan pemanfaatan komputasi beban kerja Anda melebihi nilai ini, ElastiCache akan mulai membatasi permintaan Redis.

Jika Anda mengatur batas maksimum pada `BytesUsedForCache` atau`ElastiCacheProcessingUnits`, kami sangat menyarankan untuk menyiapkan CloudWatch alarm pada nilai yang lebih rendah dari batas maksimum sehingga Anda diberi tahu saat cache Anda beroperasi mendekati batas ini. Sebaiknya atur alarm pada 75% dari batas maksimum yang Anda tetapkan. Lihat dokumentasi tentang cara mengatur CloudWatch alarm.

## Pra-penskalaan dengan Tanpa Server ElastiCache

ElastiCache Pra-penskalaan tanpa server

Dengan pra-penskalaan, juga disebut pra-pemanasan, Anda dapat menetapkan batas minimum yang didukung untuk cache Anda. ElastiCache Anda dapat mengatur minimum ini untuk Unit ElastiCache Pemrosesan (ECPU) per detik atau penyimpanan data. Ini dapat berguna dalam persiapan untuk acara penskalaan yang diantisipasi. Misalnya, jika perusahaan game mengharapkan peningkatan 5x dalam login dalam menit pertama game baru mereka diluncurkan, mereka dapat menyiapkan cache mereka untuk lonjakan penggunaan yang signifikan ini.

Anda dapat melakukan pra-penskalaan menggunakan ElastiCache konsol, CLI, atau API. ElastiCache Tanpa server memperbarui ECPU/detik yang tersedia pada cache dalam waktu 60 menit, dan mengirimkan pemberitahuan acara ketika pembaruan batas minimum selesai.

Cara kerja pra-penskalaan

Ketika batas minimum untuk ECPU/detik atau penyimpanan data diperbarui melalui konsol, CLI, atau API, batas baru itu tersedia dalam 1 jam. ElastiCache Tanpa server mendukung 30K ECPU/

detik pada cache kosong, dan hingga 90K ECPU/detik saat menggunakan fitur Baca dari Replika. ElastiCache dapat menggandakan ECPU/detik setiap 10-12 menit. Kecepatan penskalaan ini cukup untuk sebagian besar beban kerja. Jika Anda mengantisipasi bahwa peristiwa penskalaan yang akan datang mungkin melebihi tingkat ini, maka sebaiknya atur ECPU/detik minimum ke ECPU/detik puncak yang Anda harapkan setidaknya 60 menit sebelum peristiwa puncak. Jika tidak, aplikasi mungkin mengalami peningkatan latensi dan pembatasan permintaan.

Setelah pembaruan batas minimum selesai, ElastiCache Tanpa Server akan mulai mengukur Anda untuk ECPU minimum baru per detik atau penyimpanan minimum baru. Hal ini terjadi bahkan jika aplikasi Anda tidak mengeksekusi permintaan pada cache, atau jika penggunaan penyimpanan data Anda di bawah minimum. Saat Anda menurunkan batas minimum dari pengaturan saat ini, pembaruan segera dilakukan sehingga ElastiCache Tanpa Server akan segera mulai mengukur pada batas minimum yang baru.

> **Note**
>
> - Ketika Anda menetapkan batas penggunaan minimum, Anda akan dikenakan biaya untuk batas tersebut meskipun penggunaan aktual Anda lebih rendah dari batas penggunaan minimum. ECPU atau penggunaan penyimpanan data yang melebihi batas penggunaan minimum dikenakan tarif reguler. Misalnya, jika Anda menetapkan batas penggunaan minimum 100.000 ECPU/detik maka Anda akan dikenakan biaya setidaknya $1,224 per jam (menggunakan harga ECPU di us-east-1), bahkan jika penggunaan Anda lebih rendah dari minimum yang ditetapkan.
> - ElastiCache Tanpa server mendukung skala minimum yang diminta pada tingkat agregat pada cache. ElastiCache Serverless juga mendukung maksimum 30K ECPU/detik per slot (90K ECPU/detik saat menggunakan Read from Replica menggunakan koneksi READONLY). Sebagai praktik terbaik, aplikasi Anda harus memastikan bahwa distribusi kunci di seluruh slot Redis dan lalu lintas lintas kunci seragam mungkin.

## Mengatur batas penskalaan menggunakan konsol dan AWS CLI

Menyetel batas penskalaan menggunakan Konsol AWS

1. Masuk ke AWS Management Console dan buka ElastiCache konsol di https://console.aws.amazon.com/elasticache/.
2. Pada panel navigasi, pilih mesin yang berjalan pada cache yang ingin diubah itu.

3. Muncul daftar cache yang menjalankan mesin yang dipilih.
4. Pilih cache untuk dimodifikasi dengan memilih tombol radio di sebelah kiri nama cache.
5. Pilih Tindakan, dan kemudian pilih Ubah.
6. Di bawah batas Penggunaan, tetapkan batas Memori atau Komputasi yang sesuai.
7. Klik Pratinjau perubahan dan kemudian Simpan perubahan.

### Menetapkan batas penskalaan menggunakan AWS CLI

Untuk mengubah batas penskalaan menggunakan CLI, gunakan modify-serverless-cache API.

Linux:

```
aws elasticache modify-serverless-cache --serverless-cache-name <cache name> \
--cache-usage-limits 'DataStorage={Minimum=10,Maximum=100,Unit=GB},
 ECPUPerSecond={Minimum=1000,Maximum=100000}'
```

Windows:

```
aws elasticache modify-serverless-cache --serverless-cache-name <cache name> ^
--cache-usage-limits 'DataStorage={Minimum=10,Maximum=100,Unit=GB},
 ECPUPerSecond={Minimum=1000,Maximum=100000}'
```

### Menghapus batas penskalaan menggunakan CLI

Untuk menghapus batas penskalaan menggunakan CLI, atur parameter batas Minimum dan Maksimum ke 0.

Linux:

```
aws elasticache modify-serverless-cache --serverless-cache-name <cache name> \
--cache-usage-limits 'DataStorage={Minimum=0,Maximum=0,Unit=GB},
 ECPUPerSecond={Minimum=0,Maximum=0}'
```

Windows:

```
aws elasticache modify-serverless-cache --serverless-cache-name <cache name> ^
--cache-usage-limits 'DataStorage={Minimum=0,Maximum=0,Unit=GB},
 ECPUPerSecond={Minimum=0,Maximum=0}'
```

# Penskalaan ElastiCache untuk cluster yang dirancang sendiri Redis

Jumlah data aplikasi Anda yang perlu diproses jarang statis. Hal ini meningkat dan menurun saat bisnis Anda tumbuh atau mengalami fluktuasi permintaan yang normal. Jika cache Anda dikelola sendiri, Anda perlu menyediakan perangkat keras yang memadai untuk puncak permintaan Anda, yang bisa jadi mahal. Dengan menggunakan Amazon, ElastiCache Anda dapat menskalakan untuk memenuhi permintaan saat ini, hanya membayar untuk apa yang Anda gunakan. ElastiCache memungkinkan Anda untuk menskalakan cache Anda agar sesuai dengan permintaan.

Berikut ini akan membantu Anda menemukan topik yang benar untuk tindakan penskalaan yang ingin Anda lakukan.

Menskalakan klaster Redis

| Tindakan | Redis (mode klaster dinonaktifkan) | Redis (mode klaster diaktifkan) |
|---|---|---|
| Menskalakan ke dalam | Menghapus simpul dari klaster | Penskalaan klaster di Redis (Mode Klaster Diaktifkan) |
| Menskalakan ke luar | Menambahkan simpul ke klaster | Resharding dan penyeimbangan ulang serpihan secara online untuk Redis (mode klaster diaktifkan) |
| Mengubah tipe simpul | Ke tipe simpul yang lebih besar:<br>• Menaikkan skala klaster simpul tunggal untuk Redis (Mode Klaster Dinonaktifkan)<br>• Menaikkan skala kluster Redis dengan replika<br><br>Ke tipe simpul yang lebih kecil:<br>• | Penskalaan vertikal online dengan mengubah jenis simpul |

| Tindakan | Redis (mode klaster dinonaktifkan) | Redis (mode klaster diaktifkan) |
| --- | --- | --- |
| | • Menurunkan skala klaster Redis simpul tunggal<br>• Menurunkan skala klaster Redis dengan replika | |
| Mengubah jumlah grup simpul | Tidak didukung untuk klaster Redis (mode klaster dinonaktifkan) | Penskalaan klaster di Redis (Mode Klaster Diaktifkan) |

Topik

- Penskalaan klaster untuk Redis (Mode Klaster Dinonaktifkan)
- Penskalaan klaster di Redis (Mode Klaster Diaktifkan)

# Penskalaan klaster untuk Redis (Mode Klaster Dinonaktifkan)

Klaster Redis (mode klaster dinonaktifkan) dapat berupa klaster simpul tunggal dengan 0 serpihan atau klaster multi-simpul dengan 1 serpihan. Klaster simpul tunggal menggunakan satu simpul baik untuk baca dan tulis. Klaster multi-simpul selalu memiliki 1 simpul sebagai simpul primer baca/tulis dengan 0 hingga 5 simpul replika hanya baca.

Daftar Isi



## Menskalakan klaster simpul tunggal untuk Redis (Mode klaster Dinonaktifkan)

Simpul Redis (mode klaster dinonaktifkan) harus cukup besar untuk menampung semua data cache serta overhead Redis. Untuk mengubah kapasitas data klaster Redis (mode klaster dinonaktifkan), Anda harus menskalakan secara vertikal; menaikkan skala ke jenis simpul yang lebih besar untuk meningkatkan kapasitas data, atau menurunkan skala ke jenis simpul yang lebih kecil untuk mengurangi kapasitas data.

Proses menaikkan skala ElastiCache for Redis dirancang untuk melakukan upaya terbaik guna mempertahankan data yang ada dan memerlukan replikasi Redis yang berhasil. Untuk klaster Redis (mode klaster dinonaktifkan), kami merekomendasikan agar memori yang cukup tersedia untuk Redis.

Anda tidak dapat mempartisi data Anda di beberapa klaster Redis (mode klaster dinonaktifkan). Namun, jika Anda hanya perlu untuk meningkatkan atau mengurangi kapasitas baca klaster, Anda dapat membuat klaster Redis (mode klaster dinonaktifkan) dengan simpul replika dan menambah atau menghapus replika baca. Untuk membuat klaster Redis (mode klaster dinonaktifkan) dengan simpul replika menggunakan klaster cache Redis simpul tunggal sebagai klaster primer, lihat Membuat klaster Redis (Mode Klaster Dinonaktifkan) (Konsol).

Setelah membuat klaster dengan replika, Anda dapat meningkatkan kapasitas baca dengan menambahkan replika baca. Jika diperlukan, Anda dapat mengurangi kapasitas baca dengan menghapus replika baca di lain waktu. Untuk informasi selengkapnya, lihat Meningkatkan kapasitas baca atau Mengurangi kapasitas baca.

Selain dapat menskalakan kapasitas baca, klaster Redis (mode klaster dinonaktifkan) dengan replika memberikan keuntungan bisnis lainnya. Untuk informasi selengkapnya, lihat Ketersediaan tinggi menggunakan grup replikasi.

> **Important**
>
> Jika grup parameter Anda menggunakan `reserved-memory` untuk menyisihkan memori untuk overhead Redis, sebelum mulai menskalakan pastikan bahwa Anda memiliki grup parameter khusus yang menyimpan jumlah memori yang benar untuk tipe simpul baru Anda. Atau, Anda dapat mengubah grup parameter khusus yang menggunakan `reserved-memory-percent` dan gunakan grup parameter tersebut untuk klaster baru Anda. Jika menggunakan `reserved-memory-percent`, Anda tidak perlu melakukan ini. Untuk informasi selengkapnya, lihat Mengelola Memori Terpesan.

Topik

- Menaikkan skala klaster simpul tunggal untuk Redis (Mode Klaster Dinonaktifkan)
- Menurunkan skala klaster Redis simpul tunggal

### Menaikkan skala klaster simpul tunggal untuk Redis (Mode Klaster Dinonaktifkan)

Ketika Anda menaikkan skala klaster Redis simpul tunggal, ElastiCache melakukan proses berikut, baik Anda menggunakan konsol ElastiCache, AWS CLI, atau API ElastiCache.

1. Klaster cache baru dengan tipe simpul baru berputar di Availability Zone yang sama dengan klaster cache yang ada.
2. Data cache dalam klaster cache yang ada disalin ke klaster cache baru. Berapa lama proses ini berjalan bergantung pada jenis simpul dan berapa banyak data dalam klaster cache Anda.
3. Membaca dan menulis sekarang dilayani menggunakan klaster cache baru. Karena titik akhir klaster cache baru sama seperti klaster cache lama, Anda tidak perlu memperbarui titik akhir dalam aplikasi Anda. Anda akan merasakan gangguan singkat (beberapa detik) pada baca dan tulis dari simpul primer saat entri DNS diperbarui.
4. ElastiCache menghapus klaster cache lama. Anda akan merasakan gangguan singkat (beberapa detik) pada baca dan tulis dari simpul lama karena koneksi ke simpul lama akan terputus.

> **Note**
>
> Untuk klaster yang menjalankan jenis simpul r6gd, Anda hanya dapat menskalakan ke ukuran simpul dalam keluarga simpul r6gd.

Seperti yang ditunjukkan dalam tabel berikut, operasi menaikkan skala Redis diblokir jika Anda memiliki peningkatan mesin terjadwal untuk jendela pemeliharaan berikutnya. Untuk informasi selengkapnya tentang Windows Pemeliharaan, lihat Mengelola pemeliharaan.

Operasi Redis Diblokir

| Operasi Tertunda | Operasi Diblokir |
|---|---|
| Menaikkan skala | Peningkatan mesin segera |
| Peningkatan mesin | Menaikkan skala segera |
| Menaikkan skala dan peningkatan mesin | Menaikkan skala segera |
| | Peningkatan mesin segera |

Jika memiliki operasi tertunda yang memblokir, Anda dapat melakukan salah satu hal berikut.

- Jadwalkan operasi menaikkan skala Redis Anda untuk jendela pemeliharaan berikutnya dengan menghapus kotak centang Terapkan segera (menggunakan CLI: `--no-apply-immediately`, menggunakan API: `ApplyImmediately=false`).
- Tunggu hingga jendela pemeliharaan berikutnya (atau setelahnya) untuk melakukan operasi menaikkan skala mesin Redis Anda.
- Tambahkan peningkatan mesin Redis ke pengubahan klaster cache ini dengan memilih kotak centang Terapkan Segera (menggunakan CLI: `--apply-immediately`, menggunakan API: `ApplyImmediately=true`). Ini akan membuka blokir operasi menaikkan skala dengan memicu peningkatan mesin segera dilakukan.

Anda dapat menaikkan skala klaster Redis (mode klaster dinonaktifkan) simpul tunggal menggunakan konsol ElastiCache, AWS CLI, atau API ElastiCache.

> **Important**
>
> Jika grup parameter Anda menggunakan `reserved-memory` untuk menyisihkan memori untuk overhead Redis, sebelum mulai menskalakan pastikan bahwa Anda memiliki grup parameter khusus yang menyimpan jumlah memori yang benar untuk tipe simpul baru Anda. Atau, Anda dapat mengubah grup parameter khusus yang menggunakan `reserved-memory-percent` dan gunakan grup parameter tersebut untuk klaster baru Anda. Jika menggunakan `reserved-memory-percent`, Anda tidak perlu melakukan ini. Untuk informasi selengkapnya, lihat Mengelola Memori Terpesan.

Menaikkan skala klaster simpul tunggal untuk Redis (Mode Klaster Dinonaktifkan) (Konsol)

Prosedur berikut menjelaskan cara menaikkan skala klaster Redis simpul tunggal menggunakan Konsol Manajemen ElastiCache. Selama proses ini, klaster Redis Anda akan terus melayani permintaan dengan waktu henti minimal.

Menaikkan skala klaster Redis simpul tunggal (konsol)

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Pada panel navigasi, silakan pilih Klaster Redis.

3. Dari daftar klaster, pilih klaster yang ingin Anda naikkan skalanya (harus menjalankan mesin Redis, bukan mesin Redis Berklaster).
4. Pilih Ubah.
5. Di wizard Ubah Klaster:

   a. Pilih tipe simpul yang ingin Anda skalakan dari daftar Tipe simpul.

   b. Jika Anda menggunakan `reserved-memory` untuk mengelola memori, dari daftar Grup Parameter, pilih grup parameter khusus yang menyimpan jumlah memori yang benar untuk jenis simpul baru.
6. Jika Anda ingin segera menaikkan skala, pilih kotak Terapkan segera. Jika kotak Terapkan segera tidak dipilih, proses menaikkan skala dilakukan selama jendela pemeliharaan berikutnya dari klaster ini.
7. Pilih Ubah.

   Jika Anda memilih Terapkan segera pada langkah sebelumnya, status klaster berubah ke sedang mengubah. Ketika status berubah ke tersedia, pengubahan selesai dan Anda dapat mulai menggunakan klaster baru tersebut.

Menaikkan skala klaster cache Redis simpul tunggal (AWS CLI)

Prosedur berikut menjelaskan cara menaikkan skala klaster cache Redis simpul tunggal menggunakan AWS CLI. Selama proses ini, klaster Redis Anda akan terus melayani permintaan dengan waktu henti minimal.

Menaikkan skala klaster cache Redis simpul tunggal (AWS CLI)

1. Tentukan tipe simpul mana yang dapat Anda menaikkan skalanya dengan menjalankan perintah AWS CLI `list-allowed-node-type-modifications` menggunakan parameter berikut.

   - `--cache-cluster-id`

   Untuk Linux, macOS, atau Unix:

   ```
   aws elasticache list-allowed-node-type-modifications \
        --cache-cluster-id my-cache-cluster-id
   ```

   Untuk Windows:

```
aws elasticache list-allowed-node-type-modifications ^
     --cache-cluster-id my-cache-cluster-id
```

Output dari perintah di atas terlihat seperti berikut (format JSON).

```
{
     "ScaleUpModifications": [
         "cache.m3.2xlarge",
         "cache.m3.large",
         "cache.m3.xlarge",
         "cache.m4.10xlarge",
         "cache.m4.2xlarge",
         "cache.m4.4xlarge",
         "cache.m4.large",
         "cache.m4.xlarge",
         "cache.r3.2xlarge",
         "cache.r3.4xlarge",
         "cache.r3.8xlarge",
         "cache.r3.large",
         "cache.r3.xlarge"
     ]
        "ScaleDownModifications": [
         "cache.t2.micro",
         "cache.t2.small ",
         "cache.t2.medium ",
            "cache.t1.small ",
     ],

 }
```

Untuk informasi selengkapnya, lihat list-allowed-node-type-modifications di dalam Referensi AWS CLI.

2. Ubah klaster cache yang ada untuk menentukan klaster cache yang akan dinaikkan skalanya dan tipe simpul baru yang lebih besar menggunakan perintah AWS CLI `modify-cache-cluster` dan parameter berikut.

   - `--cache-cluster-id` – Nama klaster cache yang dinaikkan skalanya.

- `--cache-node-type` – Jenis simpul baru yang akan Anda skalakan ke klaster cache. Nilai ini harus berupa salah satu dari tipe simpul yang dihasilkan oleh perintah `list-allowed-node-type-modifications` di langkah 1.
- `--cache-parameter-group-name` – [Opsional] Gunakan parameter ini jika Anda menggunakan `reserved-memory` untuk mengelola memori yang disimpan klaster. Tentukan grup parameter cache khusus yang mencadangkan jumlah memori yang sesuai untuk jenis simpul yang baru. Jika menggunakan `reserved-memory-percent`, Anda dapat menghilangkan parameter ini.
- `--apply-immediately` – Menyebabkan proses menaikkan skala segera diterapkan. Untuk menunda proses menaikkan skala ke jendela pemeliharaan berikutnya dari klaster, gunakan parameter `--no-apply-immediately`.

Untuk Linux, macOS, atau Unix:

```
aws elasticache modify-cache-cluster \
     --cache-cluster-id my-redis-cache-cluster \
     --cache-node-type cache.m3.xlarge \
     --cache-parameter-group-name redis32-m2-xl \
     --apply-immediately
```

Untuk Windows:

```
aws elasticache modify-cache-cluster ^
     --cache-cluster-id my-redis-cache-cluster ^
     --cache-node-type cache.m3.xlarge ^
     --cache-parameter-group-name redis32-m2-xl ^
     --apply-immediately
```

Output dari perintah di atas terlihat seperti berikut (format JSON).

```
{
     "CacheCluster": {
         "Engine": "redis",
         "CacheParameterGroup": {
             "CacheNodeIdsToReboot": [],
             "CacheParameterGroupName": "default.redis6.x",
             "ParameterApplyStatus": "in-sync"
         },
```

```
          "SnapshotRetentionLimit": 1,
          "CacheClusterId": "my-redis-cache-cluster",
          "CacheSecurityGroups": [],
          "NumCacheNodes": 1,
          "SnapshotWindow": "00:00-01:00",
          "CacheClusterCreateTime": "2017-02-21T22:34:09.645Z",
          "AutoMinorVersionUpgrade": true,
          "CacheClusterStatus": "modifying",
          "PreferredAvailabilityZone": "us-west-2a",
          "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
          "CacheSubnetGroupName": "default",
          "EngineVersion": "6.0",
          "PendingModifiedValues": {
              "CacheNodeType": "cache.m3.2xlarge"
          },
          "PreferredMaintenanceWindow": "tue:11:30-tue:12:30",
          "CacheNodeType": "cache.m3.medium",
           "DataTiering": "disabled"
      }
  }
```

Untuk informasi selengkapnya, lihat modify-cache-cluster di dalam Referensi AWS CLI.

3. Jika Anda menggunakan `--apply-immediately`, periksa status klaster cache baru menggunakan perintah AWS CLI `describe-cache-clusters` dengan parameter berikut. Ketika status berubah ke tersedia, Anda dapat mulai menggunakan klaster cache baru yang lebih besar.

   - `--cache-cache cluster-id` – Nama klaster cache Redis simpul tunggal Anda. Gunakan parameter ini untuk mendeskripsikan klaster cache tertentu bukannya semua klaster cache.

```
aws elasticache describe-cache-clusters --cache-cluster-id my-redis-cache-cluster
```

Untuk informasi selengkapnya, lihat describe-cache-clusters di dalam Referensi AWS CLI.

Menaikkan skala klaster cache Redis simpul tunggal (API ElastiCache)

Prosedur berikut menjelaskan cara menaikkan skala klaster cache Redis simpul tunggal menggunakan API ElastiCache. Selama proses ini, klaster Redis Anda akan terus melayani permintaan dengan waktu henti minimal.

Menaikkan skala klaster cache Redis simpul tunggal (API ElastiCache)

1. Tentukan jenis simpul mana yang dapat Anda naikkan skalanya menggunakan tindakan `ListAllowedNodeTypeModifications` API ElastiCache dengan parameter berikut.

   - `CacheClusterId` – Nama klaster cache Redis simpul tunggal yang ingin Anda naikkan skalanya.

   ```
   https://elasticache.us-west-2.amazonaws.com/
       ?Action=ListAllowedNodeTypeModifications
       &CacheClusterId=MyRedisCacheCluster
       &Version=2015-02-02
       &SignatureVersion=4
       &SignatureMethod=HmacSHA256
       &Timestamp=20150202T192317Z
       &X-Amz-Credential=<credential>
   ```

   Untuk informasi selengkapnya, lihat ListAllowedNodeTypeModifications dalam Referensi API Amazon ElastiCache.

2. Ubah klaster cache yang ada untuk menentukan klaster cache yang akan dinaikkan skalanya dan tipe simpul baru yang lebih besar menggunakan tindakan `ModifyCacheCluster` dan parameter berikut.

   - `CacheClusterId` – Nama klaster cache yang dinaikkan skalanya.
   - `CacheNodeType` – Jenis simpul baru yang Anda ingin untuk menskalakan naik klaster cachenya. Nilai ini harus berupa salah satu dari tipe simpul yang dihasilkan oleh tindakan `ListAllowedNodeTypeModifications` di langkah 1.
   - `CacheParameterGroupName` – [Opsional] Gunakan parameter ini jika Anda menggunakan `reserved-memory` untuk mengelola memori yang disimpan klaster. Tentukan grup parameter cache khusus yang mencadangkan jumlah memori yang sesuai untuk jenis simpul yang baru. Jika menggunakan `reserved-memory-percent`, Anda dapat menghilangkan parameter ini.

- `ApplyImmediately` – Atur ke `true` untuk menyebabkan proses menaikkan skala segera dilakukan. Untuk menunda proses menaikkan skala ke jendela pemeliharaan berikutnya dari klaster, gunakan `ApplyImmediately=false`.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=ModifyCacheCluster
    &ApplyImmediately=true
    &CacheClusterId=MyRedisCacheCluster
    &CacheNodeType=cache.m3.xlarge
    &CacheParameterGroupName redis32-m2-xl
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &X-Amz-Credential=<credential>
```

Untuk informasi selengkapnya, lihat ModifyCacheCluster dalam Referensi API Amazon ElastiCache.

3. Jika Anda menggunakan `ApplyImmediately=true`, periksa status klaster cache baru menggunakan tindakan `DescribeCacheClusters` dengan parameter berikut. Ketika status berubah ke tersedia, Anda dapat mulai menggunakan klaster cache baru yang lebih besar.

   - `CacheClusterId` – Nama klaster cache Redis simpul tunggal Anda. Gunakan parameter ini untuk mendeskripsikan klaster cache tertentu bukannya semua klaster cache.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeCacheClusters
    &CacheClusterId=MyRedisCacheCluster
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &X-Amz-Credential=<credential>
```

Untuk informasi selengkapnya, lihat DescribeCacheClusters dalam Referensi API Amazon ElastiCache.

## Menurunkan skala klaster Redis simpul tunggal

Bagian berikut memandu Anda dalam cara menskalakan klaster Redis simpul tunggal ke bawah ke tipe simpul yang lebih kecil. Memastikan bahwa tipe simpul baru yang lebih kecil cukup besar untuk mengakomodasi semua data dan overhead Redis penting untuk keberhasilan jangka panjang klaster Redis baru Anda. Untuk informasi selengkapnya, lihat Memastikan bahwa Anda memiliki cukup memori untuk membuat snapshot Redis.

> **Note**
>
> Untuk klaster yang menjalankan jenis simpul r6gd, Anda hanya dapat menskalakan ke ukuran simpul dalam keluarga simpul r6gd.

Topik

- menurunkan skala klaster Redis simpul tunggal (Konsol)
- Menurunkan skala klaster cache Redis simpul tunggal (AWS CLI)
- Menurunkan skala klaster cache Redis simpul tunggal (API ElastiCache)

### menurunkan skala klaster Redis simpul tunggal (Konsol)

Prosedur berikut memandu Anda dalam menskalakan klaster Redis simpul tunggal ke bawah ke jenis simpul yang lebih kecil menggunakan konsol ElastiCache.

> **Important**
>
> Jika grup parameter Anda menggunakan `reserved-memory` untuk menyisihkan memori untuk overhead Redis, sebelum mulai menskalakan pastikan bahwa Anda memiliki grup parameter khusus yang menyimpan jumlah memori yang benar untuk tipe simpul baru Anda. Atau, Anda dapat mengubah grup parameter khusus yang menggunakan `reserved-memory-percent` dan gunakan grup parameter tersebut untuk klaster baru Anda. Jika menggunakan `reserved-memory-percent`, Anda tidak perlu melakukan ini. Untuk informasi selengkapnya, lihat Mengelola Memori Terpesan.

#### Menurunkan skala klaster Redis simpul tunggal (Konsol)

1. Pastikan bahwa tipe simpul yang lebih kecil memadai untuk data dan kebutuhan overhead Anda.

2. Jika grup parameter Anda menggunakan `reserved-memory` untuk menyisihkan memori untuk overhead Redis, sebelum mulai menskalakan pastikan bahwa Anda memiliki grup parameter khusus yang menyimpan jumlah memori yang benar untuk tipe simpul baru Anda.

   Atau, Anda dapat mengubah grup parameter khusus untuk menggunakan `reserved-memory-percent`. Untuk informasi selengkapnya, lihat Mengelola Memori Terpesan.

3. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.

4. Pada daftar klaster, pilih klaster yang ingin Anda turunkan skalanya. Klaster ini harus menjalankan mesin Redis dan bukan mesin Redis Berklaster.

5. Pilih Ubah.

6. Di wizard Ubah Klaster:

   a. Pilih tipe simpul yang ingin Anda turunkan skalanya dari daftar Tipe simpul.

   b. Jika Anda menggunakan `reserved-memory` untuk mengelola memori, dari daftar Grup Parameter, pilih grup parameter khusus yang menyimpan jumlah memori yang benar untuk jenis simpul baru.

7. Jika Anda ingin segera menurunkan skala, pilih kotak centang Terapkan segera. Jika kotak centang Terapkan segera tidak dipilih, proses menurunkan skala dilakukan selama jendela pemeliharaan berikutnya dari klaster ini.

8. Pilih Ubah.

9. Ketika status klaster berubah dari mengubah ke tersedia, klaster Anda telah diskalakan ke tipe simpul baru. Tidak perlu memperbarui titik akhir dalam aplikasi Anda.

Menurunkan skala klaster cache Redis simpul tunggal (AWS CLI)

Prosedur berikut menjelaskan cara menurunkan skala klaster cache Redis simpul tunggal menggunakan AWS CLI.

Menurunkan skala klaster cache Redis simpul tunggal (AWS CLI)

1. Tentukan tipe simpul mana yang dapat Anda menurunkan skalanya dengan menjalankan perintah AWS CLI `list-allowed-node-type-modifications` menggunakan parameter berikut.

   - `--cache-cluster-id`

Untuk Linux, macOS, atau Unix:

```
aws elasticache list-allowed-node-type-modifications \
     --cache-cluster-id my-cache-cluster-id
```

Untuk Windows:

```
aws elasticache list-allowed-node-type-modifications ^
     --cache-cluster-id my-cache-cluster-id
```

Output dari perintah di atas terlihat seperti berikut (format JSON).

```
{
     "ScaleUpModifications": [
         "cache.m3.2xlarge",
         "cache.m3.large",
         "cache.m3.xlarge",
         "cache.m4.10xlarge",
         "cache.m4.2xlarge",
         "cache.m4.4xlarge",
         "cache.m4.large",
         "cache.m4.xlarge",
         "cache.r3.2xlarge",
         "cache.r3.4xlarge",
         "cache.r3.8xlarge",
         "cache.r3.large",
         "cache.r3.xlarge"
     ]
        "ScaleDownModifications": [
         "cache.t2.micro",
         "cache.t2.small ",
         "cache.t2.medium ",
            "cache.t1.small ",
     ],

 }
```

Untuk informasi selengkapnya, lihat list-allowed-node-type-modifications di dalam Referensi AWS CLI.

2. Ubah klaster cache yang ada untuk menentukan klaster cache yang akan diturunkan skalanya dan tipe simpul baru yang lebih kecil menggunakan perintah AWS CLI `modify-cache-cluster` dan parameter berikut.

   - `--cache-cluster-id` – Nama klaster cache yang diturunkan skalanya.
   - `--cache-node-type` – Jenis simpul baru yang akan Anda skalakan ke klaster cache. Nilai ini harus berupa salah satu dari tipe simpul yang dihasilkan oleh perintah `list-allowed-node-type-modifications` di langkah 1.
   - `--cache-parameter-group-name` – [Opsional] Gunakan parameter ini jika Anda menggunakan `reserved-memory` untuk mengelola memori yang disimpan klaster. Tentukan grup parameter cache khusus yang mencadangkan jumlah memori yang sesuai untuk jenis simpul yang baru. Jika menggunakan `reserved-memory-percent`, Anda dapat menghilangkan parameter ini.
   - `--apply-immediately` – Menyebabkan proses menurunkan skala segera diterapkan. Untuk menunda proses menaikkan skala ke jendela pemeliharaan berikutnya dari klaster, gunakan parameter `--no-apply-immediately`.

   Untuk Linux, macOS, atau Unix:

   ```
   aws elasticache modify-cache-cluster \
        --cache-cluster-id my-redis-cache-cluster \
        --cache-node-type cache.m3.xlarge \
        --cache-parameter-group-name redis32-m2-xl \
        --apply-immediately
   ```

   Untuk Windows:

   ```
   aws elasticache modify-cache-cluster ^
        --cache-cluster-id my-redis-cache-cluster ^
        --cache-node-type cache.m3.xlarge ^
        --cache-parameter-group-name redis32-m2-xl ^
        --apply-immediately
   ```

   Output dari perintah di atas terlihat seperti berikut (format JSON).

   ```
   {
        "CacheCluster": {
            "Engine": "redis",
   ```

```
            "CacheParameterGroup": {
                "CacheNodeIdsToReboot": [],
                "CacheParameterGroupName": "default.redis6,x",
                "ParameterApplyStatus": "in-sync"
            },
            "SnapshotRetentionLimit": 1,
            "CacheClusterId": "my-redis-cache-cluster",
            "CacheSecurityGroups": [],
            "NumCacheNodes": 1,
            "SnapshotWindow": "00:00-01:00",
            "CacheClusterCreateTime": "2017-02-21T22:34:09.645Z",
            "AutoMinorVersionUpgrade": true,
            "CacheClusterStatus": "modifying",
            "PreferredAvailabilityZone": "us-west-2a",
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "CacheSubnetGroupName": "default",
            "EngineVersion": "6.0",
            "PendingModifiedValues": {
                "CacheNodeType": "cache.m3.2xlarge"
            },
            "PreferredMaintenanceWindow": "tue:11:30-tue:12:30",
            "CacheNodeType": "cache.m3.medium",
             "DataTiering": "disabled"
        }
    }
```

Untuk informasi selengkapnya, lihat modify-cache-cluster di dalam Referensi AWS CLI.

3. Jika Anda menggunakan `--apply-immediately`, periksa status klaster cache baru menggunakan perintah AWS CLI `describe-cache-clusters` dengan parameter berikut. Ketika status berubah ke tersedia, Anda dapat mulai menggunakan klaster cache baru yang lebih besar.

   - `--cache-cache cluster-id` – Nama klaster cache Redis simpul tunggal Anda. Gunakan parameter ini untuk mendeskripsikan klaster cache tertentu bukannya semua klaster cache.

```
aws elasticache describe-cache-clusters --cache-cluster-id my-redis-cache-cluster
```

Untuk informasi selengkapnya, lihat describe-cache-clusters di dalam Referensi AWS CLI.

Menurunkan skala klaster cache Redis simpul tunggal (API ElastiCache)

Prosedur berikut menjelaskan cara menurunkan skala klaster cache Redis simpul tunggal menggunakan API ElastiCache.

Menurunkan skala klaster cache Redis simpul tunggal (API ElastiCache)

1. Tentukan jenis simpul mana yang dapat Anda turunkan skalanya menggunakan tindakan `ListAllowedNodeTypeModifications` API ElastiCache dengan parameter berikut.

   - `CacheClusterId` – Nama klaster cache Redis simpul tunggal yang ingin Anda turunkan skalanya.

   ```
   https://elasticache.us-west-2.amazonaws.com/
       ?Action=ListAllowedNodeTypeModifications
       &CacheClusterId=MyRedisCacheCluster
       &Version=2015-02-02
       &SignatureVersion=4
       &SignatureMethod=HmacSHA256
       &Timestamp=20150202T192317Z
       &X-Amz-Credential=<credential>
   ```

   Untuk informasi selengkapnya, lihat ListAllowedNodeTypeModifications dalam Referensi API Amazon ElastiCache.

2. Ubah klaster cache yang ada untuk menentukan klaster cache yang akan dinaikkan skalanya dan tipe simpul baru yang lebih besar menggunakan tindakan `ModifyCacheCluster` dan parameter berikut.

   - `CacheClusterId` – Nama klaster cache yang diturunkan skalanya.
   - `CacheNodeType` – Jenis simpul baru yang Anda inginkan untuk menurunkan skala klaster cache. Nilai ini harus berupa salah satu dari tipe simpul yang dihasilkan oleh tindakan `ListAllowedNodeTypeModifications` di langkah 1.
   - `CacheParameterGroupName` – [Opsional] Gunakan parameter ini jika Anda menggunakan `reserved-memory` untuk mengelola memori yang disimpan klaster. Tentukan grup parameter cache khusus yang mencadangkan jumlah memori yang sesuai untuk jenis simpul yang baru. Jika menggunakan `reserved-memory-percent`, Anda dapat menghilangkan parameter ini.

- `ApplyImmediately` – Atur ke `true` untuk menyebabkan proses menurunkan skala segera dilakukan. Untuk menunda proses menaikkan skala ke jendela pemeliharaan berikutnya dari klaster, gunakan `ApplyImmediately=false`.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=ModifyCacheCluster
    &ApplyImmediately=true
    &CacheClusterId=MyRedisCacheCluster
    &CacheNodeType=cache.m3.xlarge
    &CacheParameterGroupName redis32-m2-xl
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &X-Amz-Credential=<credential>
```

Untuk informasi selengkapnya, lihat ModifyCacheCluster dalam Referensi API Amazon ElastiCache.

3. Jika Anda menggunakan `ApplyImmediately=true`, periksa status klaster cache baru menggunakan tindakan `DescribeCacheClusters` dengan parameter berikut. Ketika status berubah ke tersedia, Anda dapat mulai menggunakan klaster cache baru yang lebih kecil.

   - `CacheClusterId` – Nama klaster cache Redis simpul tunggal Anda. Gunakan parameter ini untuk mendeskripsikan klaster cache tertentu bukannya semua klaster cache.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeCacheClusters
    &CacheClusterId=MyRedisCacheCluster
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &X-Amz-Credential=<credential>
```

Untuk informasi selengkapnya, lihat DescribeCacheClusters dalam Referensi API Amazon ElastiCache.

## Penskalaan Redis (Mode Cluster Dinonaktifkan) cluster dengan simpul replika

Klaster Redis dengan simpul replika (disebut grup replikasi di API/CLI) menyediakan ketersediaan tinggi melalui replikasi yang memiliki Multi-AZ dengan failover otomatis diaktifkan. Klaster dengan simpul replika adalah kumpulan logis hingga enam simpul Redis yang mana satu simpul, Primer, mampu melayani baik permintaan baca dan tulis. Semua simpul lain dalam klaster adalah replika baca-saja dari Primer. Data yang ditulis ke Primer direplikasi secara asinkron ke semua replika baca di klaster. Karena Redis (mode klaster dinonaktifkan) tidak mendukung partisi data Anda di beberapa klaster, setiap simpul dalam grup replikasi Redis (mode klaster dinonaktifkan) berisi seluruh set data cache. Klaster Redis (mode klaster diaktifkan) mendukung pembuatan partisi data Anda secara menyeluruh hingga 500 serpihan.

Untuk mengubah kapasitas data klaster, Anda harus menskalakan ke atas ke tipe simpul yang lebih besar, atau menskalakan ke bawah ke tipe simpul yang lebih kecil.

Untuk mengubah kapasitas baca klaster Anda, tambahkan lebih banyak replika baca, hingga maksimum 5, atau hapus replika baca.

Proses menaikkan skala ElastiCache dirancang untuk melakukan upaya terbaik guna mempertahankan data yang ada dan memerlukan replikasi Redis yang berhasil. Untuk klaster Redis dengan replika, kami merekomendasikan agar memori yang cukup tersedia untuk Redis.

Topik-Topik Terkait

- Ketersediaan tinggi menggunakan grup replikasi
- Replikasi: Redis (Mode Klaster Dinonaktifkan) vs Redis (Mode Klaster Diaktifkan)
- Meminimalkan waktu henti di ElastiCache for Redis dengan Multi-AZ
- Memastikan bahwa Anda memiliki cukup memori untuk membuat snapshot Redis

Topik

- Menaikkan skala kluster Redis dengan replika
- Menurunkan skala klaster Redis dengan replika
- Meningkatkan kapasitas baca
- Mengurangi kapasitas baca

### Menaikkan skala kluster Redis dengan replika

Amazon ElastiCache menyediakan konsol, CLI, dan dukungan API untuk menskalakan grup replikasi Redis (mode klaster dinonaktifkan) ke atas.

Ketika proses menaikkan skala dimulai, ElastiCache melakukan hal berikut:

1. Meluncurkan grup replikasi menggunakan tipe simpul baru.
2. Menyalin semua data dari simpul primer saat ini ke simpul primer baru.
3. Menyinkronkan replika baca baru dengan simpul primer baru.
4. Memperbarui entri DNS sehingga menunjuk ke simpul baru. Karena hal ini, Anda tidak perlu memperbarui titik akhir dalam aplikasi Anda. Untuk Redis 5.0.5 dan setelahnya, Anda dapat menskalakan klaster dengan failover otomatis diaktifkan saat klaster terus online dan melayani permintaan masuk. Pada versi 4.0.10 dan sebelumnya, Anda mungkin merasakan gangguan singkat baca dan tulis pada versi sebelumnya dari simpul primer saat entri DNS diperbarui.
5. Menghapus simpul lama (CLI/API: grup replikasi). Anda akan merasakan gangguan singkat (beberapa detik) pada baca dan tulis dari simpul lama karena koneksi ke simpul lama akan terputus.

Berapa lama proses ini berjalan bergantung pada jenis simpul dan berapa banyak data dalam klaster Anda.

Seperti yang ditunjukkan dalam tabel berikut, operasi menaikkan skala Redis diblokir jika Anda memiliki peningkatan mesin terjadwal untuk jendela pemeliharaan berikutnya dari klaster.

Operasi Redis Diblokir

| Operasi Tertunda | Operasi Diblokir |
|---|---|
| Menaikkan skala | Peningkatan mesin segera |
| Peningkatan mesin | Menaikkan skala segera |
| menaikkan skala dan peningkatan mesin | Menaikkan skala segera |
| | Peningkatan mesin segera |

Jika memiliki operasi tertunda yang memblokir, Anda dapat melakukan salah satu hal berikut.

- Jadwalkan operasi menaikkan skala Redis Anda untuk jendela pemeliharaan berikutnya dengan menghapus kotak centang Terapkan segera (menggunakan CLI: `--no-apply-immediately`, menggunakan API: `ApplyImmediately=false`).
- Tunggu hingga jendela pemeliharaan berikutnya (atau setelahnya) untuk melakukan operasi menaikkan skala mesin Redis Anda.
- Tambahkan peningkatan mesin Redis ke pengubahan klaster cache ini dengan memilih kotak centang Terapkan Segera (menggunakan CLI: `--apply-immediately`, menggunakan API: `ApplyImmediately=true`). Ini akan membuka blokir operasi menaikkan skala dengan memicu peningkatan mesin segera dilakukan.

Bagian berikut menjelaskan cara menskalakan klaster Redis Anda dengan replika ke atas menggunakan konsol ElastiCache, AWS CLI, dan API ElastiCache.

> **Important**
>
> Jika grup parameter Anda menggunakan `reserved-memory` untuk menyisihkan memori untuk overhead Redis, sebelum mulai menskalakan pastikan bahwa Anda memiliki grup parameter khusus yang menyimpan jumlah memori yang benar untuk tipe simpul baru Anda. Atau, Anda dapat mengubah grup parameter khusus yang menggunakan `reserved-memory-percent` dan gunakan grup parameter tersebut untuk klaster baru Anda. Jika menggunakan `reserved-memory-percent`, Anda tidak perlu melakukan ini. Untuk informasi selengkapnya, lihat Mengelola Memori Terpesan.

### Menaikkan skala klaster Redis dengan replika (Konsol)

Jumlah waktu yang dibutuhkan untuk menaikkan skala ke tipe simpul yang lebih besar bervariasi, bergantung pada tipe simpul dan jumlah data dalam klaster cache Anda saat ini.

Proses berikut menskalakan klaster Anda dengan replika dari jenis simpul saat ini ke jenis simpul baru yang lebih besar menggunakan konsol ElastiCache. Selama proses ini, mungkin ada gangguan singkat baca dan tulis untuk versi lain dari simpul primer saat entri DNS diperbarui. Anda mungkin melihat waktu henti kurang dari 1 detik untuk simpul yang berjalan pada versi 5.0.6 ke atas dan beberapa detik untuk versi yang lebih lama.

### Menaikkan skala klaster Redis dengan replika (Konsol)

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Pada panel navigasi, silakan pilih Klaster Redis
3. Pada daftar klaster, pilih klaster yang ingin Anda naikkan skalanya. Klaster ini harus menjalankan mesin Redis dan bukan mesin Redis Berklaster.
4. Pilih Ubah.
5. Di wizard Ubah Klaster:
   a. Pilih tipe simpul yang ingin Anda skalakan dari daftar Tipe simpul. Perhatikan bahwa tidak semua tipe simpul yang tersedia untuk diturunkan skalanya.
   b. Jika Anda menggunakan `reserved-memory` untuk mengelola memori, dari daftar Grup Parameter, pilih grup parameter khusus yang menyimpan jumlah memori yang benar untuk jenis simpul baru.
6. Jika Anda ingin segera menaikkan skala, pilih kotak centang Terapkan segera. Jika kotak centang Terapkan segera tidak dipilih, proses menaikkan skala dilakukan selama jendela pemeliharaan berikutnya dari klaster ini.
7. Pilih Ubah.
8. Ketika status klaster berubah dari mengubah ke tersedia, klaster Anda telah diskalakan ke tipe simpul baru. Tidak perlu memperbarui titik akhir dalam aplikasi Anda.

### Menaikkan skala grup replikasi Redis (AWS CLI)

Proses berikut menskalakan grup replikasi Anda dari tipe simpul saat ini ke tipe simpul baru yang lebih besar menggunakan AWS CLI. Selama proses ini, ElastiCache for Redis memperbarui entri DNS sehingga menunjuk ke simpul baru. Karena hal ini, Anda tidak perlu memperbarui titik akhir dalam aplikasi Anda. Untuk Redis 5.0.5 dan setelahnya, Anda dapat menskalakan klaster dengan failover otomatis diaktifkan saat klaster terus online dan melayani permintaan masuk. Pada versi 4.0.10 dan sebelumnya, Anda mungkin merasakan gangguan singkat baca dan tulis pada versi sebelumnya dari simpul primer saat entri DNS diperbarui.

Jumlah waktu yang dibutuhkan untuk menaikkan skala ke tipe simpul yang lebih besar bervariasi, bergantung pada tipe simpul dan jumlah data dalam klaster cache Anda saat ini.

Menaikkan skala Grup Replikasi Redis (AWS CLI)

1. Tentukan tipe simpul mana yang dapat Anda naikkan skalanya dengan menjalankan perintah AWS CLI `list-allowed-node-type-modifications` menggunakan parameter berikut.

   - `--replication-group-id` – nama grup replikasi. Gunakan parameter ini untuk mendeskripsikan grup replikasi tertentu bukannya semua grup replikasi.

   Untuk Linux, macOS, atau Unix:

   ```
   aws elasticache list-allowed-node-type-modifications \
        --replication-group-id my-repl-group
   ```

   Untuk Windows:

   ```
   aws elasticache list-allowed-node-type-modifications ^
        --replication-group-id my-repl-group
   ```

   Output dari operasi ini terlihat seperti berikut (format JSON).

   ```
   {
        "ScaleUpModifications": [
            "cache.m3.2xlarge",
            "cache.m3.large",
            "cache.m3.xlarge",
            "cache.m4.10xlarge",
            "cache.m4.2xlarge",
            "cache.m4.4xlarge",
            "cache.m4.large",
            "cache.m4.xlarge",
            "cache.r3.2xlarge",
            "cache.r3.4xlarge",
            "cache.r3.8xlarge",
            "cache.r3.large",
            "cache.r3.xlarge"
        ]
    }
   ```

   Untuk informasi selengkapnya, lihat list-allowed-node-type-modifications di dalam Referensi AWS CLI.

2. Skalakan grup replikasi Anda saat ini ke atas ke tipe simpul baru menggunakan perintah AWS CLI `modify-replication-group` dengan parameter berikut.

   - `--replication-group-id` – nama grup replikasi.
   - `--cache-node-type` – jenis simpul baru yang lebih besar dari klaster cache dalam grup replikasi ini. Nilai ini harus berupa salah satu dari tipe instans yang dihasilkan oleh perintah `list-allowed-node-type-modifications` di langkah 1.
   - `--cache-parameter-group-name` – [Opsional] Gunakan parameter ini jika Anda menggunakan `reserved-memory` untuk mengelola memori yang disimpan klaster. Tentukan grup parameter cache khusus yang mencadangkan jumlah memori yang sesuai untuk jenis simpul yang baru. Jika menggunakan `reserved-memory-percent`, Anda dapat menghilangkan parameter ini.
   - `--apply-immediately` – Menyebabkan proses menaikkan skala segera diterapkan. Untuk menunda operasi menaikkan skala ke jendela pemeliharaan berikutnya, gunakan `--no-apply-immediately`.

   Untuk Linux, macOS, atau Unix:

   ```
   aws elasticache modify-replication-group \
        --replication-group-id my-repl-group \
        --cache-node-type cache.m3.xlarge \
        --cache-parameter-group-name redis32-m3-2xl \
        --apply-immediately
   ```

   Untuk Windows:

   ```
   aws elasticache modify-replication-group ^
        --replication-group-id my-repl-group ^
        --cache-node-type cache.m3.xlarge ^
        --cache-parameter-group-name redis32-m3-2xl \
        --apply-immediately
   ```

   Output dari operasi ini terlihat seperti berikut (format JSON).

   ```
   {
    "ReplicationGroup": {
     "Status": "available",
   ```

```
  "Description": "Some description",
  "NodeGroups": [{
   "Status": "available",
   "NodeGroupMembers": [{
     "CurrentRole": "primary",
     "PreferredAvailabilityZone": "us-west-2b",
     "CacheNodeId": "0001",
     "ReadEndpoint": {
      "Port": 6379,
      "Address": "my-repl-group-001.8fdx4s.0001.usw2.cache.amazonaws.com"
     },
     "CacheClusterId": "my-repl-group-001"
    },
    {
     "CurrentRole": "replica",
     "PreferredAvailabilityZone": "us-west-2c",
     "CacheNodeId": "0001",
     "ReadEndpoint": {
      "Port": 6379,
      "Address": "my-repl-group-002.8fdx4s.0001.usw2.cache.amazonaws.com"
     },
     "CacheClusterId": "my-repl-group-002"
    }
   ],
   "NodeGroupId": "0001",
   "PrimaryEndpoint": {
    "Port": 6379,
    "Address": "my-repl-group.8fdx4s.ng.0001.usw2.cache.amazonaws.com"
   }
  }],
  "ReplicationGroupId": "my-repl-group",
  "SnapshotRetentionLimit": 1,
  "AutomaticFailover": "disabled",
  "SnapshotWindow": "12:00-13:00",
  "SnapshottingClusterId": "my-repl-group-002",
  "MemberClusters": [
   "my-repl-group-001",
   "my-repl-group-002"
  ],
  "PendingModifiedValues": {}
 }
}
```

Untuk informasi selengkapnya, lihat modify-replication-group di dalam Referensi AWS CLI.

3. Jika Anda menggunakan parameter `--apply-immediately`, pantau status grup replikasi menggunakan perintah AWS CLI `describe-replication-group` dengan parameter berikut. Saat status masih mengubah, Anda mungkin melihat waktu henti kurang dari 1 detik untuk simpul yang berjalan di versi 5.0.6 ke atas dan gangguan singkat baca dan tulis untuk versi yang lebih lama dari simpul primer saat entri DNS diperbarui.

   - `--replication-group-id` – nama grup replikasi. Gunakan parameter ini untuk mendeskripsikan grup replikasi tertentu bukannya semua grup replikasi.

   Untuk Linux, macOS, atau Unix:

   ```
   aws elasticache describe-replication-groups \
        --replication-group-id my-replication-group
   ```

   Untuk Windows:

   ```
   aws elasticache describe-replication-groups ^
        --replication-group-id my-replication-group
   ```

   Untuk informasi selengkapnya, lihat describe-replication-groups dalam Referensi AWS CLI.

Menaikkan skala grup replikasi Redis (API ElastiCache)

Proses berikut menskalakan grup replikasi Anda dari tipe simpul saat ini ke tipe simpul baru yang lebih besar menggunakan API ElastiCache. Untuk Redis 5.0.5 dan setelahnya, Anda dapat menskalakan klaster dengan failover otomatis diaktifkan saat klaster terus online dan melayani permintaan masuk. Pada versi 4.0.10 dan sebelumnya, Anda mungkin merasakan gangguan singkat baca dan tulis pada versi sebelumnya dari simpul primer saat entri DNS diperbarui.

Jumlah waktu yang dibutuhkan untuk menaikkan skala ke tipe simpul yang lebih besar bervariasi, bergantung pada tipe simpul dan jumlah data dalam klaster cache Anda saat ini.

Untuk menaikkan skala Grup Replikasi Redis (API ElastiCache)

1. Tentukan jenis simpul mana yang dapat Anda naikkan skalanya menggunakan tindakan `ListAllowedNodeTypeModifications` API ElastiCache dengan parameter berikut.

- `ReplicationGroupId` – nama grup replikasi. Gunakan parameter ini untuk mendeskripsikan grup replikasi tertentu, bukan semua grup replikasi.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=ListAllowedNodeTypeModifications
    &ReplicationGroupId=MyReplGroup
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &X-Amz-Credential=<credential>
```

Untuk informasi selengkapnya, lihat ListAllowedNodeTypeModifications dalam Referensi API Amazon ElastiCache.

2. Skalakan grup replikasi Anda saat ini ke atas ke tipe simpul baru menggunakan tindakan `ModifyRedplicationGroup` API ElastiCache dan dengan parameter berikut.

   - `ReplicationGroupId` – nama grup replikasi.
   - `CacheNodeType` – jenis simpul baru yang lebih besar dari klaster cache dalam grup replikasi ini. Nilai ini harus berupa salah satu dari tipe instans yang dihasilkan oleh tindakan `ListAllowedNodeTypeModifications` di langkah 1.
   - `CacheParameterGroupName` – [Opsional] Gunakan parameter ini jika Anda menggunakan `reserved-memory` untuk mengelola memori yang disimpan klaster. Tentukan grup parameter cache khusus yang mencadangkan jumlah memori yang sesuai untuk jenis simpul yang baru. Jika menggunakan `reserved-memory-percent`, Anda dapat menghilangkan parameter ini.
   - `ApplyImmediately` – Atur ke `true` untuk menyebabkan proses menaikkan skala segera diterapkan. Untuk menunda proses menaikkan skala ke jendela pemeliharaan berikutnya, gunakan `ApplyImmediately=false`.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=ModifyReplicationGroup
    &ApplyImmediately=true
    &CacheNodeType=cache.m3.2xlarge
    &CacheParameterGroupName=redis32-m3-2xl
    &ReplicationGroupId=myReplGroup
    &SignatureVersion=4
```

```
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &Version=2014-12-01
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20141201T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

Untuk informasi selengkapnya, lihat ModifyReplicationGroup dalam Referensi API Amazon ElastiCache.

3. Jika Anda menggunakan `ApplyImmediately=true`, pantau status grup replikasi menggunakan tindakan `DescribeReplicationGroups` API ElastiCache dengan parameter berikut. Ketika status berubah dari mengubah ke tersedia, Anda dapat mulai menulis ke grup replikasi baru Anda yang dinaikkan skalanya.

   - `ReplicationGroupId` – nama grup replikasi. Gunakan parameter ini untuk mendeskripsikan grup replikasi tertentu bukannya semua grup replikasi.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeReplicationGroups
    &ReplicationGroupId=MyReplGroup
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &X-Amz-Credential=<credential>
```

Untuk informasi selengkapnya, lihat DescribeReplicationGroups dalam Referensi API Amazon ElastiCache.

### Menurunkan skala klaster Redis dengan replika

Bagian berikut memandu Anda dalam cara menskalakan klaster cache Redis (mode klaster dinonaktifkan) dengan simpul replika ke bawah ke jenis simpul yang lebih kecil. Memastikan bahwa tipe simpul baru yang lebih kecil cukup besar untuk mengakomodasi semua data dan overhead sangat penting untuk keberhasilan. Untuk informasi selengkapnya, lihat Memastikan bahwa Anda memiliki cukup memori untuk membuat snapshot Redis.

> **Note**
>
> Untuk klaster yang menjalankan jenis simpul r6gd, Anda hanya dapat menskalakan ke ukuran simpul dalam keluarga simpul r6gd.

> **Important**
>
> Jika grup parameter Anda menggunakan `reserved-memory` untuk menyisihkan memori untuk overhead Redis, sebelum mulai menskalakan pastikan bahwa Anda memiliki grup parameter khusus yang menyimpan jumlah memori yang benar untuk tipe simpul baru Anda. Atau, Anda dapat mengubah grup parameter khusus yang menggunakan `reserved-memory-percent` dan gunakan grup parameter tersebut untuk klaster baru Anda. Jika menggunakan `reserved-memory-percent`, Anda tidak perlu melakukan ini. Untuk informasi selengkapnya, lihat Mengelola Memori Terpesan.

#### Menurunkan skala klaster Redis dengan replika (Konsol)

Proses berikut menskalakan klaster Redis Anda dengan simpul replika ke jenis simpul yang lebih kecil menggunakan konsol ElastiCache.

Untuk menurunkan skala klaster Redis dengan simpul replika (konsol)

1. Pastikan bahwa tipe simpul yang lebih kecil memadai untuk data dan kebutuhan overhead Anda.
2. Jika grup parameter Anda menggunakan `reserved-memory` untuk menyisihkan memori untuk overhead Redis, sebelum mulai menskalakan pastikan bahwa Anda memiliki grup parameter khusus yang menyimpan jumlah memori yang benar untuk tipe simpul baru Anda.

   Atau, Anda dapat mengubah grup parameter khusus untuk menggunakan `reserved-memory-percent`. Untuk informasi selengkapnya, lihat Mengelola Memori Terpesan.

3. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
4. Pada daftar klaster, pilih klaster yang ingin Anda turunkan skalanya. Klaster ini harus menjalankan mesin Redis dan bukan mesin Redis Berklaster.
5. Pilih Ubah.
6. Di wizard Ubah Klaster:
   a. Pilih tipe simpul yang ingin Anda turunkan skalanya dari daftar Tipe simpul.
   b. Jika Anda menggunakan `reserved-memory` untuk mengelola memori, dari daftar Grup Parameter, pilih grup parameter khusus yang menyimpan jumlah memori yang benar untuk jenis simpul baru.
7. Jika Anda ingin segera menurunkan skala, pilih kotak centang Terapkan segera. Jika kotak centang Terapkan segera tidak dipilih, proses menurunkan skala dilakukan selama jendela pemeliharaan berikutnya dari klaster ini.
8. Pilih Ubah.
9. Ketika status klaster berubah dari mengubah ke tersedia, klaster Anda telah diskalakan ke tipe simpul baru. Tidak perlu memperbarui titik akhir dalam aplikasi Anda.

Menurunkan skala grup replikasi Redis (AWS CLI)

Proses berikut menskalakan grup replikasi Anda dari tipe simpul saat ini ke tipe simpul baru yang lebih kecil menggunakan AWS CLI. Selama proses ini, ElastiCache for Redis memperbarui entri DNS sehingga menunjuk ke simpul baru. Karena hal ini, Anda tidak perlu memperbarui titik akhir dalam aplikasi Anda. Untuk Redis 5.0.5 dan setelahnya, Anda dapat menskalakan klaster dengan failover otomatis diaktifkan saat klaster terus online dan melayani permintaan masuk. Pada versi 4.0.10 dan sebelumnya, Anda mungkin merasakan gangguan singkat baca dan tulis pada versi sebelumnya dari simpul primer saat entri DNS diperbarui.

Namun, baca dari klaster cache replika baca tetap tidak terganggu.

Jumlah waktu yang dibutuhkan untuk menurunkan skala ke tipe simpul yang lebih kecil bervariasi, bergantung pada tipe simpul dan jumlah data dalam klaster cache Anda saat ini.

Menurunkan skala Grup Replikasi Redis (AWS CLI)

1. Tentukan tipe simpul mana yang dapat Anda turunkan skalanya dengan menjalankan perintah AWS CLI `list-allowed-node-type-modifications` menggunakan parameter berikut.

- `--replication-group-id` – nama grup replikasi. Gunakan parameter ini untuk mendeskripsikan grup replikasi tertentu bukannya semua grup replikasi.

Untuk Linux, macOS, atau Unix:

```
aws elasticache list-allowed-node-type-modifications \
     --replication-group-id my-repl-group
```

Untuk Windows:

```
aws elasticache list-allowed-node-type-modifications ^
     --replication-group-id my-repl-group
```

Output dari operasi ini terlihat seperti berikut (format JSON).

```
{
     "ScaleDownModifications": [
         "cache.m3.2xlarge",
         "cache.m3.large",
         "cache.m3.xlarge",
         "cache.m4.10xlarge",
         "cache.m4.2xlarge",
         "cache.m4.4xlarge",
         "cache.m4.large",
         "cache.m4.xlarge",
         "cache.r3.2xlarge",
         "cache.r3.4xlarge",
         "cache.r3.8xlarge",
         "cache.r3.large",
         "cache.r3.xlarge"
     ]
 }
```

Untuk informasi selengkapnya, lihat list-allowed-node-type-modifications di dalam Referensi AWS CLI.

2. Skalakan grup replikasi Anda saat ini ke atas ke tipe simpul baru menggunakan perintah AWS CLI `modify-replication-group` dengan parameter berikut.

   - `--replication-group-id` – nama grup replikasi.

- `--cache-node-type` – jenis simpul baru yang lebih kecil dari klaster cache dalam grup replikasi ini. Nilai ini harus berupa salah satu dari tipe instans yang dihasilkan oleh perintah `list-allowed-node-type-modifications` di langkah 1.
- `--cache-parameter-group-name` – [Opsional] Gunakan parameter ini jika Anda menggunakan `reserved-memory` untuk mengelola memori yang disimpan klaster. Tentukan grup parameter cache khusus yang mencadangkan jumlah memori yang sesuai untuk jenis simpul yang baru. Jika menggunakan `reserved-memory-percent`, Anda dapat menghilangkan parameter ini.
- `--apply-immediately` – Menyebabkan operasi menaikkan skala segera diterapkan. Untuk menunda operasi menaikkan skala ke jendela pemeliharaan berikutnya, gunakan `--no-apply-immediately`.

Untuk Linux, macOS, atau Unix:

```
aws elasticache modify-replication-group \
     --replication-group-id my-repl-group \
     --cache-node-type cache.t2.small  \
     --cache-parameter-group-name redis32-m3-2xl \
     --apply-immediately
```

Untuk Windows:

```
aws elasticache modify-replication-group ^
     --replication-group-id my-repl-group ^
     --cache-node-type cache.t2.small  ^
     --cache-parameter-group-name redis32-m3-2xl \
     --apply-immediately
```

Output dari operasi ini terlihat seperti berikut (format JSON).

```
{"ReplicationGroup": {
         "Status": "available",
         "Description": "Some description",
         "NodeGroups": [
             {
                 "Status": "available",
                 "NodeGroupMembers": [
```

```
                    {
                        "CurrentRole": "primary",
                        "PreferredAvailabilityZone": "us-west-2b",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Port": 6379,
                            "Address": "my-repl-
group-001.8fdx4s.0001.usw2.cache.amazonaws.com"
                        },
                        "CacheClusterId": "my-repl-group-001"
                    },
                    {
                        "CurrentRole": "replica",
                        "PreferredAvailabilityZone": "us-west-2c",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Port": 6379,
                            "Address": "my-repl-
group-002.8fdx4s.0001.usw2.cache.amazonaws.com"
                        },
                        "CacheClusterId": "my-repl-group-002"
                    }
                ],
                "NodeGroupId": "0001",
                "PrimaryEndpoint": {
                    "Port": 6379,
                    "Address": "my-repl-
group.8fdx4s.ng.0001.usw2.cache.amazonaws.com"
                }
            }
        ],
        "ReplicationGroupId": "my-repl-group",
        "SnapshotRetentionLimit": 1,
        "AutomaticFailover": "disabled",
        "SnapshotWindow": "12:00-13:00",
        "SnapshottingClusterId": "my-repl-group-002",
        "MemberClusters": [
            "my-repl-group-001",
            "my-repl-group-002",
        ],
        "PendingModifiedValues": {}
    }
 }
```

Untuk informasi selengkapnya, lihat modify-replication-group di dalam Referensi AWS CLI.

3. Jika Anda menggunakan parameter `--apply-immediately`, pantau status grup replikasi menggunakan perintah AWS CLI `describe-replication-group` dengan parameter berikut. Ketika status berubah dari mengubah ke tersedia, Anda dapat mulai menulis ke grup replikasi baru Anda yang diturunkan skalanya.

   - `--replication-group-id` – nama grup replikasi. Gunakan parameter ini untuk mendeskripsikan grup replikasi tertentu bukannya semua grup replikasi.

   Untuk Linux, macOS, atau Unix:

   ```
   aws elasticache describe-replication-group \
        --replication-group-id my-replication-group
   ```

   Untuk Windows:

   ```
   aws elasticache describe-replication-groups ^
        --replication-group-id my-replication-group
   ```

   Untuk informasi selengkapnya, lihat describe-replication-groups dalam Referensi AWS CLI.

Menurunkan skala grup replikasi Redis (API ElastiCache)

Proses berikut menskalakan grup replikasi Anda dari tipe simpul saat ini ke tipe simpul baru yang lebih kecil menggunakan API ElastiCache. Selama proses ini, ElastiCache for Redis memperbarui entri DNS sehingga menunjuk ke simpul baru. Karena hal ini, Anda tidak perlu memperbarui titik akhir dalam aplikasi Anda. Untuk Redis 5.0.5 dan setelahnya, Anda dapat menskalakan klaster dengan failover otomatis diaktifkan saat klaster terus online dan melayani permintaan masuk. Pada versi 4.0.10 dan sebelumnya, Anda mungkin merasakan gangguan singkat baca dan tulis pada versi sebelumnya dari simpul primer saat entri DNS diperbarui. Namun, baca dari klaster cache replika baca tetap tidak terganggu.

Jumlah waktu yang dibutuhkan untuk menurunkan skala ke tipe simpul yang lebih kecil bervariasi, bergantung pada tipe simpul dan jumlah data dalam klaster cache Anda saat ini.

Untuk menurunkan skala Grup Replikasi Redis (API ElastiCache)

1. Tentukan jenis simpul mana yang dapat Anda turunkan skalanya menggunakan tindakan `ListAllowedNodeTypeModifications` API ElastiCache dengan parameter berikut.

   - `ReplicationGroupId` – nama grup replikasi. Gunakan parameter ini untuk mendeskripsikan grup replikasi tertentu, bukan semua grup replikasi.

   ```
   https://elasticache.us-west-2.amazonaws.com/
       ?Action=ListAllowedNodeTypeModifications
       &ReplicationGroupId=MyReplGroup
       &Version=2015-02-02
       &SignatureVersion=4
       &SignatureMethod=HmacSHA256
       &Timestamp=20150202T192317Z
       &X-Amz-Credential=<credential>
   ```

   Untuk informasi selengkapnya, lihat ListAllowedNodeTypeModifications dalam Referensi API Amazon ElastiCache.

2. Skalakan grup replikasi Anda saat ini ke atas ke tipe simpul baru menggunakan tindakan `ModifyRedplicationGroup` API ElastiCache dan dengan parameter berikut.

   - `ReplicationGroupId` – nama grup replikasi.
   - `CacheNodeType` – jenis simpul baru yang lebih kecil dari klaster cache dalam grup replikasi ini. Nilai ini harus berupa salah satu dari tipe instans yang dihasilkan oleh tindakan `ListAllowedNodeTypeModifications` di langkah 1.
   - `CacheParameterGroupName` – [Opsional] Gunakan parameter ini jika Anda menggunakan `reserved-memory` untuk mengelola memori yang disimpan klaster. Tentukan grup parameter cache khusus yang mencadangkan jumlah memori yang sesuai untuk jenis simpul yang baru. Jika menggunakan `reserved-memory-percent`, Anda dapat menghilangkan parameter ini.
   - `ApplyImmediately` – Atur ke `true` untuk menyebabkan proses menaikkan skala segera diterapkan. Untuk menunda proses menurunkan skala ke jendela pemeliharaan berikutnya, gunakan `ApplyImmediately=false`.

   ```
   https://elasticache.us-west-2.amazonaws.com/
       ?Action=ModifyReplicationGroup
       &ApplyImmediately=true
   ```

```
    &CacheNodeType=cache.m3.2xlarge
    &CacheParameterGroupName=redis32-m3-2xl
    &ReplicationGroupId=myReplGroup
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &Version=2014-12-01
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20141201T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

Untuk informasi selengkapnya, lihat ModifyReplicationGroup dalam Referensi API Amazon ElastiCache.

3. Jika Anda menggunakan `ApplyImmediately=true`, pantau status grup replikasi menggunakan tindakan `DescribeReplicationGroups` API ElastiCache dengan parameter berikut. Ketika status berubah dari mengubah ke tersedia, Anda dapat mulai menulis ke grup replikasi baru Anda yang diturunkan skalanya.

   - `ReplicationGroupId` – nama grup replikasi. Gunakan parameter ini untuk mendeskripsikan grup replikasi tertentu bukannya semua grup replikasi.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeReplicationGroups
    &ReplicationGroupId=MyReplGroup
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &X-Amz-Credential=<credential>
```

Untuk informasi selengkapnya, lihat DescribeReplicationGroups dalam Referensi API Amazon ElastiCache.

### Meningkatkan kapasitas baca

Untuk meningkatkan kapasitas baca, tambahkan replika baca (hingga maksimum lima) ke grup replikasi Redis Anda.

Anda dapat menskalakan kapasitas baca klaster Redis Anda menggunakan konsol ElastiCache, AWS CLI, atau API ElastiCache. Untuk informasi selengkapnya, lihat Menambahkan replika baca, untuk grup replikasi Redis (Mode Klaster Dinonaktifkan).

### Mengurangi kapasitas baca

Untuk mengurangi kapasitas baca, hapus satu atau lebih replika baca dari klaster Redis Anda dengan replika (disebut grup replikasi dalam API/CLI). Jika klaster adalah Multi-AZ dengan failover otomatis diaktifkan, Anda tidak dapat menghapus replika baca terakhir tanpa menonaktifkan Multi-AZ terlebih dahulu. Untuk informasi selengkapnya, lihat Mengubah grup replikasi.

Untuk informasi selengkapnya, lihat Menghapus replika baca, untuk grup replikasi Redis (Mode Klaster Dinonaktifkan) .

## Penskalaan klaster di Redis (Mode Klaster Diaktifkan)

Seiring perubahan permintaan pada klaster, Anda mungkin memutuskan untuk meningkatkan performa atau mengurangi biaya dengan mengubah jumlah serpihan di klaster Redis (mode klaster diaktifkan) Anda. Kami rekomendasikan untuk menggunakan penskalaan horizontal online untuk melakukannya, karena memungkinkan klaster Anda terus melayani permintaan selama proses penskalaan.

Kondisi saat Anda mungkin memutuskan untuk menskalakan ulang klaster Anda meliputi berikut ini:

- Tekanan memori:

  Jika simpul di klaster Anda berada di bawah tekanan memori, Anda mungkin memutuskan untuk menskalakan ke luar agar memiliki lebih banyak sumber daya untuk menyimpan data yang lebih baik dan melayani permintaan.

  Anda dapat menentukan apakah simpul Anda berada di bawah tekanan memori dengan memantau metrik berikut: FreeableMemory, SwapUsage, dan BytesUseForCache.
- Hambatan CPU atau jaringan:

  Jika masalah latensi/throughput mengganggu klaster, Anda mungkin perlu untuk menskalakan ke luar untuk menyelesaikan masalah.

  Anda dapat memantau tingkat latensi dan throughput dengan memantau metrik berikut: CPUUtilization, NetworkBytesIn, NetworkBytesOut, CurrConnections, dan NewConnections.
- Klaster Anda diskalakan berlebih:

  Permintaan saat ini pada klaster Anda adalah sedemikian rupa sehingga penskalaan ke dalam tidak merugikan performa dan mengurangi biaya Anda.

  Anda dapat memantau penggunaan klaster untuk menentukan apakah Anda dapat dengan aman menskalakan ke dalam menggunakan metrik berikut: FreeableMemory, SwapUsage, BytesUseForCache, CPUUtilization, NetworkBytesIn, NetworkBytesOut, CurrConnections, dan NewConnections.

### Dampak Performa dari Penskalaan

Ketika Anda menskalakan menggunakan proses offline, klaster Anda offline untuk sebagian besar proses dan dengan demikian tidak dapat melayani permintaan. Ketika Anda menskalakan

menggunakan metode online, karena penskalaan adalah operasi komputasi intensif, ada beberapa penurunan dalam performa, namun, klaster Anda terus melayani permintaan selama operasi penskalaan. Berapa banyak penurunan yang Anda alami bergantung pada utilisasi CPU normal dan data Anda.

Ada dua cara untuk menskalakan klaster Redis (mode klaster diaktifkan) Anda; penskalaan horizontal dan vertikal.

- Penskalaan horizontal mengizinkan Anda mengubah jumlah grup simpul (serpihan) dalam grup replikasi dengan menambahkan atau menghapus grup simpul (serpihan). Proses resharding online memungkinkan penskalaan ke dalam/ke luar sementara klaster terus melayani permintaan masuk.

  Konfigurasikan slot di klaster baru Anda secara berbeda dari yang berada di klaster lama. Metode offline saja.
- Penskalaan Vertikal - Ubah tipe simpul untuk mengubah ukuran klaster. Penskalaan vertikal secara online memungkinkan untuk menaikkan skala/menurunkan skala sementara klaster terus melayani permintaan masuk.

Jika Anda mengurangi ukuran dan kapasitas memori klaster, baik dengan menskalakan ke dalam atau menurunkan skala, pastikan bahwa konfigurasi baru memiliki memori yang cukup untuk data Anda dan overhead Redis.

Untuk informasi selengkapnya, lihat Pilih ukuran simpul cache.

Daftar Isi

    - Penyeimbangan ulang serpihan secara online (AWS CLI)
    - Penyeimbangan kembali pecahan online (API) ElastiCache
- Penskalaan vertikal online dengan mengubah jenis simpul
  - Menaikkan skala secara online
    - Meningkatkan skala klaster cache Redis (Konsol)
    - Menaikkan skala klaster cache Redis (AWS CLI)
    - Menaikkan skala klaster cache Redis (API ElastiCache)
  - Menurunkan skala secara online
    - Menurunkan skala klaster cache Redis (Konsol)
    - Menurunkan skala klaster cache Redis (AWS CLI)
    - Menurunkan skala klaster cache Redis (API ElastiCache)

Resharding dan penyeimbangan ulang serpihan secara offline untuk Redis (mode klaster diaktifkan)

Keuntungan utama yang Anda dapatkan dari konfigurasi ulang serpihan secara offline adalah bahwa Anda dapat melakukan lebih dari sekadar menambahkan atau menghapus serpihan dari grup replikasi Anda. Ketika Anda melakukan reshard offline, selain mengubah jumlah serpihan dalam grup replikasi, Anda dapat melakukan hal berikut:

> **Note**
>
> Resharding offline tidak didukung pada klaster Redis dengan tingkatan data diaktifkan. Untuk informasi selengkapnya, lihat Tingkatan data.

- Ubah jenis simpul grup replikasi Anda.
- Tentukan Zona Ketersediaan untuk setiap simpul dalam grup replikasi.
- Tingkatkan ke versi mesin yang lebih baru.
- Tentukan jumlah simpul replika di setiap serpihan secara independen.
- Tentukan keyspace untuk setiap serpihan.

Kekurangan utama dari konfigurasi ulang serpihan secara offline adalah bahwa klaster Anda offline dimulai dari bagian pemulihan dari proses berlanjut sampai Anda memperbarui titik akhir dalam aplikasi Anda. Durasi offline klaster Anda akan bervariasi tergantung jumlah data dalam klaster Anda.

Untuk mengonfigurasi ulang klaster Redis (mode klaster diaktifkan) serpihan Anda secara offline

1. Buat cadangan manual klaster Redis yang ada. Untuk informasi selengkapnya, lihat Mengambil cadangan manual.
2. Buat klaster baru dengan memulihkan dari cadangan. Untuk informasi selengkapnya, lihat Melakukan pemulihan dari cadangan ke dalam cache baru.
3. Perbarui titik akhir di aplikasi Anda ke titik akhir klaster baru. Untuk informasi selengkapnya, lihat Menemukan titik akhir koneksi.

Resharding dan penyeimbangan ulang serpihan secara online untuk Redis (mode klaster diaktifkan)

Dengan menggunakan resharding online dan shard rebalancing dengan ElastiCache Amazon untuk Redis versi 3.2.10 atau yang lebih baru, Anda dapat menskalakan Redis (mode cluster diaktifkan) secara ElastiCache dinamis tanpa downtime. Pendekatan ini berarti bahwa klaster Anda dapat terus melayani permintaan bahkan di tengah proses penskalaan atau penyeimbangan beban.

Anda dapat melakukan tindakan berikut:

- Menskalakan ke luar – Meningkatkan kapasitas baca dan tulis dengan menambahkan serpihan (grup simpul) ke klaster Redis (mode klaster diaktifkan) (grup replikasi) Anda.

  Jika Anda menambahkan satu atau beberapa serpihan ke grup replikasi, jumlah simpul di setiap serpihan baru adalah sama dengan jumlah simpul dalam serpihan terkecil yang ada.
- Menskalakan ke dalam – Mengurangi kapasitas baca dan tulis, sekaligus biaya, dengan menghapus serpihan dari klaster Redis (mode klaster diaktifkan) Anda.
- Rebalance - Pindahkan ruang kunci di antara pecahan di cluster Redis (mode cluster diaktifkan) Anda ElastiCache sehingga mereka didistribusikan secara merata di antara pecahan sebanyak mungkin.

Anda tidak dapat melakukan hal berikut:

- Konfigurasikan serpihan secara independen:

  Anda tidak dapat menentukan keyspace untuk serpihan secara independen. Untuk melakukannya, Anda harus menggunakan proses offline.

Saat ini, batasan berikut berlaku ElastiCache untuk resharding dan rebalancing online Redis:

- Proses ini memerlukan Redis mesin versi 3.2.10 atau yang lebih baru. Untuk informasi selengkapnya peningkatan versi mesin Anda, lihat Versi mesin dan pemutakhiran .
- Terdapat pembatasan dengan slot atau keyspace dan item besar:

  Jika salah satu kunci dalam serpihan berisi item besar, kunci tersebut tidak dimigrasikan ke serpihan baru ketika menskalakan ke luar atau penyeimbangan ulang. Fungsi ini dapat mengakibatkan serpihan yang tidak seimbang.

  Jika salah satu kunci dalam serpihan berisi item besar (item yang lebih dari 256 MB setelah serialisasi), serpihan tersebut tidak dihapus saat menskalakan ke dalam. Fungsi ini dapat mengakibatkan beberapa serpihan tidak dihapus.
- Ketika menskalakan ke luar, jumlah simpul dalam serpihan baru sama dengan jumlah simpul dalam serpihan terkecil yang ada.
- Ketika menskalakan ke luar, setiap tag yang umum untuk semua serpihan yang ada akan disalin ke serpihan baru.
- Saat menskalakan cluster Global Data Store, tidak ElastiCache akan secara otomatis mereplikasi Fungsi dari salah satu node yang ada ke node baru. Sebaiknya Anda memuat Fungsi Anda di serpihan baru setelah menskalakan ke luar klaster Anda sehingga setiap serpihan memiliki fungsi yang sama.

> **Note**
>
> ElastiCache Untuk Redis versi 7 ke atas: Saat menskalakan cluster Anda, ElastiCache akan secara otomatis mereplikasi Fungsi yang dimuat di salah satu node yang ada (dipilih secara acak) ke node baru. Jika aplikasi Anda menggunakan Fungsi Redis, kami sarankan memuat semua fungsi Anda ke semua pecahan sebelum diskalakan sehingga cluster Redis Anda ElastiCache tidak berakhir dengan definisi fungsi yang berbeda pada pecahan yang berbeda.

Untuk informasi selengkapnya, lihat Perubahan ukuran klaster online.

Anda dapat menskalakan atau menyeimbangkan ulang cluster Redis (mode cluster enabled) secara horizontal menggunakan AWS Management Console, the AWS CLI, dan API. ElastiCache ElastiCache

### Menambahkan serpihan dengan resharding online

Anda dapat menambahkan pecahan ke cluster Redis (mode klaster diaktifkan) menggunakan AWS Management Console, AWS CLI, atau ElastiCache API. Ketika Anda menambahkan serpihan ke klaster Redis (mode klaster diaktifkan), setiap tag pada serpihan yang ada akan disalin ke serpihan baru.

#### Menambahkan serpihan (Konsol)

Anda dapat menggunakan AWS Management Console untuk menambahkan satu atau beberapa pecahan ke cluster Redis (mode cluster diaktifkan) Anda. Prosedur berikut menjelaskan prosesnya.

Untuk menambahkan serpihan ke klaster Redis (mode klaster diaktifkan) Anda

1. Buka ElastiCache konsol di https://console.aws.amazon.com/elasticache/.
2. Dari panel navigasi, pilih Klaster Redis.
3. Cari dan memilih nama, bukan kotak di sebelah kiri nama klaster, dari klaster Redis (mode klaster diaktifkan) tujuan serpihan akan ditambahkan.

   > **Tip**
   >
   > Redis (mode klaster diaktifkan) menampilkan Redis Berklaster di kolom Mode

4. Pilih Tambahkan serpihan.
   a. Untuk Jumlah serpihan yang akan ditambahkan, pilih jumlah serpihan yang ingin Anda tambahkan ke klaster ini.
   b. Untuk Zona Ketersediaan, pilih salah satu antara Tidak ada preferensi atau Tentukan Zona Ketersediaan.
   c. Jika Anda memilih Tentukan Availability Zone, untuk setiap simpul dalam setiap serpihan, pilih Availability Zone simpul dari daftar Availability Zone.
   d. Pilih Tambahkan.

#### Menambahkan serpihan (AWS CLI)

Proses berikut menjelaskan cara mengonfigurasi ulang serpihan di klaster Redis (mode klaster diaktifkan) Anda dengan menambahkan serpihan menggunakan AWS CLI.

Gunakan parameter berikut dengan `modify-replication-group-shard-configuration`.

Parameter

- `--apply-immediately` – Diperlukan. Menentukan operasi konfigurasi ulang serpihan yang akan segera dimulai.
- `--replication-group-id` – Diperlukan. Menentukan grup replikasi (klaster) mana tempat operasi konfigurasi ulang serpihan dilakukan.
- `--node-group-count` – Diperlukan. Menentukan jumlah serpihan (grup simpul) yang harus ada ketika operasi selesai. Saat menambahkan serpihan, nilai `--node-group-count` harus lebih besar dari jumlah serpihan saat ini.

  Opsional, Anda dapat menentukan Zona Ketersediaan untuk setiap simpul dalam grup replikasi menggunakan `--resharding-configuration`.
- `--resharding-configuration` – Opsional. Daftar Zona Ketersediaan pilihan untuk setiap simpul di setiap serpihan dalam grup replikasi. Gunakan parameter ini hanya jika nilai `--node-group-count` lebih besar dari jumlah serpihan saat ini. Jika parameter ini dihilangkan saat menambahkan pecahan, Amazon ElastiCache memilih Availability Zones untuk node baru.

Contoh berikut mengonfigurasi ulang keyspace lebih dari empat serpihan di klaster Redis `my-cluster` (mode klaster diaktifkan). Contoh ini juga menentukan Zona Ketersediaan untuk setiap simpul di setiap serpihan. Operasi segera dimulai.

Example - Menambahkan Serpihan

Untuk Linux, macOS, atau Unix:

```
aws elasticache modify-replication-group-shard-configuration \
    --replication-group-id my-cluster \
    --node-group-count 4 \
    --resharding-configuration \
        "PreferredAvailabilityZones=us-east-2a,us-east-2c" \
        "PreferredAvailabilityZones=us-east-2b,us-east-2a" \
        "PreferredAvailabilityZones=us-east-2c,us-east-2d" \
        "PreferredAvailabilityZones=us-east-2d,us-east-2c" \
    --apply-immediately
```

Untuk Windows:

```
aws elasticache modify-replication-group-shard-configuration ^
```

```
    --replication-group-id my-cluster ^
    --node-group-count 4 ^
    --resharding-configuration ^
        "PreferredAvailabilityZones=us-east-2a,us-east-2c" ^
        "PreferredAvailabilityZones=us-east-2b,us-east-2a" ^
        "PreferredAvailabilityZones=us-east-2c,us-east-2d" ^
        "PreferredAvailabilityZones=us-east-2d,us-east-2c" ^
    --apply-immediately
```

Untuk informasi selengkapnya, lihat modify-replication-group-shard-konfigurasi dalam AWS CLI dokumentasi.

Menambahkan pecahan (ElastiCache API)

Anda dapat menggunakan ElastiCache API untuk mengkonfigurasi ulang pecahan di klaster Redis (mode cluster diaktifkan) secara online dengan menggunakan operasi. `ModifyReplicationGroupShardConfiguration`

Gunakan parameter berikut dengan `ModifyReplicationGroupShardConfiguration`.

Parameter

- `ApplyImmediately=true` – Diperlukan. Menentukan operasi konfigurasi ulang serpihan yang akan segera dimulai.
- `ReplicationGroupId` – Diperlukan. Menentukan grup replikasi (klaster) mana tempat operasi konfigurasi ulang serpihan dilakukan.
- `NodeGroupCount` – Diperlukan. Menentukan jumlah serpihan (grup simpul) yang harus ada ketika operasi selesai. Saat menambahkan serpihan, nilai `NodeGroupCount` harus lebih besar dari jumlah serpihan saat ini.

  Opsional, Anda dapat menentukan Zona Ketersediaan untuk setiap simpul dalam grup replikasi menggunakan `ReshardingConfiguration`.
- `ReshardingConfiguration` – Opsional. Daftar Zona Ketersediaan pilihan untuk setiap simpul di setiap serpihan dalam grup replikasi. Gunakan parameter ini hanya jika nilai `NodeGroupCount` lebih besar dari jumlah serpihan saat ini. Jika parameter ini dihilangkan saat menambahkan pecahan, Amazon ElastiCache memilih Availability Zones untuk node baru.

Proses berikut menjelaskan cara mengkonfigurasi ulang pecahan di cluster Redis (mode cluster enabled) Anda dengan menambahkan pecahan menggunakan API. ElastiCache

Example - Menambahkan Serpihan

Contoh berikut menambahkan grup simpul ke klaster Redis `my-cluster` (mode klaster diaktifkan), sehingga total ada empat grup simpul saat operasi selesai. Contoh ini juga menentukan Zona Ketersediaan untuk setiap simpul di setiap serpihan. Operasi segera dimulai.

```
https://elasticache.us-east-2.amazonaws.com/
    ?Action=ModifyReplicationGroupShardConfiguration
    &ApplyImmediately=true
    &NodeGroupCount=4
    &ReplicationGroupId=my-cluster

 &ReshardingConfiguration.ReshardingConfiguration.1.PreferredAavailabilityZones.AvailabilityZone
east-2a

 &ReshardingConfiguration.ReshardingConfiguration.1.PreferredAvailabilityZones.AvailabilityZone
east-2c

 &ReshardingConfiguration.ReshardingConfiguration.2.PreferredAvailabilityZones.AvailabilityZone
east-2b

 &ReshardingConfiguration.ReshardingConfiguration.2.PreferredAvailabilityZones.AvailabilityZone
east-2a

 &ReshardingConfiguration.ReshardingConfiguration.3.PreferredAvailabilityZones.AvailabilityZone
east-2c

 &ReshardingConfiguration.ReshardingConfiguration.3.PreferredAvailabilityZones.AvailabilityZone
east-2d

 &ReshardingConfiguration.ReshardingConfiguration.4.PreferredAvailabilityZones.AvailabilityZone
east-2d

 &ReshardingConfiguration.ReshardingConfiguration.4.PreferredAvailabilityZones.AvailabilityZone
east-2c
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20171002T192317Z
    &X-Amz-Credential=<credential>
```

Untuk informasi selengkapnya, lihat ModifyReplicationGroupShardConfigurationdi Referensi ElastiCache API.

## Menghapus serpihan dengan resharding online

Anda dapat menghapus pecahan dari klaster Redis (mode klaster diaktifkan) menggunakan AWS Management Console, AWS CLI, atau ElastiCache API.

Topik

- Menghapus serpihan (Konsol)
- Menghapus serpihan (AWS CLI)
- Menghapus serpihan (API ElastiCache)

### Menghapus serpihan (Konsol)

Proses berikut menjelaskan cara mengonfigurasi ulang serpihan di klaster Redis (mode klaster diaktifkan) dengan menghapus serpihan menggunakan AWS Management Console.

Sebelum menghapus grup node (pecahan) dari grup replikasi Anda, ElastiCache pastikan semua data Anda sesuai dengan pecahan yang tersisa. Jika data muat, serpihan yang ditentukan dihapus dari grup replikasi seperti yang diminta. Jika data tidak muat dalam grup simpul yang tersisa, proses dihentikan dan grup replika akan dibiarkan dengan konfigurasi grup simpul yang sama seperti sebelum permintaan dibuat.

Anda dapat menggunakan AWS Management Console untuk menghapus satu atau beberapa pecahan dari cluster Redis (mode cluster diaktifkan) Anda. Anda tidak dapat menghapus semua serpihan dalam grup replikasi. Sebaliknya, Anda harus menghapus grup replikasi. Untuk informasi selengkapnya, lihat Menghapus grup replikasi. Prosedur berikut menjelaskan proses untuk menghapus satu atau beberapa serpihan.

Untuk menghapus serpihan dari klaster Redis (mode klaster diaktifkan)

1. Buka ElastiCache konsol di https://console.aws.amazon.com/elasticache/.
2. Dari panel navigasi, pilih Klaster Redis.
3. Cari dan pilih nama, bukan kotak di sebelah kiri nama klaster, dari klaster (mode klaster diaktifkan) yang berisi serpihan yang ingin Anda hapus.

> **Tip**
>
> Klaster Redis (mode klaster diaktifkan) memiliki nilai 1 atau lebih besar dalam kolom Serpihan.

4. Dari daftar serpihan, pilih kotak di sebelah kiri nama setiap serpihan yang ingin Anda hapus.
5. Pilih Hapus serpihan.

### Menghapus serpihan (AWS CLI)

Proses berikut menjelaskan cara mengonfigurasi ulang serpihan di klaster Redis (mode klaster diaktifkan) dengan menghapus serpihan menggunakan AWS CLI.

> **Important**
>
> Sebelum menghapus grup node (pecahan) dari grup replikasi Anda, ElastiCache pastikan semua data Anda sesuai dengan pecahan yang tersisa. Jika data muat, serpihan yang ditentukan (`--node-groups-to-remove`) dihapus dari grup replikasi seperti yang diminta dan keyspace-nya dipetakan ke serpihan yang tersisa. Jika data tidak muat dalam grup simpul yang tersisa, proses dihentikan dan grup replika akan dibiarkan dengan konfigurasi grup simpul yang sama seperti sebelum permintaan dibuat.

Anda dapat menggunakan AWS CLI untuk menghapus satu atau beberapa pecahan dari cluster Redis (mode cluster diaktifkan) Anda. Anda tidak dapat menghapus semua serpihan dalam grup replikasi. Sebaliknya, Anda harus menghapus grup replikasi. Untuk informasi selengkapnya, lihat Menghapus grup replikasi.

Gunakan parameter berikut dengan `modify-replication-group-shard-configuration`.

Parameter

- `--apply-immediately` – Diperlukan. Menentukan operasi konfigurasi ulang serpihan yang akan segera dimulai.
- `--replication-group-id` – Diperlukan. Menentukan grup replikasi (klaster) mana tempat operasi konfigurasi ulang serpihan dilakukan.
- `--node-group-count` – Diperlukan. Menentukan jumlah serpihan (grup simpul) yang harus ada ketika operasi selesai. Saat menghapus serpihan, nilai `--node-group-count` harus kurang dari jumlah serpihan saat ini.
- `--node-groups-to-remove` – Diperlukan jika `--node-group-count` kurang dari jumlah grup simpul (serpihan) saat ini. Daftar ID serpihan (grup simpul) yang akan dihapus dari grup replikasi.

Prosedur berikut menjelaskan proses untuk menghapus satu atau beberapa serpihan.

Example - Menghapus Serpihan

Contoh berikut menghapus dua grup simpul dari klaster Redis `my-cluster` (mode klaster diaktifkan), sehingga total ada dua grup simpul saat operasi selesai. Keyspace dari serpihan yang dihapus akan didistribusikan secara merata ke serpihan yang tersisa.

Untuk Linux, macOS, atau Unix:

```
aws elasticache modify-replication-group-shard-configuration \
    --replication-group-id my-cluster \
    --node-group-count 2 \
    --node-groups-to-remove "0002" "0003" \
    --apply-immediately
```

Untuk Windows:

```
aws elasticache modify-replication-group-shard-configuration ^
    --replication-group-id my-cluster ^
    --node-group-count 2 ^
    --node-groups-to-remove "0002" "0003" ^
    --apply-immediately
```

Menghapus serpihan (API ElastiCache)

Anda dapat menggunakan ElastiCache API untuk mengkonfigurasi ulang pecahan di klaster Redis (mode cluster diaktifkan) secara online dengan menggunakan operasi. `ModifyReplicationGroupShardConfiguration`

Proses berikut menjelaskan cara mengkonfigurasi ulang pecahan di cluster Redis (mode cluster enabled) Anda dengan menghapus pecahan menggunakan API. ElastiCache

> **Important**
>
> Sebelum menghapus grup node (pecahan) dari grup replikasi Anda, ElastiCache pastikan semua data Anda sesuai dengan pecahan yang tersisa. Jika data muat, serpihan yang ditentukan (`NodeGroupsToRemove`) dihapus dari grup replikasi seperti yang diminta dan keyspace-nya dipetakan ke serpihan yang tersisa. Jika data tidak muat dalam grup simpul

yang tersisa, proses dihentikan dan grup replika akan dibiarkan dengan konfigurasi grup simpul yang sama seperti sebelum permintaan dibuat.

Anda dapat menggunakan ElastiCache API untuk menghapus satu atau beberapa pecahan dari klaster Redis (mode klaster diaktifkan). Anda tidak dapat menghapus semua serpihan dalam grup replikasi. Sebaliknya, Anda harus menghapus grup replikasi. Untuk informasi selengkapnya, lihat Menghapus grup replikasi.

Gunakan parameter berikut dengan `ModifyReplicationGroupShardConfiguration`.

Parameter

- `ApplyImmediately=true` – Diperlukan. Menentukan operasi konfigurasi ulang serpihan yang akan segera dimulai.
- `ReplicationGroupId` – Diperlukan. Menentukan grup replikasi (klaster) mana tempat operasi konfigurasi ulang serpihan dilakukan.
- `NodeGroupCount` – Diperlukan. Menentukan jumlah serpihan (grup simpul) yang harus ada ketika operasi selesai. Saat menghapus serpihan, nilai `NodeGroupCount` harus kurang dari jumlah serpihan saat ini.
- `NodeGroupsToRemove` – Diperlukan jika `--node-group-count` kurang dari jumlah grup simpul (serpihan) saat ini. Daftar ID serpihan (grup simpul) yang akan dihapus dari grup replikasi.

Prosedur berikut menjelaskan proses untuk menghapus satu atau beberapa serpihan.

Example - Menghapus Serpihan

Contoh berikut menghapus dua grup simpul dari klaster Redis `my-cluster` (mode klaster diaktifkan), sehingga total ada dua grup simpul saat operasi selesai. Keyspace dari serpihan yang dihapus akan didistribusikan secara merata ke serpihan yang tersisa.

```
https://elasticache.us-east-2.amazonaws.com/
    ?Action=ModifyReplicationGroupShardConfiguration
    &ApplyImmediately=true
    &NodeGroupCount=2
    &ReplicationGroupId=my-cluster
    &NodeGroupsToRemove.member.1=0002
    &NodeGroupsToRemove.member.2=0003
```

```
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20171002T192317Z
    &X-Amz-Credential=<credential>
```

Penyeimbangan ulang serpihan secara online

Anda dapat menyeimbangkan kembali pecahan di klaster Redis (mode klaster diaktifkan) menggunakan AWS Management Console, AWS CLI, atau API. ElastiCache

Topik

- Penyeimbangan Ulang Serpihan Secara Online (Konsol)
- Penyeimbangan ulang serpihan secara online (AWS CLI)
- Penyeimbangan kembali pecahan online (API) ElastiCache

Penyeimbangan Ulang Serpihan Secara Online (Konsol)

Proses berikut menjelaskan cara mengonfigurasi ulang serpihan di klaster Redis (mode klaster diaktifkan) dengan menyeimbangkan ulang serpihan menggunakan AWS Management Console.

Untuk menyeimbangkan ulang keyspace di antara serpihan pada klaster Redis (mode klaster diaktifkan)

1. Buka ElastiCache konsol di https://console.aws.amazon.com/elasticache/.
2. Dari panel navigasi, pilih Klaster Redis.
3. Pilih nama, bukan kotak di sebelah kiri nama, dari klaster Redis (mode klaster diaktifkan) yang ingin diseimbangkan ulang.

   > **Tip**
   >
   > Klaster Redis (mode klaster diaktifkan) memiliki nilai 1 atau lebih besar dalam kolom Serpihan.

4. Pilih Seimbangkan ulang.
5. Saat diminta, pilih Seimbangkan ulang. Anda mungkin melihat pesan yang serupa dengan pesan ini: *Slot pada grup replikasi didistribusikan secara merata. Tidak perlu melakukan apa pun. (Layanan:AmazonElastiCache; Kode Status:*

*400; Kode Kesalahan:InvalidReplicationGroupState; ID Permintaan: 2246cebd-9721-11e7-8d5b-e1b0f086c8cf*). Jika Anda melakukannya, pilih Batalkan.

Penyeimbangan ulang serpihan secara online (AWS CLI)

Gunakan parameter berikut dengan `modify-replication-group-shard-configuration`.

Parameter

- `-apply-immediately` – Diperlukan. Menentukan operasi konfigurasi ulang serpihan yang akan segera dimulai.
- `--replication-group-id` – Diperlukan. Menentukan grup replikasi (klaster) mana tempat operasi konfigurasi ulang serpihan dilakukan.
- `--node-group-count` – Diperlukan. Untuk menyeimbangkan ulang keyspace di semua serpihan dalam klaster, nilai ini harus sama dengan jumlah serpihan saat ini.

Proses berikut menjelaskan cara mengonfigurasi ulang serpihan di klaster Redis (mode klaster diaktifkan) dengan menyeimbangkan ulang serpihan menggunakan AWS CLI.

Example - Menyeimbangkan ulang Serpihan dalam Klaster

Contoh berikut menyeimbangkan ulang slot di klaster Redis `my-cluster` (mode klaster diaktifkan) sehingga slot didistribusikan semerata mungkin. Nilai dari `--node-group-count` (4) adalah jumlah serpihan dalam klaster saat ini.

Untuk Linux, macOS, atau Unix:

```
aws elasticache modify-replication-group-shard-configuration \
    --replication-group-id my-cluster \
    --node-group-count 4 \
    --apply-immediately
```

Untuk Windows:

```
aws elasticache modify-replication-group-shard-configuration ^
    --replication-group-id my-cluster ^
    --node-group-count 4 ^
    --apply-immediately
```

Penyeimbangan kembali pecahan online (API) ElastiCache

Anda dapat menggunakan ElastiCache API untuk mengkonfigurasi ulang pecahan di klaster Redis (mode cluster diaktifkan) secara online dengan menggunakan operasi. `ModifyReplicationGroupShardConfiguration`

Gunakan parameter berikut dengan `ModifyReplicationGroupShardConfiguration`.

Parameter

- `ApplyImmediately=true` – Diperlukan. Menentukan operasi konfigurasi ulang serpihan yang akan segera dimulai.
- `ReplicationGroupId` – Diperlukan. Menentukan grup replikasi (klaster) mana tempat operasi konfigurasi ulang serpihan dilakukan.
- `NodeGroupCount` – Diperlukan. Untuk menyeimbangkan ulang keyspace di semua serpihan dalam klaster, nilai ini harus sama dengan jumlah serpihan saat ini.

Proses berikut menjelaskan cara mengkonfigurasi ulang pecahan di cluster Redis (mode cluster enabled) Anda dengan menyeimbangkan kembali pecahan menggunakan API. ElastiCache

Example - Menyeimbangkan Ulang Klaster

Contoh berikut menyeimbangkan ulang slot di klaster Redis `my-cluster` (mode klaster diaktifkan) sehingga slot didistribusikan semerata mungkin. Nilai dari `NodeGroupCount` (4) adalah jumlah serpihan dalam klaster saat ini.

```
https://elasticache.us-east-2.amazonaws.com/
    ?Action=ModifyReplicationGroupShardConfiguration
    &ApplyImmediately=true
    &NodeGroupCount=4
    &ReplicationGroupId=my-cluster
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20171002T192317Z
    &X-Amz-Credential=<credential>
```

Penskalaan vertikal online dengan mengubah jenis simpul

Dengan menggunakan penskalaan vertikal online dengan Amazon ElastiCache for Redis versi 3.2.10 atau yang lebih baru, Anda dapat menskalakan klaster Redis Anda secara dinamis dengan waktu

henti minimal. Hal ini mengizinkan klaster Redis Anda untuk melayani permintaan bahkan saat penskalaan.

> **Note**
>
> Penskalaan tidak didukung antara klaster tingkatan data (misalnya, klaster yang menggunakan jenis simpul r6gd) dan klaster yang tidak menggunakan tingkatan data (misalnya, klaster yang menggunakan jenis simpul r6g). Untuk informasi selengkapnya, lihat Tingkatan data.

Anda dapat melakukan hal berikut:

- Menaikkan skala – Meningkatkan kapasitas baca dan tulis dengan menyesuaikan jenis simpul klaster Redis Anda untuk menggunakan jenis simpul yang lebih besar.

  ElastiCache secara dinamis mengubah ukuran klaster Anda sambil tetap online dan melayani permintaan.
- Menurunkan skala – Mengurangi kapasitas baca dan tulis dengan menyesuaikan jenis simpul ke bawah untuk menggunakan simpul yang lebih kecil. Dan lagi, ElastiCache secara dinamis mengubah ukuran klaster Anda sambil tetap online dan melayani permintaan. Dalam hal ini, Anda mengurangi biaya dengan mengurangi ukuran simpul.

> **Note**
>
> Proses menaikkan dan menurunkan skala bergantung pada pembuatan klaster dengan jenis simpul yang baru dipilih dan menyinkronkan simpul baru dengan yang sebelumnya. Untuk memastikan alur penaikan/penurunan skala yang mulus, lakukan hal berikut:
>
> - Pastikan Anda memiliki kapasitas ENI (Elastic Network Interface) yang cukup. Jika menurunkan skala, pastikan simpul yang lebih kecil memiliki memori yang cukup untuk menyerap lalu lintas yang diharapkan.
>
>   Untuk praktik terbaik tentang manajemen memori, lihat Mengelola Memori Terpesan.
> - Sementara proses penskalaan vertikal dirancang untuk tetap sepenuhnya online, prosesnya bergantung pada sinkronisasi data antara simpul lama dan simpul baru. Sebaiknya Anda memulai penaikan/penurunan skala pada saat lalu lintas data Anda sedang minimum.

- Uji perilaku aplikasi Anda selama penskalaan ke dalam di lingkungan penahapan, jika memungkinkan.

Daftar Isi



## Menaikkan skala secara online

Topik



### Meningkatkan skala klaster cache Redis (Konsol)

Prosedur berikut menjelaskan cara menaikkan skala klaster Redis menggunakan Konsol Manajemen ElastiCache. Selama proses ini, klaster Redis Anda akan terus melayani permintaan dengan waktu henti minimal.

Untuk menaikkan skala klaster Redis (konsol)

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Dari panel navigasi, pilih Klaster Redis.
3. Dari daftar klaster, pilih klaster.
4. Pilih Ubah.

5. Di wizard Ubah Klaster:

   - Pilih tipe simpul sebagai tujuan penskalaan dari daftar Jenis simpul. Untuk menaikkan skala, pilih tipe simpul yang lebih besar dari simpul saat ini.

6. Jika Anda ingin segera menaikkan skala, pilih kotak Terapkan segera. Jika kotak Terapkan segera tidak dipilih, proses menaikkan skala dilakukan selama jendela pemeliharaan berikutnya dari klaster ini.

7. Pilih Ubah.

   Jika Anda memilih Terapkan segera pada langkah sebelumnya, status klaster berubah ke sedang mengubah. Ketika status berubah ke tersedia, pengubahan selesai dan Anda dapat mulai menggunakan klaster baru tersebut.

### Menaikkan skala klaster cache Redis (AWS CLI)

Prosedur berikut menjelaskan cara menaikkan skala klaster cache Redis menggunakan AWS CLI. Selama proses ini, klaster Redis Anda akan terus melayani permintaan dengan waktu henti minimal.

Untuk menaikkan skala klaster cache Redis (AWS CLI)

1. Tentukan tipe simpul mana yang dapat Anda menaikkan skalanya dengan menjalankan perintah AWS CLI `list-allowed-node-type-modifications` menggunakan parameter berikut.

   Untuk Linux, macOS, atau Unix:

   ```
   aws elasticache list-allowed-node-type-modifications \
        --replication-group-id my-replication-group-id
   ```

   Untuk Windows:

   ```
   aws elasticache list-allowed-node-type-modifications ^
        --replication-group-id my-replication-group-id
   ```

   Output dari operasi di atas terlihat seperti berikut (format JSON).

   ```
   {
        "ScaleUpModifications": [
            "cache.m3.2xlarge",
            "cache.m3.large",
   ```

```
          "cache.m3.xlarge",
          "cache.m4.10xlarge",
          "cache.m4.2xlarge",
          "cache.m4.4xlarge",
          "cache.m4.large",
          "cache.m4.xlarge",
          "cache.r3.2xlarge",
          "cache.r3.4xlarge",
          "cache.r3.8xlarge",
          "cache.r3.large",
          "cache.r3.xlarge"
      ]
         "ScaleDownModifications": [
          "cache.t2.micro",
          "cache.t2.small ",
          "cache.t2.medium",
          "cache.t1.small "
      ],
}
```

Untuk informasi selengkapnya, lihat list-allowed-node-type-modifications dalam Referensi AWS CLI.

2. Ubah grup replikasi Anda untuk menaikkan skala ke tipe simpul baru yang lebih besar menggunakan perintah AWS CLI `modify-replication-group` dan parameter berikut.

   - `--replication-group-id` – Nama grup replikasi untuk hasil penaikan skala yang dipilih.
   - `--cache-node-type` – Jenis simpul baru yang akan Anda skalakan ke klaster cache. Nilai ini harus berupa salah satu jenis simpul yang dikembalikan oleh perintah `list-allowed-node-type-modifications` di langkah 1.
   - `--cache-parameter-group-name` – [Opsional] Gunakan parameter ini jika Anda menggunakan `reserved-memory` untuk mengelola memori terpesan klaster. Tentukan grup parameter cache khusus yang mencadangkan jumlah memori yang sesuai untuk jenis simpul yang baru. Jika menggunakan `reserved-memory-percent`, Anda dapat menghilangkan parameter ini.
   - `--apply-immediately` – Membuat proses penaikan skala segera diterapkan. Untuk menunda proses menaikkan skala ke masa pemeliharaan berikutnya dari klaster, gunakan parameter `--no-apply-immediately`.

   Untuk Linux, macOS, atau Unix:

```
aws elasticache modify-replication-group  \
     --replication-group-id my-redis-cluster \
     --cache-node-type cache.m3.xlarge \
     --apply-immediately
```

Untuk Windows:

```
aws elasticache modify-replication-group ^
     --replication-group-id my-redis-cluster ^
     --cache-node-type cache.m3.xlarge ^
     --apply-immediately
```

Output dari operasi di atas terlihat seperti berikut (format JSON).

```
{
  "ReplicationGroup": {
        "Status": "modifying",
        "Description": "my-redis-cluster",
        "NodeGroups": [
            {
                "Status": "modifying",
                "Slots": "0-16383",
                "NodeGroupId": "0001",
                "NodeGroupMembers": [
                    {
                        "PreferredAvailabilityZone": "us-east-1f",
                        "CacheNodeId": "0001",
                        "CacheClusterId": "my-redis-cluster-0001-001"
                    },
                    {
                        "PreferredAvailabilityZone": "us-east-1d",
                        "CacheNodeId": "0001",
                        "CacheClusterId": "my-redis-cluster-0001-002"
                    }
                ]
            }
        ],
        "ConfigurationEndpoint": {
            "Port": 6379,
```

```
            "Address": "my-redis-
cluster.r7gdfi.clustercfg.use1.cache.amazonaws.com"
        },
        "ClusterEnabled": true,
        "ReplicationGroupId": "my-redis-cluster",
        "SnapshotRetentionLimit": 1,
        "AutomaticFailover": "enabled",
        "SnapshotWindow": "07:30-08:30",
        "MemberClusters": [
            "my-redis-cluster-0001-001",
            "my-redis-cluster-0001-002"
        ],
        "CacheNodeType": "cache.m3.xlarge",
         "DataTiering": "disabled"
        "PendingModifiedValues": {}
    }
}
```

Untuk informasi selengkapnya, lihat modify-replication-group dalam Referensi AWS CLI.

3. Jika Anda menggunakan `--apply-immediately`, periksa status klaster cache menggunakan perintah `describe-cache-clusters` AWS CLI dengan parameter berikut. Ketika status berubah ke tersedia, Anda dapat mulai menggunakan simpul klaster cache baru yang lebih besar.

### Menaikkan skala klaster cache Redis (API ElastiCache)

Proses berikut menjelaskan cara menskalakan klaster cache dari tipe simpul saat ini ke tipe simpul baru yang lebih besar menggunakan API ElastiCache. Selama proses ini, ElastiCache for Redis memperbarui entri DNS sehingga mengarah ke simpul baru. Karena hal ini, Anda tidak perlu memperbarui titik akhir dalam aplikasi Anda. Untuk Redis versi 5.0.5 ke atas, Anda dapat menskalakan klaster yang mengaktifkan failover otomatis selagi klaster tetap online dan melayani permintaan masuk. Pada versi 4.0.10 ke bawah, Anda mungkin merasakan gangguan singkat terhadap operasi baca dan tulis pada versi sebelumnya dari simpul primer saat entri DNS diperbarui.

Jumlah waktu yang dibutuhkan untuk menaikkan skala ke tipe simpul yang lebih besar bervariasi, bergantung pada tipe simpul dan jumlah data dalam klaster cache Anda saat ini.

Untuk menaikkan skala klaster cache Redis (API ElastiCache)

1. Tentukan tipe simpul yang dapat dipilih untuk penaikan skala menggunakan tindakan `ListAllowedNodeTypeModifications` API ElastiCache dengan parameter berikut.

   - `ReplicationGroupId` – nama grup replikasi. Gunakan parameter ini untuk mendeskripsikan grup replikasi tertentu, bukan semua grup replikasi.

   ```
   https://elasticache.us-west-2.amazonaws.com/
       ?Action=ListAllowedNodeTypeModifications
       &ReplicationGroupId=MyReplGroup
       &Version=2015-02-02
       &SignatureVersion=4
       &SignatureMethod=HmacSHA256
       &Timestamp=20150202T192317Z
       &X-Amz-Credential=<credential>
   ```

   Untuk informasi selengkapnya, lihat ListAllowedNodeTypeModifications di Referensi API Amazon ElastiCache.

2. Skalakan grup replikasi Anda saat ini ke tipe simpul baru menggunakan tindakan API ElastiCache `ModifyReplicationGroup` dan dengan parameter berikut.

   - `ReplicationGroupId` – nama grup replikasi.
   - `CacheNodeType` – tipe simpul baru yang lebih besar dari klaster cache dalam grup replikasi ini. Nilai ini harus berupa salah satu tipe instans yang ditampilkan oleh tindakan `ListAllowedNodeTypeModifications` di langkah 1.
   - `CacheParameterGroupName` – [Opsional] Gunakan parameter ini jika Anda menggunakan `reserved-memory` untuk mengelola memori terpesan klaster. Tentukan grup parameter cache khusus yang mencadangkan jumlah memori yang sesuai untuk jenis simpul yang baru. Jika menggunakan `reserved-memory-percent`, Anda dapat menghilangkan parameter ini.
   - `ApplyImmediately` – Tetapkan ke `true` agar proses penaikkan skala segera diterapkan. Untuk menunda proses penaikkan skala ke masa pemeliharaan berikutnya, gunakan `ApplyImmediately=false`.

   ```
   https://elasticache.us-west-2.amazonaws.com/
       ?Action=ModifyReplicationGroup
       &ApplyImmediately=true
   ```

```
    &CacheNodeType=cache.m3.2xlarge
    &CacheParameterGroupName=redis32-m3-2xl
    &ReplicationGroupId=myReplGroup
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &Version=2014-12-01
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20141201T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

Untuk informasi selengkapnya, lihat ModifyReplicationGroup di Referensi API Amazon ElastiCache.

3. Jika Anda menggunakan `ApplyImmediately=true`, pantau status grup replikasi menggunakan tindakan `DescribeReplicationGroups` API ElastiCache dengan parameter berikut. Ketika status berubah dari mengubah ke tersedia, Anda dapat mulai menulis ke grup replikasi baru yang telah dinaikkan skalanya.

   - `ReplicationGroupId` – nama grup replikasi. Gunakan parameter ini untuk mendeskripsikan grup replikasi tertentu, bukan semua grup replikasi.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeReplicationGroups
    &ReplicationGroupId=MyReplGroup
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &X-Amz-Credential=<credential>
```

Untuk informasi selengkapnya, lihat DescribeReplicationGroups di Referensi API Amazon ElastiCache.

## Menurunkan skala secara online

Topik

- Menurunkan skala klaster cache Redis (Konsol)
- Menurunkan skala klaster cache Redis (AWS CLI)
- Menurunkan skala klaster cache Redis (API ElastiCache)

### Menurunkan skala klaster cache Redis (Konsol)

Prosedur berikut menjelaskan cara menurunkan skala klaster Redis menggunakan Konsol Manajemen ElastiCache. Selama proses ini, klaster Redis Anda akan terus melayani permintaan dengan waktu henti minimal.

Untuk menurunkan skala klaster Redis (konsol)

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Dari panel navigasi, pilih Klaster Redis.
3. Dari daftar klaster, pilih klaster pilihan Anda.
4. Pilih Ubah.
5. Di wizard Ubah Klaster:
    - Pilih tipe simpul sebagai tujuan penskalaan dari daftar Jenis simpul. Untuk menurunkan skala, pilih jenis simpul yang lebih kecil dari simpul Anda saat ini. Perhatikan bahwa tidak semua jenis simpul tersedia sebagai pilihan penurunan skala.
6. Jika Anda ingin segera menurunkan skala, pilih kotak Terapkan segera. Jika kotak Terapkan segera tidak dipilih, proses penurunan skala dilakukan selama jendela pemeliharaan berikutnya dari klaster ini.
7. Pilih Ubah.

    Jika Anda memilih Terapkan segera pada langkah sebelumnya, status klaster berubah ke sedang mengubah. Ketika status berubah ke tersedia, pengubahan selesai dan Anda dapat mulai menggunakan klaster baru tersebut.

### Menurunkan skala klaster cache Redis (AWS CLI)

Prosedur berikut menjelaskan cara menurunkan skala klaster cache Redis menggunakan AWS CLI. Selama proses ini, klaster Redis Anda akan terus melayani permintaan dengan waktu henti minimal.

Untuk menurunkan skala klaster cache Redis (AWS CLI)

1. Tentukan jenis simpul mana yang dapat Anda turunkan skalanya dengan menjalankan perintah `list-allowed-node-type-modifications` AWS CLI menggunakan parameter berikut.

   Untuk Linux, macOS, atau Unix:

   ```
   aws elasticache list-allowed-node-type-modifications \
        --replication-group-id my-replication-group-id
   ```

   Untuk Windows:

   ```
   aws elasticache list-allowed-node-type-modifications ^
        --replication-group-id my-replication-group-id
   ```

   Output dari operasi di atas terlihat seperti berikut (format JSON).

   ```
   {
        "ScaleUpModifications": [
            "cache.m3.2xlarge",
            "cache.m3.large",
            "cache.m3.xlarge",
            "cache.m4.10xlarge",
            "cache.m4.2xlarge",
            "cache.m4.4xlarge",
            "cache.m4.large",
            "cache.m4.xlarge",
            "cache.r3.2xlarge",
            "cache.r3.4xlarge",
            "cache.r3.8xlarge",
            "cache.r3.large",
            "cache.r3.xlarge"
        ]

           "ScaleDownModifications": [
            "cache.t2.micro",
            "cache.t2.small ",
            "cache.t2.medium ",
            "cache.t1.small"
        ]
   }
   ```

Untuk informasi selengkapnya, lihat list-allowed-node-type-modifications dalam Referensi AWS CLI.

2. Ubah grup replikasi Anda untuk menurunkan skala ke jenis simpul baru yang lebih kecil menggunakan perintah `modify-replication-group` AWS CLI dan parameter berikut.

   - `--replication-group-id` – Nama grup replikasi tujuan penurunan skala.
   - `--cache-node-type` – Jenis simpul baru yang akan Anda skalakan ke klaster cache. Nilai ini harus berupa salah satu jenis simpul yang dikembalikan oleh perintah `list-allowed-node-type-modifications` di langkah 1.
   - `--cache-parameter-group-name` – [Opsional] Gunakan parameter ini jika Anda menggunakan `reserved-memory` untuk mengelola memori terpesan klaster. Tentukan grup parameter cache khusus yang mencadangkan jumlah memori yang sesuai untuk jenis simpul yang baru. Jika menggunakan `reserved-memory-percent`, Anda dapat menghilangkan parameter ini.
   - `--apply-immediately` – Membuat proses penaikan skala segera diterapkan. Untuk menunda proses penurunan skala ke jendela pemeliharaan berikutnya klaster, gunakan parameter `--no-apply-immediately`.

   Untuk Linux, macOS, atau Unix:

   ```
   aws elasticache modify-replication-group  \
        --replication-group-id my-redis-cluster \
        --cache-node-type cache.t2.micro \
        --apply-immediately
   ```

   Untuk Windows:

   ```
   aws elasticache modify-replication-group ^
        --replication-group-id my-redis-cluster ^
        --cache-node-type cache.t2.micro ^
        --apply-immediately
   ```

   Output dari operasi di atas terlihat seperti berikut (format JSON).

   ```
   {
   ```

```
  "ReplicationGroup": {
        "Status": "modifying",
        "Description": "my-redis-cluster",
        "NodeGroups": [
            {
                "Status": "modifying",
                "Slots": "0-16383",
                "NodeGroupId": "0001",
                "NodeGroupMembers": [
                    {
                        "PreferredAvailabilityZone": "us-east-1f",
                        "CacheNodeId": "0001",
                        "CacheClusterId": "my-redis-cluster-0001-001"
                    },
                    {
                        "PreferredAvailabilityZone": "us-east-1d",
                        "CacheNodeId": "0001",
                        "CacheClusterId": "my-redis-cluster-0001-002"
                    }
                ]
            }
        ],
        "ConfigurationEndpoint": {
            "Port": 6379,
            "Address": "my-redis-
cluster.r7gdfi.clustercfg.use1.cache.amazonaws.com"
        },
        "ClusterEnabled": true,
        "ReplicationGroupId": "my-redis-cluster",
        "SnapshotRetentionLimit": 1,
        "AutomaticFailover": "enabled",
        "SnapshotWindow": "07:30-08:30",
        "MemberClusters": [
            "my-redis-cluster-0001-001",
            "my-redis-cluster-0001-002"
        ],
        "CacheNodeType": "cache.t2.micro",
         "DataTiering": "disabled"
        "PendingModifiedValues": {}
    }
}
```

Untuk informasi selengkapnya, lihat modify-replication-group dalam Referensi AWS CLI.

3. Jika Anda menggunakan `--apply-immediately`, periksa status klaster cache menggunakan perintah `describe-cache-clusters` AWS CLI dengan parameter berikut. Ketika status berubah ke tersedia, Anda dapat mulai menggunakan simpul klaster cache baru yang lebih kecil.

Menurunkan skala klaster cache Redis (API ElastiCache)

Proses berikut menskalakan grup replikasi Anda dari jenis simpul saat ini ke simpul baru yang lebih kecil menggunakan API ElastiCache. Selama proses ini, klaster Redis Anda akan terus melayani permintaan dengan waktu henti minimal.

Jumlah waktu yang dibutuhkan untuk menurunkan skala ke jenis simpul yang lebih kecil bervariasi, bergantung pada jenis simpul dan jumlah data dalam klaster cache Anda saat ini.

Menurunkan skala (API ElastiCache)

1. Tentukan jenis simpul baru untuk penurunan skala menggunakan tindakan `ListAllowedNodeTypeModifications` API ElastiCache dengan parameter berikut.

   - `ReplicationGroupId` – nama grup replikasi. Gunakan parameter ini untuk mendeskripsikan grup replikasi tertentu, bukan semua grup replikasi.

   ```
   https://elasticache.us-west-2.amazonaws.com/
       ?Action=ListAllowedNodeTypeModifications
       &ReplicationGroupId=MyReplGroup
       &Version=2015-02-02
       &SignatureVersion=4
       &SignatureMethod=HmacSHA256
       &Timestamp=20150202T192317Z
       &X-Amz-Credential=<credential>
   ```

   Untuk informasi selengkapnya, lihat ListAllowedNodeTypeModifications di Referensi API Amazon ElastiCache.

2. Turunkan skala grup replikasi Anda saat ini ke jenis simpul baru menggunakan tindakan `ModifyReplicationGroup` API ElastiCache dan dengan parameter berikut.

   - `ReplicationGroupId` – nama grup replikasi.

- `CacheNodeType` – jenis simpul baru yang lebih kecil dari klaster cache dalam grup replikasi ini. Nilai ini harus berupa salah satu tipe instans yang ditampilkan oleh tindakan `ListAllowedNodeTypeModifications` di langkah 1.
- `CacheParameterGroupName` – [Opsional] Gunakan parameter ini jika Anda menggunakan `reserved-memory` untuk mengelola memori terpesan klaster. Tentukan grup parameter cache khusus yang mencadangkan jumlah memori yang sesuai untuk jenis simpul yang baru. Jika menggunakan `reserved-memory-percent`, Anda dapat menghilangkan parameter ini.
- `ApplyImmediately` – Diatur ke `true` agar membuat proses penurunan skala diterapkan segera. Untuk menunda proses penurunan skala ke masa pemeliharaan berikutnya, gunakan `ApplyImmediately=false`.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=ModifyReplicationGroup
    &ApplyImmediately=true
    &CacheNodeType=cache.t2.micro
    &CacheParameterGroupName=redis32-m3-2xl
    &ReplicationGroupId=myReplGroup
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &Version=2014-12-01
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20141201T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

Untuk informasi selengkapnya, lihat ModifyReplicationGroup di Referensi API Amazon ElastiCache.

# Mulai menggunakan JSON di ElastiCache for Redis

ElastiCache for Redis mendukung format JavaScript Object Notation (JSON) asli, yang merupakan cara sederhana dan tanpa skema untuk mengenkode set data yang kompleks di dalam klaster Redis. Anda dapat menyimpan dan mengakses data secara native menggunakan format JavaScript Object

Notation (JSON) di dalam klaster Redis, dan memperbarui data JSON yang disimpan dalam klaster tersebut—tanpa perlu mengelola kode kustom untuk melakukan serialisasi dan deserialisasinya.

Selain menggunakan operasi API Redis untuk aplikasi yang beroperasi melalui JSON, Anda sekarang dapat dengan efisien mengambil dan memperbarui bagian tertentu dari dokumen JSON tanpa perlu memanipulasi seluruh objek. Hal ini dapat meningkatkan performa dan mengurangi biaya. Anda juga dapat menelusuri konten dokumen JSON Anda menggunakan kueri `JSONPath` bergaya Goessner.

Setelah Anda membuat klaster dengan versi mesin yang didukung, jenis data JSON dan perintah terkait akan tersedia secara otomatis. Hal ini memiliki kompatibilitas API dan RDB dengan modul RedisJSON versi 2, sehingga Anda dapat dengan mudah memigrasikan aplikasi Redis berbasis JSON yang ada ke ElastiCache for Redis. Untuk informasi selengkapnya tentang perintah Redis yang didukung, lihat Perintah JSON Redis yang didukung.

Metrik `JsonBasedCmds` dan `JsonBasedCmdsLatency` yang terkait JSON dimasukkan ke dalam CloudWatch untuk memantau penggunaan jenis data ini. Untuk informasi selengkapnya, lihat Metrik untuk Redis.

> **Note**
>
> Untuk menggunakan JSON, Anda harus menjalankan Redis versi 6.2.6 atau lebih baru.

Topik

- Gambaran umum jenis data Redis JSON
- Perintah JSON Redis yang didukung

## Gambaran umum jenis data Redis JSON

ElastiCache for Redis mendukung sejumlah perintah Redis untuk menangani jenis data JSON. Berikut ini adalah gambaran umum jenis data JSON dan daftar mendetail yang berisi perintah Redis yang didukung.

## Terminologi

| Istilah | Deskripsi |
| --- | --- |
| Dokumen JSON | Mengacu pada nilai kunci Redis JSON. |
| Nilai JSON | Mengacu pada subset dari dokumen JSON, termasuk root yang merepresentasikan seluruh dokumen. Nilai bisa berupa kontainer atau entri dalam kontainer. |
| Elemen JSON | Setara dengan nilai JSON. |

### Standar JSON yang didukung

Format JSON sesuai dengan standar pertukaran data JSON RFC 7159 dan ECMA-404. Mendukung UTF-8 Unicode pada teks JSON.

### Elemen root

Elemen root dapat berupa jenis data JSON apa pun. Perhatikan bahwa di RFC 4627 sebelumnya, hanya objek atau array yang diizinkan sebagai nilai root. Sejak pembaruan ke RFC 7159, root dokumen JSON dapat berupa jenis data JSON apa pun.

### Batas ukuran dokumen

Dokumen JSON disimpan secara internal dalam format yang dioptimalkan untuk modifikasi dan akses cepat. Format ini biasanya menimbulkan konsumsi cukup banyak memori daripada representasi terserialisasi yang setara dari dokumen yang sama.

Konsumsi memori oleh satu dokumen JSON dibatasi hingga 64 MB, yang merupakan ukuran struktur data dalam memori, bukan string JSON. Anda dapat memeriksa jumlah memori yang dikonsumsi oleh dokumen JSON dengan menggunakan perintah `JSON.DEBUG MEMORY`.

### ACL JSON

- Serupa dengan kategori per jenis data yang ada (@string, @hash, dll.), kategori baru @json ditambahkan untuk menyederhanakan pengelolaan akses ke perintah dan data JSON. Tidak

ada perintah Redis lain yang merupakan anggota kategori @json. Semua perintah JSON memberlakukan pembatasan dan izin ruang kunci atau perintah apa pun.

- Ada lima kategori Redis ACL yang ada yang diperbarui untuk menyertakan perintah JSON baru: @read, @write, @fast, @slow dan @admin. Tabel berikut menunjukkan pemetaan perintah JSON ke kategori yang sesuai.

ACL

| Perintah JSON | @read | @write | @fast | @slow | @admin |
|---|---|---|---|---|---|
| JSON.ARRAPPEND | | y | y | | |
| JSON.ARRINDEX | y | | y | | |
| JSON.ARRINSERT | | y | y | | |
| JSON.ARRLEN | y | | y | | |
| JSON.ARRPOP | | y | y | | |
| JSON.ARRTRIM | | y | y | | |
| JSON.CLEAR | | y | y | | |
| JSON.DEBUG | y | | | y | y |
| JSON.DEL | | y | y | | |
| JSON.FORGET | | y | y | | |

| Perintah JSON | @read | @write | @fast | @slow | @admin |
|---|---|---|---|---|---|
| JSON.GET | y | | y | | |
| JSON.MGET | y | | y | | |
| JSON.NUMINCRBY | | y | y | | |
| JSON.NUMMULTBY | | y | y | | |
| JSON.OBJKEYS | y | | y | | |
| JSON.OBJLEN | y | | y | | |
| JSON.RESP | y | | y | | |
| JSON.SET | | y | | y | |
| JSON.STRAPPEND | | y | y | | |
| JSON.STRLEN | y | | y | | |
| JSON.STRLEN | y | | y | | |
| JSON.TOGGLE | | y | y | | |
| JSON.TYPE | y | | y | | |
| JSON.NUMINCRBY | | y | y | | |

## Batas kedalaman bersarang

Ketika objek atau array JSON memiliki elemen yang merupakan objek atau array JSON lain, objek dalam atau array dianggap "bersarang" di dalam objek luar atau array. Batas kedalaman bersarang maksimum adalah 128. Setiap percobaan untuk membuat dokumen yang berisi kedalaman bersarang lebih dari 128 akan ditolak dengan kesalahan.

## Sintaks perintah

Kebanyakan perintah memerlukan nama kunci Redis sebagai argumen pertama. Beberapa perintah juga memiliki argumen jalur. Argumen jalur ditetapkan secara default ke root jika bersifat opsional dan tidak disediakan.

Notasi:

- Argumen wajib diapit oleh tanda kurung sudut. Misalnya: <key>
- Argumen opsional diapit oleh tanda kurung siku. Misalnya: [path]
- Argumen opsional tambahan ditunjukkan oleh elipsis ("..."). Misalnya: [json...]

## Sintaks jalur

Redis JSON mendukung dua jenis sintaksis jalur:

- Sintaks yang ditingkatkan – Mengikuti sintaksis JSONPath yang dijelaskan oleh Goessner, seperti yang ditunjukkan pada tabel berikut. Kami telah menyusun ulang dan memodifikasi deskripsi dalam tabel agar jelas.
- Sintaks terbatas – Memiliki kemampuan kueri terbatas.

> **Note**
>
> Hasil dari beberapa perintah bersifat sensitif terhadap jenis sintaksis jalur yang digunakan.

Jika jalur kueri diawali dengan '$', kueri tersebut menunjukkan penggunaan sintaksis yang ditingkatkan. Jika tidak, maka kueri tersebut menggunakan sintaksis terbatas.

Sintaks yang ditingkatkan

| Simbol/Ekspresi | Deskripsi |
| --- | --- |
| $ | Elemen root. |
| . atau [] | Operator turunan. |
| .. | Penurunan rekursif. |
| * | Wildcard. Semua elemen dalam sebuah objek atau array. |
| [] | Operator subskrip array. Indeks berbasis 0. |
| [,] | Operator Union. |
| [start:end:step] | Operator irisan array. |
| ?() | Menerapkan ekspresi filter (skrip) ke array atau objek saat ini. |
| () | Ekspresi filter. |
| @ | Digunakan dalam ekspresi filter yang merujuk ke simpul saat ini yang sedang diproses. |
| == | Sama dengan, digunakan dalam ekspresi filter. |
| != | Tidak sama dengan, digunakan dalam ekspresi filter. |
| > | Lebih dari, digunakan dalam ekspresi filter. |
| >= | Lebih dari atau sama dengan, digunakan dalam ekspresi filter. |
| < | Kurang dari, digunakan dalam ekspresi filter. |
| <= | Kurang dari atau sama dengan, digunakan dalam ekspresi filter. |

| Simbol/Ekspresi | Deskripsi |
| --- | --- |
| && | Logika AND, digunakan untuk menggabungkan beberapa ekspresi filter. |
| \|\| | Logika OR, digunakan untuk menggabungkan beberapa ekspresi filter. |

Contoh

Contoh-contoh berikut dibuat berdasarkan contoh data XML Goessner, yang telah kami modifikasi dengan menambahkan bidang tambahan.

```
{ "store": {
    "book": [
      { "category": "reference",
        "author": "Nigel Rees",
        "title": "Sayings of the Century",
        "price": 8.95,
        "in-stock": true,
        "sold": true
      },
      { "category": "fiction",
        "author": "Evelyn Waugh",
        "title": "Sword of Honour",
        "price": 12.99,
        "in-stock": false,
        "sold": true
      },
      { "category": "fiction",
        "author": "Herman Melville",
        "title": "Moby Dick",
        "isbn": "0-553-21311-3",
        "price": 8.99,
        "in-stock": true,
        "sold": false
      },
      { "category": "fiction",
        "author": "J. R. R. Tolkien",
        "title": "The Lord of the Rings",
        "isbn": "0-395-19395-8",
```

```
        "price": 22.99,
        "in-stock": false,
        "sold": false
      }
    ],
    "bicycle": {
      "color": "red",
      "price": 19.95,
      "in-stock": true,
      "sold": false
    }
  }
}
```

| Jalur | Deskripsi |
| --- | --- |
| $.store.book[*].author | Penulis semua buku di toko. |
| $..author | Semua penulis. |
| $.store.* | Semua anggota toko. |
| $["store"].* | Semua anggota toko. |
| $.store..price | Harga semua yang ada di toko. |
| $..* | Semua anggota rekursif dari struktur JSON. |
| $..book[*] | Semua buku. |
| $..book[0] | Buku pertama. |
| $..book[-1] | Buku terakhir. |
| $..book[0:2] | Dua buku pertama. |
| $..book[0,1] | Dua buku pertama. |
| $..book[0:4] | Buku dari indeks 0 hingga 3 (indeks akhir tidak inklusif). |
| $..book[0:4:2] | Buku pada indeks 0, 2. |

| Jalur | Deskripsi |
| --- | --- |
| $..book[?(@.isbn)] | Semua buku dengan nomor ISBN. |
| $..book[?(@.price<10)] | Semua buku yang lebih murah dari $10. |
| ‘$..book[?(@.price < 10)]’ | Semua buku yang lebih murah dari $10. (Jalur harus diberi tanda kutip jika berisi spasi.) |
| ‘$..book[?(@.["price"] < 10)]’ | Semua buku yang lebih murah dari $10. |
| '$..book[?(@.["price"] < 10)]' | Semua buku yang lebih murah dari $10. |
| $..book[?(@.price>=10&&@.price<=100)] | Semua buku dalam kisaran harga $10 hingga $100, inklusif. |
| '$..book[?(@.price>=10 && @.price<=100)]' | Semua buku dalam kisaran harga $10 hingga $100, inklusif. (Jalur harus diberi tanda kutip jika berisi spasi.) |
| $..book[?(@.sold==true\|\|@.in-stock==false)] | Semua buku yang terjual atau habis. |
| '$..book[?(@.sold == true \|\| @.in-stock == false)]' | Semua buku yang terjual atau habis. (Jalur harus diberi tanda kutip jika berisi spasi.) |
| '$.store.book[?(@.["category"] == "fiction")]' | Semua buku dalam kategori fiksi. |
| '$.store.book[?(@.["category"] != "fiction")]' | Semua buku dalam kategori nonfiksi. |

Contoh ekspresi filter tambahan:

```
127.0.0.1:6379> JSON.SET k1 . '{"books": [{"price":5,"sold":true,"in-
stock":true,"title":"foo"}, {"price":15,"sold":false,"title":"abc"}]}'
OK
127.0.0.1:6379> JSON.GET k1 $.books[?(@.price>1&&@.price<20&&@.in-stock)]
"[{\"price\":5,\"sold\":true,\"in-stock\":true,\"title\":\"foo\"}]"
127.0.0.1:6379> JSON.GET k1 '$.books[?(@.price>1 && @.price<20 && @.in-stock)]'
"[{\"price\":5,\"sold\":true,\"in-stock\":true,\"title\":\"foo\"}]"
127.0.0.1:6379> JSON.GET k1 '$.books[?((@.price>1 && @.price<20) && (@.sold==false))]'
"[{\"price\":15,\"sold\":false,\"title\":\"abc\"}]"
127.0.0.1:6379> JSON.GET k1 '$.books[?(@.title == "abc")]'
```

```
[{"price":15,"sold":false,"title":"abc"}]

127.0.0.1:6379> JSON.SET k2 . '[1,2,3,4,5]'
127.0.0.1:6379> JSON.GET k2 $.*.[?(@>2)]
"[3,4,5]"
127.0.0.1:6379> JSON.GET k2 '$.*.[?(@ > 2)]'
"[3,4,5]"

127.0.0.1:6379> JSON.SET k3 . '[true,false,true,false,null,1,2,3,4]'
OK
127.0.0.1:6379> JSON.GET k3 $.*.[?(@==true)]
"[true,true]"
127.0.0.1:6379> JSON.GET k3 '$.*.[?(@ == true)]'
"[true,true]"
127.0.0.1:6379> JSON.GET k3 $.*.[?(@>1)]
"[2,3,4]"
127.0.0.1:6379> JSON.GET k3 '$.*.[?(@ > 1)]'
"[2,3,4]"
```

Sintaks terbatas

| Simbol/Ekspresi | Deskripsi |
| --- | --- |
| . atau [] | Operator turunan. |
| [] | Operator subskrip array. Indeks berbasis 0. |

Contoh

| Jalur | Deskripsi |
| --- | --- |
| .store.book[0].author | Penulis dari buku pertama. |
| .store.book[-1].author | Penulis dari buku terakhir. |
| .address.city | Nama kota. |
| ["store"]["book"][0]["title"] | Judul buku pertama. |
| ["store"]["book"][-1]["title"] | Judul buku terakhir. |

> **Note**
>
> Semua konten Goessner yang dikutip dalam dokumentasi ini diatur dengan Lisensi Creative Commons.

## Awalan kesalahan umum

Setiap pesan kesalahan memiliki awalan. Berikut ini adalah daftar awalan kesalahan umum.

| Awalan | Deskripsi |
| --- | --- |
| ERR | Kesalahan umum. |
| LIMIT | Kesalahan yang terjadi ketika batas ukuran terlampaui. Misalnya, batas ukuran dokumen atau batas kedalaman bersarang telah terlampaui. |
| NONEXISTENT | Kunci atau jalur tidak ada. |
| OUTOFBOUNDARIES | Indeks array di luar batas. |
| SYNTAXERR | Kesalahan sintaks. |
| WRONGTYPE | Jenis nilai yang salah. |

## Metrik terkait JSON

Tersedia metrik info JSON berikut:

| Info | Deskripsi |
| --- | --- |
| json_total_memory_bytes | Total memori yang dialokasikan untuk objek JSON. |
| json_num_documents | Jumlah total dokumen di Redis. |

Untuk mengkueri metrik inti, jalankan perintah Redis berikut:

```
info json_core_metrics
```

## Cara ElastiCache for Redis berinteraksi dengan JSON

Bagian berikut menjelaskan cara ElastiCache for Redis berinteraksi dengan jenis data JSON.

Prasyarat operator

Saat mengevaluasi ekspresi bersyarat untuk pemfilteran, &&s diutamakan terlebih dahulu, lalu ||s dievaluasi, seperti yang umum di sebagian besar bahasa. Operasi di dalam tanda kurung dijalankan terlebih dahulu.

Perilaku batas bersarang jalur maksimum

Batas bersarang jalur maksimum di ElastiCache for Redis adalah 128. Jadi nilai seperti `$.a.b.c.d...` hanya bisa mencapai 128 tingkat.

Menangani nilai numerik

JSON tidak memiliki jenis data terpisah untuk angka integer dan floating point. Semuanya disebut angka.

Representasi numerik:

Ketika angka JSON diterima pada input, angka tersebut diubah menjadi salah satu dari dua representasi biner internal: integer bertanda 64-bit atau floating point presisi ganda IEEE 64-bit. String asli dan semua formatnya tidak dipertahankan. Jadi, ketika angka dihasilkan sebagai bagian dari respons JSON, angka tersebut dikonversi dari representasi biner internal ke string yang dapat dicetak yang menggunakan aturan pemformatan generik. Aturan-aturan ini dapat menghasilkan string yang berbeda dari string yang diterima.

Perintah aritmatika `NUMINCRBY` dan `NUMMULTBY`:

- Jika kedua angka ini adalah integer dan hasilnya berada di luar rentang `int64`, maka angka tersebut secara otomatis akan menjadi angka floating point presisi ganda IEEE 64-bit.
- Jika minimal salah satu angkanya adalah floating point, hasilnya adalah angka floating point presisi ganda IEEE 64-bit.
- Jika hasilnya melebihi rentang IEEE 64-bit ganda, perintah ini menampilkan kesalahan `OVERFLOW`.

Untuk daftar lengkap perintah yang tersedia, lihat Perintah JSON Redis yang didukung.

Pemfilteran array langsung

ElastiCache for Redis memfilter objek array secara langsung.

Untuk data seperti `[0,1,2,3,4,5,6]` dan kueri jalur seperti `$[?(@<4)]`, atau data seperti `{"my_key":[0,1,2,3,4,5,6]}` dan kueri jalur seperti `$.my_key[?(@<4)]`, ElastiCache for Redis akan menampilkan [1,2,3] dalam kedua situasi ini.

Perilaku pengindeksan array

ElastiCache for Redis memungkinkan indeks positif dan negatif untuk array. Untuk array dengan panjang lima, 0 akan mengkueri elemen pertama, 1 yang kedua, dan seterusnya. Angka negatif dimulai pada akhir array, jadi -1 akan mengkueri elemen kelima, -2 akan mengkueri elemen keempat, dan seterusnya.

Untuk memastikan perilaku yang dapat diprediksi bagi pelanggan, ElastiCache for Redis tidak membulatkan indeks array ke bawah atau ke atas, jadi jika Anda memiliki array dengan panjang 5, memanggil indeks 5 atau lebih tinggi, atau -6 atau lebih rendah, tidak akan memberikan hasil.

Evaluasi sintaksis yang ketat

MemoryDB tidak mengizinkan jalur JSON dengan sintaksis yang tidak valid, bahkan jika subset dari jalur berisi jalur yang valid. Hal ini dimaksudkan untuk menjaga perilaku yang benar bagi pelanggan kami.

## Perintah JSON Redis yang didukung

ElastiCache for Redis mendukung perintah JSON Redis berikut:

Topik

- JSON.ARRAPPEND
- JSON.ARRINDEX
- JSON.ARRINSERT
- JSON.ARRLEN
- JSON.ARRPOP
- JSON.ARRTRIM

- JSON.CLEAR
- JSON.DEBUG
- JSON.DEL
- JSON.FORGET
- JSON.GET
- JSON.MGET
- JSON.NUMINCRBY
- JSON.NUMMULTBY
- JSON.OBJLEN
- JSON.OBJKEYS
- JSON.RESP
- JSON.SET
- JSON.STRAPPEND
- JSON.STRLEN
- JSON.TOGGLE
- JSON.TYPE

## JSON.ARRAPPEND

Menambahkan satu atau beberapa nilai ke nilai susunan di jalur.

Sintaks

```
JSON.ARRAPPEND <key> <path> <json> [json ...]
```

- kunci (wajib) – Kunci Redis dari jenis dokumen JSON.
- path (diperlukan) – Sebuah jalur JSON.
- json (diperlukan) - Nilai JSON yang akan ditambahkan ke susunan.

Mengembalikan

Jika jalur adalah sintaksis yang ditingkatkan:

- Susunan integer yang mewakili panjang baru susunan di setiap jalur.
- Jika nilai bukan susunan, nilai yang akan dikembalikan adalah kosong.
- Kesalahan `SYNTAXERR` jika salah satu argumen input json bukan string JSON yang valid.
- Kesalahan `NONEXISTENT` jika jalur tidak ada.

Jika jalur adalah sintaksis terbatas:

- Integer, panjang baru susunan.
- Jika memilih beberapa nilai susunan, perintah mengembalikan panjang baru dari susunan yang terakhir diperbarui.
- Kesalahan `WRONGTYPE` jika nilai di jalur bukan susunan.
- Kesalahan `SYNTAXERR` jika salah satu argumen input json bukan string JSON yang valid.
- Kesalahan `NONEXISTENT` jika jalur tidak ada.

Contoh

Sintaksis jalur yang ditingkatkan:

```
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"]]'
OK
127.0.0.1:6379> JSON.ARRAPPEND  k1 $[*] '"c"'
1) (integer) 1
2) (integer) 2
3) (integer) 3
127.0.0.1:6379> JSON.GET k1
"[[\"c\"],[\"a\",\"c\"],[\"a\",\"b\",\"c\"]]"
```

Sintaksis jalur terbatas:

```
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"]]'
OK
127.0.0.1:6379> JSON.ARRAPPEND  k1 [-1] '"c"'
(integer) 3
127.0.0.1:6379> JSON.GET k1
"[[],[\"a\"],[\"a\",\"b\",\"c\"]]"
```

# JSON.ARRINDEX

Mencari kemunculan pertama dari nilai JSON skalar dalam susunan di jalur.

- Kesalahan di luar jangkauan diatasi dengan membulatkan indeks ke awal dan akhir susunan.
- Jika awal > akhir, mengembalikan -1 (tidak ditemukan).

## Sintaks

```
JSON.ARRINDEX <key> <path> <json-scalar> [start [end]]
```

- kunci (wajib) – Kunci Redis dari jenis dokumen JSON.
- path (diperlukan) – Sebuah jalur JSON.
- json-skalar (diperlukan) - Nilai skalar yang dicari. Skalar JSON mengacu pada nilai yang bukan merupakan objek atau susunan. Yaitu, string, angka, Boolean, dan kosong (null) adalah nilai skalar.
- Awal (opsional) - Indeks awal, inklusif. Default ke 0 jika tidak disediakan.
- akhir (opsional) - Indeks akhir, eksklusif. Default ke 0 jika tidak disediakan, yang berarti bahwa elemen terakhir disertakan. 0 atau -1 berarti elemen terakhir disertakan.

## Mengembalikan

Jika jalur adalah sintaksis yang ditingkatkan:

- Susunan integer. Setiap nilai adalah indeks dari elemen yang cocok dalam susunan di jalur. Nilainya -1 jika tidak ditemukan.
- Jika nilai bukan susunan, nilai yang akan dikembalikan adalah kosong.

Jika jalur adalah sintaksis terbatas:

- Integer, indeks elemen yang cocok, atau -1 jika tidak ditemukan.
- Kesalahan `WRONGTYPE` jika nilai di jalur bukan susunan.

## Contoh

Sintaksis jalur yang ditingkatkan:

```
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"], ["a", "b", "c"]]'
OK
127.0.0.1:6379> JSON.ARRINDEX k1 $[*] '"b"'
1) (integer) -1
2) (integer) -1
3) (integer) 1
4) (integer) 1
```

Sintaksis jalur terbatas:

```
127.0.0.1:6379> JSON.SET k1 . '{"children": ["John", "Jack", "Tom", "Bob", "Mike"]}'
OK
127.0.0.1:6379> JSON.ARRINDEX k1 .children '"Tom"'
(integer) 2
```

## JSON.ARRINSERT

Menyisipkan satu atau beberapa nilai ke dalam nilai susunan di jalur sebelum indeks.

Sintaks

```
JSON.ARRINSERT <key> <path> <index> <json> [json ...]
```

- kunci (wajib) – Kunci Redis dari jenis dokumen JSON.
- path (diperlukan) - Sebuah jalur JSON.
- index (diperlukan) — Sebuah indeks susunan yang dimasukkan setelah nilai.
- json (diperlukan) - Nilai JSON yang akan ditambahkan ke susunan.

Mengembalikan

Jika jalur adalah sintaksis yang ditingkatkan:

- Susunan integer yang mewakili panjang baru susunan di setiap jalur.
- Jika nilai adalah susunan kosong, nilai yang akan dikembalikan adalah kosong.
- Jika nilai bukan susunan, nilai yang akan dikembalikan adalah kosong.
- Kesalahan `OUTOFBOUNDARIES` jika argumen indeks di luar batas.

Jika jalur adalah sintaksis terbatas:

- Integer, panjang baru susunan.
- Kesalahan `WRONGTYPE` jika nilai di jalur bukan susunan.
- Kesalahan `OUTOFBOUNDARIES` jika argumen indeks di luar batas.

Contoh

Sintaksis jalur yang ditingkatkan:

```
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"]]'
OK
127.0.0.1:6379> JSON.ARRINSERT k1 $[*] 0 '"c"'
1) (integer) 1
2) (integer) 2
3) (integer) 3
127.0.0.1:6379> JSON.GET k1
"[[\"c\"],[\"c\",\"a\"],[\"c\",\"a\",\"b\"]]"
```

Sintaksis jalur terbatas:

```
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"]]'
OK
127.0.0.1:6379> JSON.ARRINSERT k1 . 0 '"c"'
(integer) 4
127.0.0.1:6379> JSON.GET k1
"[\"c\",[],[\"a\"],[\"a\",\"b\"]]"
```

## JSON.ARRLEN

Mendapat panjang nilai susunan di jalur.

Sintaks

```
JSON.ARRLEN <key> [path]
```

- kunci (wajib) – Kunci Redis dari jenis dokumen JSON.
- jalur (opsional) – Sebuah jalur JSON. Diatur secara default ke root jika tidak disediakan.

Nilai yang ditampilkan

Jika jalur adalah sintaksis yang ditingkatkan:

- Susunan integer yang mewakili panjang baru susunan di setiap jalur.
- Jika nilai bukan susunan, nilai yang akan dikembalikan adalah kosong.
- Kosong jika kunci dokumen tidak ada.

Jika jalur adalah sintaksis terbatas:

- Susunan string massal. Setiap elemen adalah nama kunci dalam objek.
- Integer, panjang susunan.
- Jika memilih beberapa objek, perintah mengembalikan panjang susunan pertama.
- Kesalahan `WRONGTYPE` jika nilai di jalur bukan susunan.
- Kesalahan `WRONGTYPE` jika jalur tidak ada.
- Kosong jika kunci dokumen tidak ada.

Contoh

Sintaksis jalur yang ditingkatkan:

```
127.0.0.1:6379> JSON.SET k1 . '[[], [\"a\"], [\"a\", \"b\"], [\"a\", \"b\", \"c\"]]'
(error) SYNTAXERR Failed to parse JSON string due to syntax error
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"], ["a", "b", "c"]]'
OK
127.0.0.1:6379> JSON.ARRLEN k1 $[*]
1) (integer) 0
2) (integer) 1
3) (integer) 2
4) (integer) 3

127.0.0.1:6379> JSON.SET k2 . '[[], "a", ["a", "b"], ["a", "b", "c"], 4]'
OK
127.0.0.1:6379> JSON.ARRLEN k2 $[*]
1) (integer) 0
2) (nil)
3) (integer) 2
4) (integer) 3
5) (nil)
```

Sintaksis jalur terbatas:

```
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"], ["a", "b", "c"]]'
OK
127.0.0.1:6379> JSON.ARRLEN k1 [*]
(integer) 0
127.0.0.1:6379> JSON.ARRLEN k1 $[3]
1) (integer) 3

127.0.0.1:6379> JSON.SET k2 . '[[], "a", ["a", "b"], ["a", "b", "c"], 4]'
OK
127.0.0.1:6379> JSON.ARRLEN k2 [*]
(integer) 0
127.0.0.1:6379> JSON.ARRLEN k2 $[1]
1) (nil)
127.0.0.1:6379> JSON.ARRLEN k2 $[2]
1) (integer) 2
```

## JSON.ARRPOP

Menghapus dan mengembalikan elemen pada indeks dari susunan. Memunculkan susunan kosong mengembalikan null.

Sintaks

```
JSON.ARRPOP <key> [path [index]]
```

- kunci (wajib) – Kunci Redis dari jenis dokumen JSON.
- jalur (opsional) – Sebuah jalur JSON. Default ke root jika tidak disediakan.
- indeks (opsional) - Posisi dalam susunan yang menjadi asal kemunculan.
  - Default ke -1 jika tidak disediakan, yang berarti elemen terakhir.
  - Nilai negatif berarti posisi dari elemen terakhir.
  - Indeks di luar batas akan dibulatkan ke batas susunan masing-masing.

Mengembalikan

Jika jalur adalah sintaksis yang ditingkatkan:

- Susunan string massal yang mewakili nilai yang muncul di setiap jalur.
- Jika nilai adalah susunan kosong, nilai yang akan dikembalikan adalah kosong.
- Jika nilai bukan susunan, nilai yang akan dikembalikan adalah kosong.

Jika jalur adalah sintaksis terbatas:

- String massal, yang mewakili nilai JSON yang muncul.
- Null jika susunan kosong.
- Kesalahan `WRONGTYPE` jika nilai di jalur bukan susunan.

Contoh

Sintaksis jalur yang ditingkatkan:

```
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"]]'
OK
127.0.0.1:6379> JSON.ARRPOP k1 $[*]
1) (nil)
2) "\"a\""
3) "\"b\""
127.0.0.1:6379> JSON.GET k1
"[[],[],[\"a\"]]"
```

Sintaksis jalur terbatas:

```
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"]]'
OK
127.0.0.1:6379> JSON.ARRPOP k1
"[\"a\",\"b\"]"
127.0.0.1:6379> JSON.GET k1
"[[],[\"a\"]]"

127.0.0.1:6379> JSON.SET k2 . '[[], ["a"], ["a", "b"]]'
OK
127.0.0.1:6379> JSON.ARRPOP k2 . 0
"[]"
127.0.0.1:6379> JSON.GET k2
```

```
"[[\"a\"],[\"a\",\"b\"]]"
```

## JSON.ARRTRIM

Memangkas susunan di jalur sehingga menjadi subarray [awal, akhir], keduanya inklusif.

- Jika susunan kosong, tidak melakukan apa pun, mengembalikan 0.
- Jika awal <0, perlakukan itu sebagai 0.
- Jika akhir >= ukuran (ukuran susunan), perlakukan itu sebagai ukuran-1.
- Jika awal >= ukuran atau awal > akhir, kosongkan susunan dan mengembalikan 0.

### Sintaks

```
JSON.ARRINSERT <key> <path> <start> <end>
```

- kunci (wajib) – Kunci Redis dari jenis dokumen JSON.
- path (diperlukan) – Sebuah jalur JSON.
- Awal (diperlukan) – Indeks awal, inklusif.
- akhir (diperlukan) – Indeks akhir, inklusif.

### Mengembalikan

Jika jalur adalah sintaksis yang ditingkatkan:

- Susunan integer yang mewakili panjang baru susunan di setiap jalur.
- Jika nilai adalah susunan kosong, nilai yang akan dikembalikan adalah kosong.
- Jika nilai bukan susunan, nilai yang akan dikembalikan adalah kosong.
- Kesalahan `OUTOFBOUNDARIES` jika argumen indeks di luar batas.

Jika jalur adalah sintaksis terbatas:

- Integer, panjang baru susunan.
- Null jika susunan kosong.
- Kesalahan `WRONGTYPE` jika nilai di jalur bukan susunan.

- Kesalahan `OUTOFBOUNDARIES` jika argumen indeks di luar batas.

Contoh

Sintaksis jalur yang ditingkatkan:

```
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"], ["a", "b", "c"]]'
OK
127.0.0.1:6379> JSON.ARRTRIM k1 $[*] 0 1
1) (integer) 0
2) (integer) 1
3) (integer) 2
4) (integer) 2
   127.0.0.1:6379> JSON.GET k1
   "[[],[\"a\"],[\"a\",\"b\"],[\"a\",\"b\"]]"
```

Sintaksis jalur terbatas:

```
127.0.0.1:6379> JSON.SET k1 . '{"children": ["John", "Jack", "Tom", "Bob", "Mike"]}'
OK
127.0.0.1:6379> JSON.ARRTRIM k1 .children 0 1
(integer) 2
127.0.0.1:6379> JSON.GET k1 .children
"[\"John\",\"Jack\"]"
```

## JSON.CLEAR

Membersihkan susunan atau objek di jalur.

Sintaks

```
JSON.CLEAR <key> [path]
```

- kunci (wajib) – Kunci Redis dari jenis dokumen JSON.
- jalur (opsional) – Sebuah jalur JSON. Diatur secara default ke root jika tidak disediakan.

Mengembalikan

- Integer, jumlah kontainer dihapus.
- Menghapus susunan kosong atau akun objek untuk 1 kontainer yang dihapus.
- Menghapus nilai non-kontainer mengembalikan 0.

Contoh

```
127.0.0.1:6379> JSON.SET k1 . '[[], [0], [0,1], [0,1,2], 1, true, null, "d"]'
OK
127.0.0.1:6379>  JSON.CLEAR k1  $[*]
(integer) 7
127.0.0.1:6379> JSON.CLEAR k1  $[*]
(integer) 4
127.0.0.1:6379> JSON.SET k2 . '{"children": ["John", "Jack", "Tom", "Bob", "Mike"]}'
OK
127.0.0.1:6379> JSON.CLEAR k2 .children
(integer) 1
127.0.0.1:6379> JSON.GET k2 .children
"[]"
```

## JSON.DEBUG

Melaporkan informasi. Subperintah yang didukung adalah:

- MEMORY <key> [path] – Melaporkan penggunaan memori dalam byte dari nilai JSON. Jalur akan diatur secara default ke root jika tidak disediakan.
- FIELDS <key> [path] – Melaporkan jumlah bidang di jalur dokumen yang ditentukan. Jalur akan diatur secara default ke root jika tidak disediakan. Setiap nilai JSON non-kontainer dihitung sebagai satu bidang. Objek dan array secara rekursif menghitung satu bidang untuk masing-masing nilai JSON. Setiap nilai kontainer, kecuali kontainer root, dihitung sebagai satu bidang tambahan.
- HELP – Mencetak pesan bantuan dari perintah.

Sintaks

```
JSON.DEBUG <subcommand & arguments>
```

Tergantung pada subperintah:

MEMORY

- Jika jalur adalah sintaksis yang ditingkatkan:
  - Menampilkan array integer yang merepresentasikan ukuran memori (dalam byte) dari nilai JSON di setiap jalur.
  - Menampilkan array kosong jika kunci Redis tidak ada.
- Jika jalur adalah sintaksis terbatas:
  - Menampilkan integer, ukuran memori, dan nilai JSON dalam byte.
  - Menampilkan kosong jika kunci Redis tidak ada.

FIELD

- Jika jalur adalah sintaksis yang ditingkatkan:
  - Menampilkan array integer yang merepresentasikan jumlah bidang nilai JSON di setiap jalur.
  - Menampilkan array kosong jika kunci Redis tidak ada.
- Jika jalur adalah sintaksis terbatas:
  - Menampilkan integer, jumlah bidang nilai JSON.
  - Menampilkan kosong jika kunci Redis tidak ada.

HELP – Menampilkan array pesan bantuan.

Contoh

Sintaksis jalur yang ditingkatkan:

```
127.0.0.1:6379> JSON.SET k1 . '[1, 2.3, "foo", true, null, {}, [], {"a":1, "b":2},
 [1,2,3]]'
OK
127.0.0.1:6379> JSON.DEBUG MEMORY k1 $[*]
1) (integer) 16
2) (integer) 16
3) (integer) 19
4) (integer) 16
5) (integer) 16
6) (integer) 16
7) (integer) 16
8) (integer) 50
9) (integer) 64
127.0.0.1:6379> JSON.DEBUG FIELDS k1 $[*]
```

```
1) (integer) 1
2) (integer) 1
3) (integer) 1
4) (integer) 1
5) (integer) 1
6) (integer) 0
7) (integer) 0
8) (integer) 2
9) (integer) 3
```

Sintaksis jalur terbatas:

```
127.0.0.1:6379> JSON.SET k1 .
 '{"firstName":"John","lastName":"Smith","age":27,"weight":135.25,"isAlive":true,"address":
{"street":"21 2nd Street","city":"New
 York","state":"NY","zipcode":"10021-3100"},"phoneNumbers":
[{"type":"home","number":"212 555-1234"},{"type":"office","number":"646
 555-4567"}],"children":[],"spouse":null}'
OK
127.0.0.1:6379> JSON.DEBUG MEMORY k1
(integer) 632
127.0.0.1:6379> JSON.DEBUG MEMORY k1 .phoneNumbers
(integer) 166

127.0.0.1:6379> JSON.DEBUG FIELDS k1
(integer) 19
127.0.0.1:6379> JSON.DEBUG FIELDS k1 .address
(integer) 4

127.0.0.1:6379> JSON.DEBUG HELP
1) JSON.DEBUG MEMORY <key> [path] - report memory size (bytes) of the JSON element.
 Path defaults to root if not provided.
2) JSON.DEBUG FIELDS <key> [path] - report number of fields in the JSON element. Path
 defaults to root if not provided.
3) JSON.DEBUG HELP - print help message.
```

## JSON.DEL

Menghapus nilai JSON pada jalur dalam kunci dokumen. Jika jalurnya adalah root, itu sama dengan menghapus kunci dari Redis.

### Sintaks

```
JSON.DEL <key> [path]
```

- kunci (wajib) – Kunci Redis dari jenis dokumen JSON.
- jalur (opsional) – Sebuah jalur JSON. Diatur secara default ke root jika tidak disediakan.

### Mengembalikan

- Jumlah elemen yang dihapus.
- 0 jika kunci Redis tidak ada.
- 0 jika jalur JSON tidak valid atau tidak ada.

### Contoh

Sintaksis jalur yang ditingkatkan:

```
127.0.0.1:6379> JSON.SET k1 . '{"a":{}, "b":{"a":1}, "c":{"a":1, "b":2}, "d":{"a":1,
 "b":2, "c":3}, "e": [1,2,3,4,5]}'
OK
127.0.0.1:6379> JSON.DEL k1 $.d.*
(integer) 3
127.0.0.1:6379> JSOn.GET k1
"{\"a\":{},\"b\":{\"a\":1},\"c\":{\"a\":1,\"b\":2},\"d\":{},\"e\":[1,2,3,4,5]}"
127.0.0.1:6379> JSON.DEL k1 $.e[*]
(integer) 5
127.0.0.1:6379> JSOn.GET k1
"{\"a\":{},\"b\":{\"a\":1},\"c\":{\"a\":1,\"b\":2},\"d\":{},\"e\":[]}"
```

Sintaksis jalur terbatas:

```
127.0.0.1:6379> JSON.SET k1 . '{"a":{}, "b":{"a":1}, "c":{"a":1, "b":2}, "d":{"a":1,
 "b":2, "c":3}, "e": [1,2,3,4,5]}'
OK
127.0.0.1:6379> JSON.DEL k1 .d.*
(integer) 3
127.0.0.1:6379> JSON.GET k1
"{\"a\":{},\"b\":{\"a\":1},\"c\":{\"a\":1,\"b\":2},\"d\":{},\"e\":[1,2,3,4,5]}"
```

```
127.0.0.1:6379> JSON.DEL k1 .e[*]
(integer) 5
127.0.0.1:6379> JSON.GET k1
"{\"a\":{},\"b\":{\"a\":1},\"c\":{\"a\":1,\"b\":2},\"d\":{},\"e\":[]}"
```

## JSON.FORGET

Sebuah alias dari JSON.DEL.

## JSON.GET

Mengembalikan JSON serial pada satu atau beberapa jalur.

Sintaks

```
JSON.GET <key>
[INDENT indentation-string]
[NEWLINE newline-string]
[SPACE space-string]
[NOESCAPE]
[path ...]
```

- key (diperlukan) – Kunci Redis dari jenis dokumen JSON.
- INDENT/NEWLINE/SPACE (opsional) - Mengontrol format string JSON yang dikembalikan, yaitu, “cetak cantik” (pretty print). Nilai default masing-masing adalah string kosong. Itu dapat diganti dalam kombinasi apa pun. Serta dapat ditentukan dalam urutan apa pun.
- NOESCAPE - Opsional, diizinkan hadir untuk kompatibilitas lama dan tidak memiliki efek lain.
- jalur (opsional) - Nol atau lebih jalur JSON, default ke root jika tidak ada yang diberikan. Argumen jalur harus ditempatkan di akhir.

Mengembalikan

Sintaks jalur yang ditingkatkan:

Jika disediakan satu jalur:

- Mengembalikan string serial dari susunan nilai.
- Jika tidak ada nilai yang dipilih, perintah mengembalikan susunan kosong.

Jika diberikan beberapa jalur:

- Mengembalikan objek JSON ter-string, di mana setiap jalur adalah kunci.
- Jika terdapat campuran sintaks jalur yang ditingkatkan dan dibatasi, hasilnya sesuai dengan sintaks yang ditingkatkan.
- Jika jalur tidak ada, nilai yang sesuai adalah susunan kosong.

Contoh

Sintaksis jalur yang ditingkatkan:

```
127.0.0.1:6379> JSON.SET k1 .
 '{"firstName":"John","lastName":"Smith","age":27,"weight":135.25,"isAlive":true,"address":
{"street":"21 2nd Street","city":"New
 York","state":"NY","zipcode":"10021-3100"},"phoneNumbers":
[{"type":"home","number":"212 555-1234"},{"type":"office","number":"646
 555-4567"}],"children":[],"spouse":null}'
OK
127.0.0.1:6379> JSON.GET k1 $.address.*
"[\"21 2nd Street\",\"New York\",\"NY\",\"10021-3100\"]"
127.0.0.1:6379> JSON.GET k1 indent "\t" space " " NEWLINE "\n" $.address.*
"[\n\t\"21 2nd Street\",\n\t\"New York\",\n\t\"NY\",\n\t\"10021-3100\"\n]"
127.0.0.1:6379> JSON.GET k1 $.firstName $.lastName $.age
"{\"$.firstName\":[\"John\"],\"$.lastName\":[\"Smith\"],\"$.age\":[27]}"
127.0.0.1:6379> JSON.SET k2 . '{"a":{}, "b":{"a":1}, "c":{"a":1, "b":2}}'
OK
127.0.0.1:6379> json.get k2 $..*
"[{},{\"a\":1},{\"a\":1,\"b\":2},1,1,2]"
```

Sintaksis jalur terbatas:

```
 127.0.0.1:6379> JSON.SET k1 .
 '{"firstName":"John","lastName":"Smith","age":27,"weight":135.25,"isAlive":true,"address":
{"street":"21 2nd Street","city":"New
 York","state":"NY","zipcode":"10021-3100"},"phoneNumbers":
[{"type":"home","number":"212 555-1234"},{"type":"office","number":"646
 555-4567"}],"children":[],"spouse":null}'
OK
127.0.0.1:6379> JSON.GET k1 .address
```

```
"{\"street\":\"21 2nd Street\",\"city\":\"New York\",\"state\":\"NY\",\"zipcode\":
\"10021-3100\"}"
127.0.0.1:6379> JSON.GET k1 indent "\t" space " " NEWLINE "\n" .address
"{\n\t\"street\": \"21 2nd Street\",\n\t\"city\": \"New York\",\n\t\"state\": \"NY\",\n
\t\"zipcode\": \"10021-3100\"\n}"
127.0.0.1:6379> JSON.GET k1 .firstName .lastName .age
"{\".firstName\":\"John\",\".lastName\":\"Smith\",\".age\":27}"
```

## JSON.MGET

Mendapatkan JSON serial di jalur dari beberapa kunci dokumen. Itu mengembalikan data kosong untuk kunci atau jalur JSON yang tidak ada.

Sintaks

```
JSON.MGET <key> [key ...] <path>
```

- kunci (wajib) - Satu atau beberapa kunci Redis dari jenis dokumen.
- path (diperlukan) – Sebuah jalur JSON.

Mengembalikan

- Susunan string massal. Ukuran susunan sama dengan jumlah kunci dalam perintah. Setiap elemen susunan diisi dengan (a) JSON serial seperti yang terletak di jalur atau (b) kosong jika kunci tidak ada, jalur tidak ada dalam dokumen, atau jalur tidak valid (kesalahan sintaks).
- Jika salah satu kunci yang ditentukan ada dan bukan kunci JSON, perintah mengembalikan kesalahan `WRONGTYPE`.

Contoh

Sintaksis jalur yang ditingkatkan:

```
127.0.0.1:6379> JSON.SET k1 . '{"address":{"street":"21 2nd Street","city":"New
 York","state":"NY","zipcode":"10021"}}'
OK
127.0.0.1:6379> JSON.SET k2 . '{"address":{"street":"5 main
 Street","city":"Boston","state":"MA","zipcode":"02101"}}'
OK
```

```
127.0.0.1:6379> JSON.SET k3 . '{"address":{"street":"100 Park
 Ave","city":"Seattle","state":"WA","zipcode":"98102"}}'
OK
127.0.0.1:6379> JSON.MGET k1 k2 k3 $.address.city
1) "[\"New York\"]"
2) "[\"Boston\"]"
3) "[\"Seattle\"]"
```

Sintaksis jalur terbatas:

```
127.0.0.1:6379> JSON.SET k1 . '{"address":{"street":"21 2nd Street","city":"New
 York","state":"NY","zipcode":"10021"}}'
OK
127.0.0.1:6379> JSON.SET k2 . '{"address":{"street":"5 main
 Street","city":"Boston","state":"MA","zipcode":"02101"}}'
OK
127.0.0.1:6379> JSON.SET k3 . '{"address":{"street":"100 Park
 Ave","city":"Seattle","state":"WA","zipcode":"98102"}}'
OK

127.0.0.1:6379> JSON.MGET k1 k2 k3 .address.city
1) "\"New York\""
2) "\"Seattle\""
3) "\"Seattle\""
```

## JSON.NUMINCRBY

Menambah nilai angka di jalur dengan nomor tertentu.

Sintaks

```
JSON.NUMINCRBY <key> <path> <number>
```

- kunci (wajib) – Kunci Redis dari jenis dokumen JSON.
- jalur (wajib) – Sebuah jalur JSON.
- nomor (wajib) — Sebuah angka.

Mengembalikan

Jika jalur adalah sintaksis yang ditingkatkan:

- Susunan string massal yang mewakili nilai yang muncul di setiap jalur.
- Jika nilai bukan nomor, nilai yang akan dikembalikan adalah kosong.
- Kesalahan `WRONGTYPE` jika nomor tidak dapat diuraikan.
- Kesalahan `OVERFLOW` jika hasilnya di luar rentang 64-bit IEEE ganda.
- `NONEXISTENT` jika kunci dokumen tidak ada.

Jika jalur adalah sintaksis terbatas:

- String massal yang mewakili nilai yang dikembalikan.
- Jika memilih beberapa nilai, perintah mengembalikan hasil nilai yang terakhir diperbarui.
- Kesalahan `WRONGTYPE` jika nilai di jalur bukan nomor.
- Kesalahan `WRONGTYPE` jika nomor tidak dapat diuraikan.
- Kesalahan `OVERFLOW` jika hasilnya di luar rentang 64-bit IEEE ganda.
- `NONEXISTENT` jika kunci dokumen tidak ada.

Contoh

Sintaksis jalur yang ditingkatkan:

```
127.0.0.1:6379> JSON.SET k1 . '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
OK
127.0.0.1:6379> JSON.NUMINCRBY k1 $.d[*] 10
"[11,12,13]"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[1],\"c\":[1,2],\"d\":[11,12,13]}"

127.0.0.1:6379> JSON.SET k1 $ '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
OK
127.0.0.1:6379> JSON.NUMINCRBY k1 $.a[*] 1
"[]"
127.0.0.1:6379> JSON.NUMINCRBY k1 $.b[*] 1
"[2]"
127.0.0.1:6379> JSON.NUMINCRBY k1 $.c[*] 1
"[2,3]"
127.0.0.1:6379> JSON.NUMINCRBY k1 $.d[*] 1
"[2,3,4]"
```

```
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[2],\"c\":[2,3],\"d\":[2,3,4]}"

127.0.0.1:6379> JSON.SET k2 $ '{"a":{}, "b":{"a":1}, "c":{"a":1, "b":2}, "d":{"a":1,
 "b":2, "c":3}}'
OK
127.0.0.1:6379> JSON.NUMINCRBY k2 $.a.* 1
"[]"
127.0.0.1:6379> JSON.NUMINCRBY k2 $.b.* 1
"[2]"
127.0.0.1:6379> JSON.NUMINCRBY k2 $.c.* 1
"[2,3]"
127.0.0.1:6379> JSON.NUMINCRBY k2 $.d.* 1
"[2,3,4]"
127.0.0.1:6379> JSON.GET k2
"{\"a\":{},\"b\":{\"a\":2},\"c\":{\"a\":2,\"b\":3},\"d\":{\"a\":2,\"b\":3,\"c\":4}}"

127.0.0.1:6379> JSON.SET k3 $ '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a",
 "b":"b"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.NUMINCRBY k3 $.a.* 1
"[null]"
127.0.0.1:6379> JSON.NUMINCRBY k3 $.b.* 1
"[null,2]"
127.0.0.1:6379> JSON.NUMINCRBY k3 $.c.* 1
"[null,null]"
127.0.0.1:6379> JSON.NUMINCRBY k3 $.d.* 1
"[2,null,4]"
127.0.0.1:6379> JSON.GET k3
"{\"a\":{\"a\":\"a\"},\"b\":{\"a\":\"a\",\"b\":2},\"c\":{\"a\":\"a\",\"b\":\"b\"},\"d
\":{\"a\":2,\"b\":\"b\",\"c\":4}}"
```

Sintaksis jalur terbatas:

```
127.0.0.1:6379> JSON.SET k1 . '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
OK
127.0.0.1:6379> JSON.NUMINCRBY k1 .d[1] 10
"12"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[1],\"c\":[1,2],\"d\":[1,12,3]}"

127.0.0.1:6379> JSON.SET k1 . '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
```

```
OK
127.0.0.1:6379> JSON.NUMINCRBY k1 .a[*] 1
(error) NONEXISTENT JSON path does not exist
127.0.0.1:6379> JSON.NUMINCRBY k1 .b[*] 1
"2"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[2],\"c\":[1,2],\"d\":[1,2,3]}"
127.0.0.1:6379> JSON.NUMINCRBY k1 .c[*] 1
"3"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[2],\"c\":[2,3],\"d\":[1,2,3]}"
127.0.0.1:6379> JSON.NUMINCRBY k1 .d[*] 1
"4"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[2],\"c\":[2,3],\"d\":[2,3,4]}"

127.0.0.1:6379> JSON.SET k2 . '{"a":{}, "b":{"a":1}, "c":{"a":1, "b":2}, "d":{"a":1,
 "b":2, "c":3}}'
OK
127.0.0.1:6379> JSON.NUMINCRBY k2 .a.* 1
(error) NONEXISTENT JSON path does not exist
127.0.0.1:6379> JSON.NUMINCRBY k2 .b.* 1
"2"
127.0.0.1:6379> JSON.GET k2
"{\"a\":{},\"b\":{\"a\":2},\"c\":{\"a\":1,\"b\":2},\"d\":{\"a\":1,\"b\":2,\"c\":3}}"
127.0.0.1:6379> JSON.NUMINCRBY k2 .c.* 1
"3"
127.0.0.1:6379> JSON.GET k2
"{\"a\":{},\"b\":{\"a\":2},\"c\":{\"a\":2,\"b\":3},\"d\":{\"a\":1,\"b\":2,\"c\":3}}"
127.0.0.1:6379> JSON.NUMINCRBY k2 .d.* 1
"4"
127.0.0.1:6379> JSON.GET k2
"{\"a\":{},\"b\":{\"a\":2},\"c\":{\"a\":2,\"b\":3},\"d\":{\"a\":2,\"b\":3,\"c\":4}}"

127.0.0.1:6379> JSON.SET k3 . '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a",
 "b":"b"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.NUMINCRBY k3 .a.* 1
(error) WRONGTYPE JSON element is not a number
127.0.0.1:6379> JSON.NUMINCRBY k3 .b.* 1
"2"
127.0.0.1:6379> JSON.NUMINCRBY k3 .c.* 1
(error) WRONGTYPE JSON element is not a number
127.0.0.1:6379> JSON.NUMINCRBY k3 .d.* 1
```

```
"4"
```

## JSON.NUMMULTBY

Menggandakan nilai angka di jalur dengan nomor tertentu.

Sintaks

```
JSON.NUMMULTBY <key> <path> <number>
```

- kunci (wajib) – Kunci Redis dari jenis dokumen JSON.
- jalur (wajib) – Sebuah jalur JSON.
- nomor (wajib) — Sebuah angka.

Mengembalikan

Jika jalur adalah sintaksis yang ditingkatkan:

- Susunan string massal yang mewakili nilai yang muncul di setiap jalur.
- Jika nilai bukan nomor, nilai yang akan dikembalikan adalah kosong.
- Kesalahan `WRONGTYPE` jika nomor tidak dapat diuraikan.
- Kesalahan `OVERFLOW` jika hasilnya di luar rentang 64-bit IEEE angka floating point presisi ganda.
- `NONEXISTENT` jika kunci dokumen tidak ada.

Jika jalur adalah sintaksis terbatas:

- String massal yang mewakili nilai yang dikembalikan.
- Jika memilih beberapa nilai, perintah mengembalikan hasil nilai yang terakhir diperbarui.
- Kesalahan `WRONGTYPE` jika nilai di jalur bukan nomor.
- Kesalahan `WRONGTYPE` jika nomor tidak dapat diuraikan.
- Kesalahan `OVERFLOW` jika hasilnya di luar rentang 64-bit IEEE ganda.
- `NONEXISTENT` jika kunci dokumen tidak ada.

Contoh

Sintaksis jalur yang ditingkatkan:

```
127.0.0.1:6379> JSON.SET k1 . '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k1 $.d[*] 2
"[2,4,6]"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[1],\"c\":[1,2],\"d\":[2,4,6]}"

127.0.0.1:6379> JSON.SET k1 $ '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k1 $.a[*] 2
"[]"
127.0.0.1:6379> JSON.NUMMULTBY k1 $.b[*] 2
"[2]"
127.0.0.1:6379> JSON.NUMMULTBY k1 $.c[*] 2
"[2,4]"
127.0.0.1:6379> JSON.NUMMULTBY k1 $.d[*] 2
"[2,4,6]"

127.0.0.1:6379> JSON.SET k2 $ '{"a":{}, "b":{"a":1}, "c":{"a":1, "b":2}, "d":{"a":1,
 "b":2, "c":3}}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k2 $.a.* 2
"[]"
127.0.0.1:6379> JSON.NUMMULTBY k2 $.b.* 2
"[2]"
127.0.0.1:6379> JSON.NUMMULTBY k2 $.c.* 2
"[2,4]"
127.0.0.1:6379> JSON.NUMMULTBY k2 $.d.* 2
"[2,4,6]"

127.0.0.1:6379> JSON.SET k3 $ '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a",
 "b":"b"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k3 $.a.* 2
"[null]"
127.0.0.1:6379> JSON.NUMMULTBY k3 $.b.* 2
"[null,2]"
127.0.0.1:6379> JSON.NUMMULTBY k3 $.c.* 2
"[null,null]"
127.0.0.1:6379> JSON.NUMMULTBY k3 $.d.* 2
"[2,null,6]"
```

Sintaksis jalur terbatas:

```
127.0.0.1:6379> JSON.SET k1 . '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k1 .d[1] 2
"4"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[1],\"c\":[1,2],\"d\":[1,4,3]}"

127.0.0.1:6379> JSON.SET k1 . '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k1 .a[*] 2
(error) NONEXISTENT JSON path does not exist
127.0.0.1:6379> JSON.NUMMULTBY k1 .b[*] 2
"2"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[2],\"c\":[1,2],\"d\":[1,2,3]}"
127.0.0.1:6379> JSON.NUMMULTBY k1 .c[*] 2
"4"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[2],\"c\":[2,4],\"d\":[1,2,3]}"
127.0.0.1:6379> JSON.NUMMULTBY k1 .d[*] 2
"6"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[2],\"c\":[2,4],\"d\":[2,4,6]}"

127.0.0.1:6379> JSON.SET k2 . '{"a":{}, "b":{"a":1}, "c":{"a":1, "b":2}, "d":{"a":1,
 "b":2, "c":3}}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k2 .a.* 2
(error) NONEXISTENT JSON path does not exist
127.0.0.1:6379> JSON.NUMMULTBY k2 .b.* 2
"2"
127.0.0.1:6379> JSON.GET k2
"{\"a\":{},\"b\":{\"a\":2},\"c\":{\"a\":1,\"b\":2},\"d\":{\"a\":1,\"b\":2,\"c\":3}}"
127.0.0.1:6379> JSON.NUMMULTBY k2 .c.* 2
"4"
127.0.0.1:6379> JSON.GET k2
"{\"a\":{},\"b\":{\"a\":2},\"c\":{\"a\":2,\"b\":4},\"d\":{\"a\":1,\"b\":2,\"c\":3}}"
127.0.0.1:6379> JSON.NUMMULTBY k2 .d.* 2
"6"
```

```
127.0.0.1:6379> JSON.GET k2
"{\"a\":{},\"b\":{\"a\":2},\"c\":{\"a\":2,\"b\":4},\"d\":{\"a\":2,\"b\":4,\"c\":6}}"

127.0.0.1:6379> JSON.SET k3 . '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a",
 "b":"b"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k3 .a.* 2
(error) WRONGTYPE JSON element is not a number
127.0.0.1:6379> JSON.NUMMULTBY k3 .b.* 2
"2"
127.0.0.1:6379> JSON.GET k3
"{\"a\":{\"a\":\"a\"},\"b\":{\"a\":\"a\",\"b\":2},\"c\":{\"a\":\"a\",\"b\":\"b\"},\"d
\":{\"a\":1,\"b\":\"b\",\"c\":3}}"
127.0.0.1:6379> JSON.NUMMULTBY k3 .c.* 2
(error) WRONGTYPE JSON element is not a number
127.0.0.1:6379> JSON.NUMMULTBY k3 .d.* 2
"6"
127.0.0.1:6379> JSON.GET k3
"{\"a\":{\"a\":\"a\"},\"b\":{\"a\":\"a\",\"b\":2},\"c\":{\"a\":\"a\",\"b\":\"b\"},\"d
\":{\"a\":2,\"b\":\"b\",\"c\":6}}"
```

## JSON.OBJLEN

Mendapat jumlah kunci dalam nilai objek di jalur.

Sintaks

```
JSON.OBJLEN <key> [path]
```

- kunci (wajib) – Kunci Redis dari jenis dokumen JSON.
- jalur (opsional) – Sebuah jalur JSON. Diatur secara default ke root jika tidak disediakan.

Nilai yang ditampilkan

Jika jalur adalah sintaksis yang ditingkatkan:

- Susunan integer yang mewakili panjang susunan di setiap jalur.
- Jika nilai bukan objek, nilai yang akan dikembalikan adalah nilai yang kosong.
- Kosong jika kunci dokumen tidak ada.

Jika jalur adalah sintaksis terbatas:

- Integer, jumlah kunci dalam objek.
- Jika memilih beberapa objek, perintah mengembalikan panjang susunan pertama.
- Kesalahan WRONGTYPE jika nilai di jalur bukan objek.
- Kesalahan WRONGTYPE jika jalur tidak ada.
- Kosong jika kunci dokumen tidak ada.

Contoh

Sintaksis jalur yang ditingkatkan:

```
127.0.0.1:6379> JSON.SET k1 $ '{"a":{}, "b":{"a":"a"}, "c":{"a":"a", "b":"bb"}, "d":
{"a":1, "b":"b", "c":{"a":3,"b":4}}, "e":1}'
OK
127.0.0.1:6379> JSON.OBJLEN k1 $.a
1) (integer) 0
127.0.0.1:6379> JSON.OBJLEN k1 $.a.*
(empty array)
127.0.0.1:6379> JSON.OBJLEN k1 $.b
1) (integer) 1
127.0.0.1:6379> JSON.OBJLEN k1 $.b.*
1) (nil)
127.0.0.1:6379> JSON.OBJLEN k1 $.c
1) (integer) 2
127.0.0.1:6379> JSON.OBJLEN k1 $.c.*
1) (nil)
2) (nil)
127.0.0.1:6379> JSON.OBJLEN k1 $.d
1) (integer) 3
127.0.0.1:6379> JSON.OBJLEN k1 $.d.*
1) (nil)
2) (nil)
3) (integer) 2
127.0.0.1:6379> JSON.OBJLEN k1 $.*
1) (integer) 0
2) (integer) 1
3) (integer) 2
4) (integer) 3
5) (nil)
```

Sintaksis jalur terbatas:

```
127.0.0.1:6379> JSON.SET k1 . '{"a":{}, "b":{"a":"a"}, "c":{"a":"a", "b":"bb"}, "d":
{"a":1, "b":"b", "c":{"a":3,"b":4}}, "e":1}'
OK
127.0.0.1:6379> JSON.OBJLEN k1 .a
(integer) 0
127.0.0.1:6379> JSON.OBJLEN k1 .a.*
(error) NONEXISTENT JSON path does not exist
127.0.0.1:6379> JSON.OBJLEN k1 .b
(integer) 1
127.0.0.1:6379> JSON.OBJLEN k1 .b.*
(error) WRONGTYPE JSON element is not an object
127.0.0.1:6379> JSON.OBJLEN k1 .c
(integer) 2
127.0.0.1:6379> JSON.OBJLEN k1 .c.*
(error) WRONGTYPE JSON element is not an object
127.0.0.1:6379> JSON.OBJLEN k1 .d
(integer) 3
127.0.0.1:6379> JSON.OBJLEN k1 .d.*
(integer) 2
127.0.0.1:6379> JSON.OBJLEN k1 .*
(integer) 0
```

## JSON.OBJKEYS

Mendapat nama kunci pada nilai objek di jalur.

Sintaks

```
JSON.OBJKEYS <key> [path]
```

- kunci (wajib) – Kunci Redis dari jenis dokumen JSON.
- jalur (opsional) – Sebuah jalur JSON. Diatur secara default ke root jika tidak disediakan.

Nilai yang ditampilkan

Jika jalur adalah sintaksis yang ditingkatkan:

- Susunan string massal. Setiap elemen adalah susunan kunci dalam objek yang cocok.

- Jika nilai bukan objek, nilai yang akan dikembalikan adalah nilai yang kosong.
- Kosong jika kunci dokumen tidak ada.

Jika jalur adalah sintaksis terbatas:

- Susunan string massal. Setiap elemen adalah nama kunci dalam objek.
- Jika memilih beberapa objek, perintah mengembalikan kunci objek pertama.
- Kesalahan `WRONGTYPE` jika nilai di jalur bukan objek.
- Kesalahan `WRONGTYPE` jika jalur tidak ada.
- Kosong jika kunci dokumen tidak ada.

Contoh

Sintaksis jalur yang ditingkatkan:

```
127.0.0.1:6379> JSON.SET k1 $ '{"a":{}, "b":{"a":"a"}, "c":{"a":"a", "b":"bb"}, "d":
{"a":1, "b":"b", "c":{"a":3,"b":4}}, "e":1}'
OK
127.0.0.1:6379> JSON.OBJKEYS k1 $.*
1) (empty array)
2) 1) "a"
3) 1) "a"
   2) "b"
4) 1) "a"
   2) "b"
   3) "c"
5) (empty array)
127.0.0.1:6379> JSON.OBJKEYS k1 $.d
1) 1) "a"
   2) "b"
   3) "c"
```

Sintaksis jalur terbatas:

```
127.0.0.1:6379> JSON.SET k1 $ '{"a":{}, "b":{"a":"a"}, "c":{"a":"a", "b":"bb"}, "d":
{"a":1, "b":"b", "c":{"a":3,"b":4}}, "e":1}'
OK
```

```
127.0.0.1:6379> JSON.OBJKEYS k1 .*
1) "a"
127.0.0.1:6379> JSON.OBJKEYS k1 .d
1) "a"
2) "b"
3) "c"
```

## JSON.RESP

Menampilkan nilai JSON di jalur yang diberikan dalam Protokol Serialisasi Redis (RESP). Jika nilainya adalah kontainer, responsnya adalah array RESP atau array bersarang.

- JSON kosong dipetakan ke String Massal Kosong RESP.
- Nilai JSON Boolean dipetakan ke masing-masing String Sederhana RESP.
- Angka integer dipetakan ke Integer RESP.
- Nomor floating point ganda IEEE 64-bit dipetakan ke String Massal RESP.
- String JSON dipetakan ke String Massal RESP.
- Array JSON direpresentasikan sebagai Array RESP, dengan elemen pertama adalah string sederhana [, diikuti oleh elemen array.
- Objek JSON direpresentasikan sebagai Array RESP, dengan elemen pertama adalah string sederhana {, diikuti oleh pasangan kunci-nilai, yang masing-masing adalah string massal RESP.

### Sintaks

```
JSON.RESP <key> [path]
```

- kunci (wajib) – Kunci Redis dari jenis dokumen JSON.
- jalur (opsional) – Sebuah jalur JSON. Diatur secara default ke root jika tidak disediakan.

### Nilai yang ditampilkan

Jika jalur adalah sintaksis yang ditingkatkan:

- Array dari array. Setiap elemen array merepresentasikan bentuk RESP dari nilai pada satu jalur.
- Array kosong jika kunci dokumen tidak ada.

Jika jalur adalah sintaksis terbatas:

- Array yang merepresentasikan format RESP dari nilai pada jalur.
- Kosong jika kunci dokumen tidak ada.

Contoh

Sintaksis jalur yang ditingkatkan:

```
127.0.0.1:6379> JSON.SET k1 .
 '{"firstName":"John","lastName":"Smith","age":27,"weight":135.25,"isAlive":true,"address":
{"street":"21 2nd Street","city":"New
 York","state":"NY","zipcode":"10021-3100"},"phoneNumbers":
[{"type":"home","number":"212 555-1234"},{"type":"office","number":"646
 555-4567"}],"children":[],"spouse":null}'
OK

127.0.0.1:6379> JSON.RESP k1 $.address
1) 1) {
   2) 1) "street"
      2) "21 2nd Street"
   3) 1) "city"
      2) "New York"
   4) 1) "state"
      2) "NY"
   5) 1) "zipcode"
      2) "10021-3100"

127.0.0.1:6379> JSON.RESP k1 $.address.*
1) "21 2nd Street"
2) "New York"
3) "NY"
4) "10021-3100"

127.0.0.1:6379> JSON.RESP k1 $.phoneNumbers
1) 1) [
   2) 1) {
      2) 1) "type"
         2) "home"
      3) 1) "number"
         2) "555 555-1234"
   3) 1) {
```

```
      2) 1) "type"
         2) "office"
      3) 1) "number"
         2) "555 555-4567"

127.0.0.1:6379> JSON.RESP k1 $.phoneNumbers[*]
1) 1) {
   2) 1) "type"
      2) "home"
   3) 1) "number"
      2) "212 555-1234"
2) 1) {
   2) 1) "type"
      2) "office"
   3) 1) "number"
      2) "555 555-4567"
```

Sintaksis jalur terbatas:

```
127.0.0.1:6379> JSON.SET k1 .
 '{"firstName":"John","lastName":"Smith","age":27,"weight":135.25,"isAlive":true,"address":
{"street":"21 2nd Street","city":"New
 York","state":"NY","zipcode":"10021-3100"},"phoneNumbers":
[{"type":"home","number":"212 555-1234"},{"type":"office","number":"646
 555-4567"}],"children":[],"spouse":null}'
OK

127.0.0.1:6379> JSON.RESP k1 .address
1) {
2) 1) "street"
   2) "21 2nd Street"
3) 1) "city"
   2) "New York"
4) 1) "state"
   2) "NY"
5) 1) "zipcode"
   2) "10021-3100"

127.0.0.1:6379> JSON.RESP k1
 1) {
 2) 1) "firstName"
    2) "John"
```

```
 3) 1) "lastName"
    2) "Smith"
 4) 1) "age"
    2) (integer) 27
 5) 1) "weight"
    2) "135.25"
 6) 1) "isAlive"
    2) true
 7) 1) "address"
    2) 1) {
       2) 1) "street"
          2) "21 2nd Street"
       3) 1) "city"
          2) "New York"
       4) 1) "state"
          2) "NY"
       5) 1) "zipcode"
          2) "10021-3100"
 8) 1) "phoneNumbers"
    2) 1) [
       2) 1) {
          2) 1) "type"
             2) "home"
          3) 1) "number"
             2) "212 555-1234"
       3) 1) {
          2) 1) "type"
             2) "office"
          3) 1) "number"
             2) "555 555-4567"
 9) 1) "children"
    2) 1) [
10) 1) "spouse"
    2) (nil)
```

## JSON.SET

Menetapkan nilai JSON di jalur.

Jika jalur memanggil anggota objek:

- Jika elemen induk tidak ada, perintah mengembalikan kesalahan NONEXISTENT.

- Jika elemen induk tidak ada tapi bukan objek, perintah mengembalikan ERROR.
- Jika elemen induk ada dan merupakan objek:
  - Jika anggota tidak ada, anggota baru akan ditambahkan ke objek induk jika dan hanya jika objek induk adalah turunan terakhir di jalur. Jika tidak, perintah mengembalikan kesalahan NONEXISTENT.
  - Jika anggota ada, nilainya akan diganti dengan nilai JSON.

Jika jalur memanggil indeks susunan:

- Jika elemen induk tidak ada, perintah mengembalikan kesalahan NONEXISTENT.
- Jika elemen induk ada tetapi bukan susunan, perintah mengembalikan ERROR.
- Jika elemen induk ada tetapi indeks di luar batas, perintah mengembalikan kesalahan OUTOFBOUNDARIES.
- Jika elemen induk ada dan indeks valid, elemen akan diganti dengan nilai JSON baru.

Jika jalur memanggil objek atau susunan, nilai (objek atau susunan) akan digantikan oleh nilai JSON baru.

Sintaks

```
JSON.SET <key> <path> <json> [NX | XX]
```

[NX | XX] Di mana Anda dapat memiliki 0 atau 1 pengidentifikasi [NX | XX].

- kunci (wajib) – Kunci Redis dari jenis dokumen JSON.
- jalur (wajib) – Sebuah jalur JSON. Untuk kunci Redis baru, jalur JSON harus berupa root “.”.
- NX (opsional) - Jika jalurnya adalah root, atur nilainya hanya jika kunci Redis tidak ada. Artinya, masukkan dokumen baru. Jika jalur bukan root, atur nilainya hanya jika jalur tidak ada. Artinya, masukkan nilai ke dalam dokumen.
- XX (opsional) - Jika jalurnya adalah root, atur nilainya hanya jika kunci Redis ada. Artinya, ganti dokumen yang ada. Jika jalur bukan root, atur nilainya hanya jika jalur ada. Artinya, perbarui nilai yang ada.

Mengembalikan

- String sederhana 'OK' pada berhasil.
- Kosong jika kondisi NX atau XX tidak terpenuhi.

Contoh

Sintaksis jalur yang ditingkatkan:

```
127.0.0.1:6379> JSON.SET k1 . '{"a":{"a":1, "b":2, "c":3}}'
OK
127.0.0.1:6379> JSON.SET k1 $.a.* '0'
OK
127.0.0.1:6379> JSON.GET k1
"{\"a\":{\"a\":0,\"b\":0,\"c\":0}}"

127.0.0.1:6379> JSON.SET k2 . '{"a": [1,2,3,4,5]}'
OK
127.0.0.1:6379> JSON.SET k2 $.a[*] '0'
OK
127.0.0.1:6379> JSON.GET k2
"{\"a\":[0,0,0,0,0]}"
```

Sintaksis jalur terbatas:

```
127.0.0.1:6379> JSON.SET k1 . '{"c":{"a":1, "b":2}, "e": [1,2,3,4,5]}'
OK
127.0.0.1:6379> JSON.SET k1 .c.a '0'
OK
127.0.0.1:6379> JSON.GET k1
"{\"c\":{\"a\":0,\"b\":2},\"e\":[1,2,3,4,5]}"
127.0.0.1:6379> JSON.SET k1 .e[-1] '0'
OK
127.0.0.1:6379> JSON.GET k1
"{\"c\":{\"a\":0,\"b\":2},\"e\":[1,2,3,4,0]}"
127.0.0.1:6379> JSON.SET k1 .e[5] '0'
(error) OUTOFBOUNDARIES Array index is out of bounds
```

## JSON.STRAPPEND

Menambahkan string ke string JSON di jalur.

### Sintaks

```
JSON.STRAPPEND <key> [path] <json_string>
```

- kunci (wajib) – Kunci Redis dari jenis dokumen JSON.
- jalur (opsional) – Sebuah jalur JSON. Default ke root jika tidak disediakan.
- json_string (wajib) - Representasi JSON dari string. Perhatikan bahwa string JSON harus dikutip. Misalnya: ' "contoh string" '.

### Mengembalikan

Jika jalur adalah sintaksis yang ditingkatkan:

- Susunan integer yang mewakili panjang baru susunan di setiap jalur.
- Jika nilai di jalur bukan string, nilai yang akan dikembalikan adalah kosong.
- Kesalahan `SYNTAXERR` jika salah satu argumen input json bukan string JSON yang valid.
- Kesalahan `NONEXISTENT` jika jalur tidak ada.

Jika jalur adalah sintaksis terbatas:

- Integer, panjang baru susunan.
- Jika memilih beberapa nilai string, perintah mengembalikan panjang baru dari susunan yang terakhir diperbarui.
- Kesalahan `WRONGTYPE` jika nilai di jalur bukan string.
- Kesalahan `WRONGTYPE` jika salah satu argumen input json bukan string JSON yang valid.
- Kesalahan `NONEXISTENT` jika jalur tidak ada.

### Contoh

Sintaksis jalur yang ditingkatkan:

```
127.0.0.1:6379> JSON.SET k1 $ '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a",
 "b":"bb"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.STRAPPEND k1 $.a.a '"a"'
1) (integer) 2
```

```
127.0.0.1:6379> JSON.STRAPPEND k1 $.a.* '"a"'
1) (integer) 3
127.0.0.1:6379> JSON.STRAPPEND k1 $.b.* '"a"'
1) (integer) 2
2) (nil)
127.0.0.1:6379> JSON.STRAPPEND k1 $.c.* '"a"'
1) (integer) 2
2) (integer) 3
127.0.0.1:6379> JSON.STRAPPEND k1 $.c.b '"a"'
1) (integer) 4
127.0.0.1:6379> JSON.STRAPPEND k1 $.d.* '"a"'
1) (nil)
2) (integer) 2
3) (nil)
```

Sintaksis jalur terbatas:

```
127.0.0.1:6379> JSON.SET k1 . '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a",
 "b":"bb"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.STRAPPEND k1 .a.a '"a"'
(integer) 2
127.0.0.1:6379> JSON.STRAPPEND k1 .a.* '"a"'
(integer) 3
127.0.0.1:6379> JSON.STRAPPEND k1 .b.* '"a"'
(integer) 2
127.0.0.1:6379> JSON.STRAPPEND k1 .c.* '"a"'
(integer) 3
127.0.0.1:6379> JSON.STRAPPEND k1 .c.b '"a"'
(integer) 4
127.0.0.1:6379> JSON.STRAPPEND k1 .d.* '"a"'
(integer) 2
```

## JSON.STRLEN

Mendapatkan panjang nilai string JSON di jalur.

Sintaks

```
JSON.STRLEN <key> [path]
```

- kunci (wajib) – Kunci Redis dari jenis dokumen JSON.
- jalur (opsional) – Sebuah jalur JSON. Diatur secara default ke root jika tidak disediakan.

Nilai yang ditampilkan

Jika jalur adalah sintaksis yang ditingkatkan:

- Susunan integer yang mewakili panjang nilai susunan di tiap jalur.
- Jika nilai bukan string, nilai yang akan dikembalikan adalah kosong.
- Kosong jika kunci dokumen tidak ada.

Jika jalur adalah sintaksis terbatas:

- Integer, panjang string.
- Jika memilih beberapa nilai string, perintah mengembalikan panjang string pertama.
- Kesalahan `WRONGTYPE` jika nilai di jalur bukan string.
- Kesalahan `NONEXISTENT` jika jalur tidak ada.
- Kosong jika kunci dokumen tidak ada.

Contoh

Sintaksis jalur yang ditingkatkan:

```
127.0.0.1:6379> JSON.SET k1 $ '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a",
 "b":"bb"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.STRLEN k1 $.a.a
1) (integer) 1
127.0.0.1:6379> JSON.STRLEN k1 $.a.*
1) (integer) 1
127.0.0.1:6379> JSON.STRLEN k1 $.c.*
1) (integer) 1
2) (integer) 2
127.0.0.1:6379> JSON.STRLEN k1 $.c.b
1) (integer) 2
127.0.0.1:6379> JSON.STRLEN k1 $.d.*
1) (nil)
2) (integer) 1
```

```
3) (nil)
```

Sintaksis jalur terbatas:

```
127.0.0.1:6379> JSON.SET k1 $ '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a",
 "b":"bb"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.STRLEN k1 .a.a
(integer) 1
127.0.0.1:6379> JSON.STRLEN k1 .a.*
(integer) 1
127.0.0.1:6379> JSON.STRLEN k1 .c.*
(integer) 1
127.0.0.1:6379> JSON.STRLEN k1 .c.b
(integer) 2
127.0.0.1:6379> JSON.STRLEN k1 .d.*
(integer) 1
```

## JSON.TOGGLE

Mengalihkan nilai Boolean antara true dan false di jalur.

Sintaks

```
JSON.TOGGLE <key> [path]
```

- kunci (wajib) – Kunci Redis dari jenis dokumen JSON.
- jalur (opsional) – Sebuah jalur JSON. Diatur secara default ke root jika tidak disediakan.

Nilai yang ditampilkan

Jika jalur adalah sintaksis yang ditingkatkan:

- Susunan integer (0 - false, 1 - true) yang mewakili nilai Boolean yang dihasilkan di setiap jalur.
- Jika nilai bukan nilai Boolean, nilai yang akan dikembalikan adalah kosong.
- `NONEXISTENT` jika kunci dokumen tidak ada.

Jika jalur adalah sintaksis terbatas:

- String (“true” /"false”) yang mewakili nilai Boolean yang dihasilkan.
- `NONEXISTENT` jika kunci dokumen tidak ada.
- Kesalahan `WRONGTYPE` jika nilai di jalur bukan nilai Boolean.

### Contoh

Sintaksis jalur yang ditingkatkan:

```
127.0.0.1:6379> JSON.SET k1 . '{"a":true, "b":false, "c":1, "d":null, "e":"foo", "f":
[], "g":{}}'
OK
127.0.0.1:6379> JSON.TOGGLE k1 $.*
1) (integer) 0
2) (integer) 1
3) (nil)
4) (nil)
5) (nil)
6) (nil)
7) (nil)
127.0.0.1:6379> JSON.TOGGLE k1 $.*
1) (integer) 1
2) (integer) 0
3) (nil)
4) (nil)
5) (nil)
6) (nil)
7) (nil)
```

Sintaksis jalur terbatas:

```
127.0.0.1:6379> JSON.SET k1 . true
OK
127.0.0.1:6379> JSON.TOGGLE k1
"false"
127.0.0.1:6379> JSON.TOGGLE k1
"true"

127.0.0.1:6379> JSON.SET k2 . '{"isAvailable": false}'
OK
127.0.0.1:6379> JSON.TOGGLE k2 .isAvailable
```

```
"true"
127.0.0.1:6379> JSON.TOGGLE k2 .isAvailable
"false"
```

## JSON.TYPE

Melaporkan jenis nilai di jalur yang diberikan.

Sintaks

```
JSON.TYPE <key> [path]
```

- kunci (wajib) – Kunci Redis dari jenis dokumen JSON.
- jalur (opsional) – Sebuah jalur JSON. Diatur secara default ke root jika tidak disediakan.

Nilai yang ditampilkan

Jika jalur adalah sintaksis yang ditingkatkan:

- Susunan string yang mewakili nilai yang muncul di setiap jalur. Jenisnya adalah salah satu dari {"null", “boolean”, “string”, “number”, “integer”, “object” dan “array"}.
- Jika jalur tidak ada, nilai kembalinya adalah kosong.
- Susunan kosong jika kunci dokumen tidak ada.

Jika jalur adalah sintaksis terbatas:

- String, jenis nilai
- Kosong jika kunci dokumen tidak ada.
- Kosong jika jalur JSON tidak valid atau tidak ada.

Contoh

Sintaksis jalur yang ditingkatkan:

```
127.0.0.1:6379> JSON.SET k1 . '[1, 2.3, "foo", true, null, {}, []]'
OK
```

```
127.0.0.1:6379> JSON.TYPE k1 $[*]
1) integer
2) number
3) string
4) boolean
5) null
6) object
7) array
```

Sintaksis jalur terbatas:

```
127.0.0.1:6379> JSON.SET k1 .
 '{"firstName":"John","lastName":"Smith","age":27,"weight":135.25,"isAlive":true,"address":
{"street":"21 2nd Street","city":"New
 York","state":"NY","zipcode":"10021-3100"},"phoneNumbers":
[{"type":"home","number":"212 555-1234"},{"type":"office","number":"646
 555-4567"}],"children":[],"spouse":null}'
OK
127.0.0.1:6379> JSON.TYPE k1
object
127.0.0.1:6379> JSON.TYPE k1 .children
array
127.0.0.1:6379> JSON.TYPE k1 .firstName
string
127.0.0.1:6379> JSON.TYPE k1 .age
integer
127.0.0.1:6379> JSON.TYPE k1 .weight
number
127.0.0.1:6379> JSON.TYPE k1 .isAlive
boolean
127.0.0.1:6379> JSON.TYPE k1 .spouse
null
```

# Menandai sumber daya ElastiCache Anda

Untuk membantu Anda mengelola klaster dan sumber daya ElastiCache lainnya, Anda dapat menetapkan metadata Anda sendiri ke setiap sumber daya dalam bentuk tanda. Dengan tanda, Anda dapat mengategorikan sumber daya AWS Anda dengan berbagai cara, misalnya, berdasarkan tujuan, pemilik, atau lingkungan. Hal ini berguna ketika Anda memiliki banyak sumber daya dengan

tipe yang sama—Anda dapat dengan cepat mengidentifikasi sumber daya tertentu berdasarkan tanda yang telah Anda tetapkan. Topik ini memberikan penjelasan tentang tanda dan menunjukkan cara membuatnya.

> **Warning**
>
> Sebagai praktik terbaik, sebaiknya Anda tidak menyertakan data sensitif ke dalam tanda.

## Dasar-dasar tanda

Tanda merupakan label yang Anda tetapkan ke sumber daya AWS. Setiap tanda terdiri dari kunci dan nilai opsional, yang keduanya Anda tentukan. Tanda memungkinkan Anda mengategorikan sumber daya AWS Anda dengan berbagai cara, misalnya, berdasarkan tujuan atau pemilik. Misalnya, Anda dapat menentukan serangkaian tanda untuk klaster ElastiCache akun Anda yang dapat membantu melacak setiap pemilik dan grup pengguna dari setiap instans.

Sebaiknya rancang serangkaian kunci tanda yang memenuhi kebutuhan setiap tipe sumber daya. Penggunaan set kunci tanda yang konsisten akan memudahkan pengelolaan sumber daya Anda. Anda dapat mencari dan memfilter sumber daya berdasarkan tanda yang Anda tambahkan. Untuk informasi selengkapnya tentang cara mengimplementasikan strategi penandaan sumber daya yang efektif, lihat Laporan resmi AWS Praktik Terbaik Penandaan.

Tanda tidak memiliki makna semantik pada ElastiCache dan ditafsirkan sebagai string karakter. Selain itu, tanda tidak secara otomatis ditetapkan ke sumber daya Anda. Anda dapat mengedit kunci dan nilai tanda, serta menghapus tanda dari sumber daya kapan saja. Anda dapat menetapkan nilai tanda ke `null`. Jika Anda menambahkan tanda yang memiliki kunci yang sama dengan tanda yang telah ada di sumber daya tersebut, nilai yang baru akan menimpa nilai yang lama. Jika Anda menghapus sumber daya, semua tanda untuk sumber daya tersebut juga akan dihapus. Selain itu, jika Anda menambahkan atau menghapus tanda di grup replikasi, semua simpul di dalam grup replikasi itu juga akan mendapat penambahan atau penghapusan tanda yang sama.

Anda dapat bekerja dengan tanda menggunakan AWS Management Console, AWS CLI, dan API ElastiCache.

Jika menggunakan IAM, Anda dapat mengontrol pengguna mana di akun AWS Anda yang memiliki izin untuk membuat, mengedit, atau menghapus tanda. Untuk informasi selengkapnya, lihat Izin tingkat sumber daya.

## Sumber daya yang dapat Anda tandai

Anda dapat menandai sebagian besar sumber daya ElastiCache yang sudah ada dalam akun Anda. Tabel di bawah ini mencantumkan sumber daya yang mendukung penandaan. Jika Anda menggunakan AWS Management Console, Anda dapat menerapkan tanda ke sumber daya dengan menggunakan Editor Tanda. Beberapa layar sumber daya memungkinkan Anda menentukan tanda untuk sebuah sumber daya saat sumber daya tersebut dibuat; misalnya, tanda dengan kunci Nama dan nilai yang Anda tentukan. Dalam kebanyakan kasus, konsol menerapkan tanda segera setelah sumber daya dibuat (alih-alih selama pembuatan sumber daya). Konsol dapat mengatur sumber daya sesuai dengan tanda Nama, tetapi tanda ini tidak memiliki makna semantik pada layanan ElastiCache.

Selain itu, beberapa tindakan pembuatan sumber daya memungkinkan Anda menentukan tanda untuk sumber daya saat sumber daya tersebut dibuat. Jika tanda tidak dapat diterapkan selama pembuatan sumber daya, kami akan mengembalikan proses pembuatan sumber daya. Hal ini untuk memastikan bahwa sumber daya dibuat dengan tanda atau tidak akan dibuat sama sekali, dan tidak akan ada sumber daya yang dibiarkan tidak bertanda. Dengan menandai sumber daya saat pembuatan, Anda tidak perlu menjalankan skrip penandaan kustom setelah pembuatan sumber daya.

Jika Anda menggunakan API Amazon ElastiCache, CLI AWS, atau SDK AWS, Anda dapat menggunakan parameter `Tags` pada tindakan ElastiCache API yang relevan untuk menerapkan tanda. File tersebut adalah:

- `CreateServerlessCache`
- `CreateCacheCluster`
- `CreateReplicationGroup`
- `CopyServerlessCacheSnapshot`
- `CopySnapshot`
- `CreateCacheParameterGroup`
- `CreateCacheSecurityGroup`
- `CreateCacheSubnetGroup`
- `CreateServerlessCacheSnapshot`
- `CreateSnapshot`
- `CreateUserGroup`

- `CreateUser`
- `PurchaseReservedCacheNodesOffering`

Tabel berikut menjelaskan sumber daya ElastiCache yang dapat ditandai, dan sumber daya yang dapat ditandai saat pembuatan dengan menggunakan API ElastiCache, CLI AWS, atau SDK AWS.

Dukungan penandaan untuk sumber daya ElastiCache

| Mendukung tanda | Mendukung penandaan saat pembuatan |
|---|---|
| Ya | Ya |
| Ya | Ya |
| Ya | Ya |
| Ya | Ya |
| Ya | Ya |
| Ya | Ya |
| Ya | Ya |
| Ya | Ya |
| Ya | Ya |
| Ya | Ya |

| Mendukung tanda | Mendukung penandaan saat pembuatan |
| --- | --- |
| Ya | Ya |

> **Note**
>
> Anda tidak dapat menandai Penyimpanan Data Global.

Anda dapat menerapkan izin tingkat sumber daya berbasis tanda dalam kebijakan IAM pada tindakan API ElastiCache yang mendukung penandaan saat pembuatan guna mengimplementasikan kontrol terperinci atas pengguna dan grup yang dapat menandai sumber daya saat pembuatan. Sumber daya Anda diamankan secara tepat sejak pembuatan—tanda yang diterapkan segera ke sumber daya Anda. Oleh karena itu, izin tingkat sumber daya berbasis tanda apa pun yang mengontrol penggunaan sumber daya akan langsung diterapkan. Sumber daya Anda dapat dilacak dan dilaporkan dengan lebih akurat. Anda dapat menerapkan penggunaan penandaan pada sumber daya baru serta mengontrol kunci dan nilai tanda mana yang ditetapkan pada sumber daya Anda.

Untuk informasi selengkapnya, lihat Contoh penandaan sumber daya.

Untuk informasi selanjutnya tentang penandaan sumber daya Anda untuk penagihan, lihat Memantau biaya dengan tanda alokasi biaya.

### Memberi tag pada cache dan snapshot

Aturan berikut berlaku untuk penandaan sebagai bagian dari operasi permintaan:

- CreateReplicationGroup:
  - Jika parameter `--primary-cluster-id` dan `--tags` termasuk dalam permintaan, tanda permintaan akan ditambahkan ke grup replikasi dan disebarkan ke semua klaster cache di grup replikasi. Jika klaster cache primer sebelumnya sudah memiliki tanda, tanda ini akan ditimpa dengan tanda permintaan agar tanda menjadi konsisten di semua simpul.

    Jika tidak ada tanda permintaan, tanda klaster cache primer akan ditambahkan ke grup replikasi dan disebarkan ke semua klaster cache.
  - Jika `--snapshot-name` atau `--serverless-cache-snapshot-name` disediakan:

Jika tag disertakan dalam permintaan, grup replikasi hanya akan ditandai dengan tag tersebut. Jika tidak ada tag yang disertakan dalam permintaan, tag snapshot akan ditambahkan ke grup replikasi.

- Jika `--global-replication-group-id` disediakan:

  Jika tag disertakan dalam permintaan, tag permintaan akan ditambahkan ke grup replikasi dan disebarkan ke semua klaster cache.

- CreateCacheCluster :
  - Jika `--replication-group-id` disediakan:

    Jika tanda disertakan dalam permintaan, klaster cache akan ditandai hanya dengan tanda tersebut. Jika tidak ada tanda yang disertakan dalam permintaan, klaster cache akan mewarisi tanda grup replikasi, bukan tanda dari klaster cache primer.
  - Jika `--snapshot-name` disediakan:

    Jika tanda disertakan dalam permintaan, klaster cache akan ditandai hanya dengan tanda tersebut. Jika tidak ada tanda yang disertakan dalam permintaan, tanda snapshot akan ditambahkan ke klaster cache.
- CreateServerlessCache:
  - Jika tag disertakan dalam permintaan, hanya tag permintaan yang akan ditambahkan ke cache nirserver.
- CreateSnapshot :
  - Jika `--replication-group-id` disediakan:

    Jika tanda disertakan dalam permintaan, hanya tanda permintaan yang akan ditambahkan ke snapshot. Jika tidak ada tanda yang disertakan dalam permintaan, tanda grup replikasi akan ditambahkan ke snapshot.
  - Jika `--cache-cluster-id` disediakan:

    Jika tanda disertakan dalam permintaan, hanya tanda permintaan yang akan ditambahkan ke snapshot. Jika tidak ada tanda yang disertakan dalam permintaan, tanda klaster cache akan ditambahkan ke snapshot.
  - Untuk snapshot otomatis:

    Tanda akan disebarkan dari tanda grup replikasi.
- CreateServerlessCacheSnapshot:

  - Jika tag disertakan dalam permintaan, hanya tag permintaan yang akan ditambahkan ke snapshot cache nirserver.
- CopySnapshot :
  - Jika tanda disertakan dalam permintaan, hanya tanda permintaan yang akan ditambahkan ke snapshot. Jika tidak ada tanda yang disertakan dalam permintaan, tanda snapshot sumber akan ditambahkan ke snapshot salinan.
- CopyServerlessCacheSnapshot:
  - Jika tag disertakan dalam permintaan, hanya tag permintaan yang akan ditambahkan ke snapshot cache nirserver.
- AddTagsToResource dan RemoveTagsFromResource :
  - Tanda akan ditambahkan/dihapus dari grup replikasi dan tindakan akan disebarkan ke semua klaster dalam grup replikasi.

> **Note**
>
> AddTagsToResource dan RemoveTagsFromResource tidak dapat digunakan untuk parameter dan grup keamanan default.

- IncreaseReplicaCount dan ModifyReplicationGroupShardConfiguration:
  - Semua klaster baru yang ditambahkan ke grup replikasi akan memiliki tag yang sama dengan grup replikasi.

## Batasan tanda

Batasan dasar berikut berlaku untuk tanda:

- Jumlah maksimum tanda per sumber daya – 50
- Untuk setiap sumber daya, setiap kunci tanda harus unik, dan setiap kunci tanda hanya dapat memiliki satu nilai.
- Panjang kunci maksimum – 128 karakter Unicode dalam UTF-8.
- Panjang nilai maksimum – 256 karakter Unicode dalam UTF-8.
- Meskipun ElastiCache memungkinkan karakter apa pun dalam tandanya, layanan lain mungkin lebih terbatas. Karakter yang diizinkan di semua layanan adalah huruf, angka, dan spasi yang dapat direpresentasikan dalam UTF-8, serta karakter berikut: + - = . _ : / @
- Kunci dan nilai tanda peka huruf besar dan kecil.

- Prefiks `aws:` disimpan untuk penggunaan AWS. Jika tanda memiliki kunci tanda dengan prefiks ini, Anda tidak dapat mengedit atau menghapus kunci atau nilai tanda tersebut. Tanda dengan prefiks `aws:` tidak dihitung terhadap tanda per batas sumber daya.

Anda tidak dapat mengakhiri, menghentikan, atau menghapus sumber daya berdasarkan tandanya saja; Anda harus menentukan pengidentifikasi sumber daya tersebut. Misalnya, untuk menghapus snapshot yang Anda tandai dengan tanda kunci yang disebut `DeleteMe`, Anda harus menggunakan tindakan `DeleteSnapshot` dengan pengidentifikasi sumber daya snapshot tersebut, seperti `snap-1234567890abcdef0`.

Untuk informasi selengkapnya tentang sumber daya ElastiCache yang dapat ditandai, lihat Sumber daya yang dapat Anda tandai.

## Contoh penandaan sumber daya

- Membuat cache nirserver menggunakan tanda

```
aws elasticache create-serverless-cache \
    --serverless-cache-name CacheName \
    --engine redis
    --tags Key="Cost Center", Value="1110001" Key="project",Value="XYZ"
```

- Menambahkan tanda ke cache nirserver

```
aws elasticache add-tags-to-resource \
--resource-name arn:aws:elasticache:us-east-1:111111222233:serverlesscache:my-cache \
--tags Key="project",Value="XYZ" Key="Elasticache",Value="Service"
```

- Menambahkan tanda ke Grup Replikasi.

```
aws elasticache add-tags-to-resource \
--resource-name arn:aws:elasticache:us-east-1:111111222233:replicationgroup:my-rg \
--tags Key="project",Value="XYZ" Key="Elasticache",Value="Service"
```

- Membuat Klaster Cache menggunakan tanda.

```
aws elasticache create-cache-cluster \
--cluster-id testing-tags \
--cluster-description cluster-test \
--cache-subnet-group-name test \
--cache-node-type cache.t2.micro \
```

```
--engine redis \
--tags Key="project",Value="XYZ" Key="Elasticache",Value="Service"
```

- Membuat snapshot nirserver dengan tag.

```
aws elasticache create-serverless-cache-snapshot \
--serverless-cache-name testing-tags \
--serverless-cache-snapshot-name bkp-testing-tags-scs \
--tags Key="work",Value="foo"
```

- Membuat Snapshot dengan tanda.

  Untuk kasus ini, jika Anda menambahkan tanda pada permintaan, bahkan jika grup replikasi berisi tanda, snapshot hanya akan menerima tanda permintaan.

```
aws elasticache create-snapshot \
--replication-group-id testing-tags \
--snapshot-name bkp-testing-tags-rg \
--tags Key="work",Value="foo"
```

## Contoh kebijakan kontrol akses Berbasis Tanda

1. Mengizinkan tindakan `AddTagsToResource` untuk klaster hanya jika klaster memiliki tanda Project=XYZ.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "elasticache:AddTagsToResource",
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Project": "XYZ"
                }
            }
        }
```

```
    ]
}
```

2. Mengizinkan tindakan `RemoveTagsFromResource` dari grup replikasi jika grup berisi tanda Project dan Service serta kunci yang berbeda dari Project dan Service.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "elasticache:RemoveTagsFromResource",
            "Resource": [
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Service": "Elasticache",
                    "aws:ResourceTag/Project": "XYZ"
                },
                "ForAnyValue:StringNotEqualsIgnoreCase": {
                    "aws:TagKeys": [
                        "Project",
                        "Service"
                    ]
                }
            }
        }
    ]
}
```

3. Mengizinkan `AddTagsToResource` untuk sumber daya apa pun hanya jika tanda berbeda dari Project dan Service.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "elasticache:AddTagsToResource",
            "Resource": [
                "arn:aws:elasticache:*:*:*:*"
            ],
```

```
            "Condition": {
                "ForAnyValue:StringNotEqualsIgnoreCase": {
                    "aws:TagKeys": [
                        "Service",
                        "Project"
                    ]
                }
            }
        }
    ]
}
```

4. Menolak tindakan `CreateReplicationGroup` jika permintaan memiliki `Tag Project=Foo`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "elasticache:CreateReplicationGroup",
            "Resource": [
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/Project": "Foo"
                }
            }
        }
    ]
}
```

5. Menolak tindakan `CopySnapshot` jika sumber snapshot memiliki tanda Project=XYZ dan tanda permintaan adalah Service=Elasticache.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "elasticache:CopySnapshot",
            "Resource": [
                "arn:aws:elasticache:*:*:snapshot:*"
```

```
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Project": "XYZ",
                    "aws:RequestTag/Service": "Elasticache"
                }
            }
        }
    ]
}
```

6. Menolak `CreateCacheCluster` tindakan jika tag permintaan `Project` hilang atau tidak sama dengan `Dev`, `QA` atau `Prod`.

```
{
    "Version": "2012-10-17",
    "Statement": [
          {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*",
                "arn:aws:elasticache:*:*:securitygroup:*",
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ]
        },
        {
            "Effect": "Deny",
            "Action": [
                "elasticache:CreateCacheCluster"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*"
            ],
            "Condition": {
                "Null": {
                    "aws:RequestTag/Project": "true"
                }
            }
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:AddTagsToResource"
            ],
            "Resource": "arn:aws:elasticache:*:*:cluster:*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/Project": [
                        "Dev",
                        "Prod",
                        "QA"
                    ]
                }
            }
        }
    ]
}
```

Untuk informasi terkait kunci syarat ini, lihat Menggunakan kunci syarat.

## Memantau biaya dengan tanda alokasi biaya

Ketika Anda menambahkan tanda alokasi biaya ke sumber daya Anda di Amazon ElastiCache, Anda dapat melacak biaya dengan mengelompokkan pengeluaran pada faktur Anda berdasarkan nilai tanda sumber daya.

Tanda alokasi biaya ElastiCache adalah pasangan kunci dan nilai yang Anda tentukan dan kaitkan dengan sumber daya ElastiCache. Kunci dan nilai peka terhadap huruf besar dan kecil. Anda dapat menggunakan kunci tanda untuk menentukan kategori, dan nilai tanda dapat berupa item dalam kategori tersebut. Misalnya, Anda dapat menentukan kunci tanda `CostCenter` dan nilai tanda `10010`, yang menunjukkan bahwa sumber daya tersebut ditetapkan ke pusat pembiayaan 10010. Anda juga dapat menggunakan tanda untuk menunjukkan bahwa sumber daya sedang digunakan untuk pengujian atau produksi dengan menggunakan kunci seperti `Environment` dan nilai seperti `test` atau `production`. Sebaiknya gunakan kumpulan kunci tanda yang konsisten untuk mempermudah pelacakan biaya yang terkait dengan sumber daya Anda.

Gunakan tanda alokasi biaya untuk mengatur tagihan AWS guna mencerminkan struktur biaya Anda sendiri. Agar dapat melakukannya, daftar untuk mendapatkan tagihan akun AWS Anda dengan

menyertakan nilai kunci tanda. Kemudian, untuk melihat biaya sumber daya gabungan, atur informasi penagihan Anda sesuai dengan sumber daya Anda dengan nilai kunci tanda yang sama. Misalnya, Anda dapat menandai beberapa sumber daya dengan nama aplikasi tertentu, kemudian mengatur informasi penagihan untuk melihat biaya total aplikasi tersebut pada beberapa layanan.

Anda juga dapat menggabungkan tanda untuk melacak biaya dengan tingkat detail yang lebih besar. Misalnya, untuk melacak biaya layanan Anda menurut wilayah, Anda dapat menggunakan kunci tanda `Service` dan `Region`. Di salah satu sumber daya Anda mungkin memiliki nilai `ElastiCache` dan `Asia Pacific (Singapore)`, serta di sumber daya lain Anda mempunyai nilai `ElastiCache` dan `Europe (Frankfurt)`. Anda kemudian dapat melihat total biaya ElastiCache Anda yang dikelompokkan menurut wilayah. Untuk informasi selengkapnya, lihat Menggunakan Tanda Alokasi Biaya di Panduan Pengguna AWS Billing.

Anda dapat menambahkan tanda alokasi biaya ElastiCache ke simpul Redis. Saat Anda menambah, menampilkan daftar, mengubah, menyalin, atau menghapus tanda, operasi tersebut hanya akan diterapkan ke simpul yang ditentukan.

Karakteristik tanda alokasi biaya ElastiCache

- Tanda alokasi biaya diterapkan ke sumber daya ElastiCache yang ditentukan dalam operasi CLI dan API sebagai ARN. Jenis sumber daya akan berupa "klaster".

  Contoh ARN: `arn:aws:elasticache:<region>:<customer-id>:<resource-type>:<resource-name>`

  Contoh arn: `arn:aws:elasticache:us-west-2:1234567890:cluster:my-cluster`
- Kunci tag adalah nama tanda yang wajib diisi. Nilai string kunci dapat terdiri dari 1 hingga 128 karakter Unicode dan tidak boleh diawali dengan `aws:`. String dapat berisi hanya kumpulan huruf Unicode, angka, spasi kosong, garis bawah (_), titik (.), titik dua (:), garis miring terbalik (\), tanda sama dengan (=), tanda plus (+), tanda hubung (-), atau tanda aroba (@).

- Nilai tanda adalah nilai tanda opsional. Nilai string dari nilai dapat terdiri dari 1 hingga 256 karakter Unicode dan tidak boleh diawali dengan `aws:`. String dapat berisi hanya kumpulan huruf Unicode, angka, spasi kosong, garis bawah ( _ ), titik ( . ), titik dua ( : ), garis miring terbalik ( \ ), tanda sama dengan ( = ), tanda plus ( + ), tanda hubung ( - ), atau tanda aroba ( @ ).

- Sumber daya ElastiCache dapat memiliki maksimum 50 tanda.

- Nilai tidak harus unik di dalam kumpulan tanda. Misalnya, Anda dapat memiliki kumpulan tanda dengan baik kunci `Service` dan `Application` yang memiliki nilai `ElastiCache`.

AWS tidak menerapkan makna semantik pada tanda Anda. Tanda ditafsirkan hanya sebagai string karakter. AWS tidak menetapkan secara otomatis tanda apa pun pada sumber daya ElastiCache.

## Mengelola tanda alokasi biaya Anda menggunakan AWS CLI

Anda dapat menggunakan AWS CLI untuk menambah, mengubah, atau menghapus tanda alokasi biaya.

Contoh arn: `arn:aws:elasticache:us-west-2:1234567890:cluster:my-cluster`

Tanda alokasi biaya diterapkan ke simpul ElastiCache for Redis. Simpul yang akan ditandai ditentukan menggunakan ARN (Amazon Resource Name).

Contoh arn: `arn:aws:elasticache:us-west-2:1234567890:cluster:my-cluster`

Topik

- Menampilkan daftar tanda menggunakan AWS CLI
- Menambah tanda menggunakan AWS CLI
- Mengubah tanda menggunakan AWS CLI
- Menghapus tanda menggunakan AWS CLI

### Menampilkan daftar tanda menggunakan AWS CLI

Anda dapat menggunakan AWS CLI untuk menampilkan daftar tanda di sumber daya ElastiCache yang ada dengan menggunakan operasi list-tags-for-resource.

Kode berikut menggunakan AWS CLI untuk menampilkan daftar tanda pada simpul Redis `my-cluster-001` di klaster `my-cluster` di wilayah us-west-2.

Untuk Linux, macOS, atau Unix:

```
aws elasticache list-tags-for-resource \
  --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster-001
```

Untuk Windows:

```
aws elasticache list-tags-for-resource ^
  --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster-001
```

Output dari operasi ini akan terlihat seperti berikut ini, daftar dari semua tanda pada sumber daya.

```
{
   "TagList": [
      {
         "Value": "10110",
         "Key": "CostCenter"
      },
      {
         "Value": "EC2",
         "Key": "Service"
      }
   ]
}
```

Jika tidak ada tanda pada sumber daya, output akan menjadi TagList kosong.

```
{
   "TagList": []
}
```

Untuk informasi selengkapnya, silakan lihat AWS CLI untuk ElastiCache list-tags-for-resource.

## Menambah tanda menggunakan AWS CLI

Anda dapat menggunakan AWS CLI untuk menampilkan daftar tanda di sumber daya ElastiCache yang ada dengan menggunakan operasi CLI add-tags-to-resource. Jika kunci tanda tidak ada di sumber daya, kunci dan nilai akan ditambahkan ke sumber daya. Jika kunci sudah ada di sumber daya, nilai yang terkait dengan kunci tersebut akan diperbarui ke nilai yang baru.

Kode berikut menggunakan AWS CLI untuk menambahkan kunci `Service` dan `Region` dengan nilai `elasticache` dan `us-west-2` masing-masing ke simpul `my-cluster-001` dalam klaster `my-cluster` di wilayah us-west-2.

Untuk Linux, macOS, atau Unix:

```
aws elasticache add-tags-to-resource \
```

```
 --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster-001 \
 --tags Key=Service,Value=elasticache \
        Key=Region,Value=us-west-2
```

Untuk Windows:

```
aws elasticache add-tags-to-resource ^
 --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster-001 ^
 --tags Key=Service,Value=elasticache ^
        Key=Region,Value=us-west-2
```

Output dari operasi ini akan terlihat seperti berikut ini, daftar dari semua tanda pada sumber daya mengikuti operasi tersebut.

```
{
   "TagList": [
      {
         "Value": "elasticache",
         "Key": "Service"
      },
      {
         "Value": "us-west-2",
         "Key": "Region"
      }
   ]
}
```

Untuk informasi selengkapnya, silakan lihat AWS CLI untuk ElastiCache add-tags-to-resource.

Anda juga dapat menggunakan AWS CLI untuk menambahkan tanda ke klaster ketika Anda membuat sebuah klaster baru dengan menggunakan operasi create-cache-cluster. Anda tidak dapat menambahkan tanda saat membuat klaster menggunakan konsol manajemen ElastiCache. Setelah klaster dibuat, Anda kemudian dapat menggunakan konsol untuk menambahkan tanda pada kluster tersebut.

## Mengubah tanda menggunakan AWS CLI

Anda dapat menggunakan AWS CLI untuk mengubah tanda pada simpul di klaster ElastiCache for Redis.

Untuk mengubah tanda:

- Gunakan add-tags-to-resource untuk menambahkan tanda dan nilai baru atau untuk mengubah nilai yang terkait dengan tanda yang ada.
- Gunakan remove-tags-from-resource untuk menghapus tanda tertentu dari sumber daya.

Output dari kedua operasi tersebut akan berupa daftar tanda dan nilai-nilainya di klaster yang ditentukan.

### Menghapus tanda menggunakan AWS CLI

Anda dapat menggunakan AWS CLI untuk menghapus tanda dari simpul yang ada di klaster ElastiCache for Redis dengan menggunakan operasi remove-tags-from-resource.

Kode berikut menggunakan AWS CLI untuk menghapus tanda dengan kunci `Service` dan `Region` dari simpul `my-cluster-001` di klaster `my-cluster` di wilayah us-west-2.

Untuk Linux, macOS, atau Unix:

```
aws elasticache remove-tags-from-resource \
 --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster-001 \
 --tag-keys PM Service
```

Untuk Windows:

```
aws elasticache remove-tags-from-resource ^
 --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster-001 ^
 --tag-keys PM Service
```

Output dari operasi ini akan terlihat seperti berikut ini, daftar dari semua tanda pada sumber daya mengikuti operasi tersebut.

```
{
   "TagList": []
}
```

Untuk informasi selengkapnya, silakan lihat AWS CLI untuk ElastiCache remove-tags-from-resource.

## Mengelola tanda alokasi biaya Anda menggunakan API ElastiCache

Anda dapat menggunakan API ElastiCache untuk menambah, mengubah, atau menghapus tanda alokasi biaya.

Tanda alokasi biaya diterapkan ke klaster ElastiCache for Memcached. Klaster yang akan ditandai ditentukan menggunakan ARN (Amazon Resource Name).

Contoh arn: `arn:aws:elasticache:us-west-2:1234567890:cluster:my-cluster`

Topik

- Menampilkan daftar tanda menggunakan API ElastiCache
- Menambahkan tanda menggunakan API ElastiCache
- Mengubah tanda menggunakan API ElastiCache
- Menghapus tanda menggunakan API ElastiCache

## Menampilkan daftar tanda menggunakan API ElastiCache

Anda dapat menggunakan API ElastiCache untuk menampilkan daftar tanda di sumber daya yang ada dengan menggunakan operasi ListTagsForResource.

Kode berikut menggunakan API ElastiCache untuk menampilkan daftar tanda di sumber daya `my-cluster-001` di wilayah us-west-2.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=ListTagsForResource
   &ResourceName=arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster-001
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Version=2015-02-02
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

## Menambahkan tanda menggunakan API ElastiCache

Anda dapat menggunakan API ElastiCache untuk menambahkan tanda ke klaster ElastiCache yang ada dengan menggunakan operasi AddTagsToResource. Jika kunci tanda tidak ada di sumber daya, kunci dan nilai akan ditambahkan ke sumber daya. Jika kunci sudah ada di sumber daya, nilai yang terkait dengan kunci tersebut akan diperbarui ke nilai yang baru.

Kode berikut menggunakan API ElastiCache untuk menambahkan kunci `Service` dan `Region` dengan nilai `elasticache` dan `us-west-2` masing-masing ke sumber daya `my-cluster-001` di wilayah us-west-2.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=AddTagsToResource
   &ResourceName=arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster-001
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Tags.member.1.Key=Service
   &Tags.member.1.Value=elasticache
   &Tags.member.2.Key=Region
   &Tags.member.2.Value=us-west-2
   &Version=2015-02-02
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

Untuk informasi selengkapnya, lihat AddTagsToResource di Referensi API Amazon ElastiCache.

### Mengubah tanda menggunakan API ElastiCache

Anda dapat menggunakan API ElastiCache untuk mengubah tanda di klaster ElastiCache.

Untuk mengubah nilai tanda:

- Gunakan operasi AddTagsToResource untuk menambahkan tanda dan nilai baru atau untuk mengubah nilai tanda yang ada.
- Gunakan RemoveTagsFromResource untuk menghapus tanda dari sumber daya.

Hasil dari kedua operasi tersebut akan berupa daftar tanda dan nilai-nilainya di sumber daya yang ditentukan.

Gunakan RemoveTagsFromResource untuk menghapus tanda dari sumber daya.

### Menghapus tanda menggunakan API ElastiCache

Anda dapat menggunakan API ElastiCache untuk menghapus tanda dari simpul ElastiCache for Redis yang ada dengan menggunakan operasi RemoveTagsFromResource.

Kode berikut menggunakan API ElastiCache untuk menghapus tanda dengan kunci `Service` dan `Region` dari simpul `my-cluster-001` di klaster `my-cluster` di wilayah us-west-2.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=RemoveTagsFromResource
   &ResourceName=arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster-001
```

```
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&TagKeys.member.1=Service
&TagKeys.member.2=Region
&Version=2015-02-02
&Timestamp=20150202T192317Z
&X-Amz-Credential=<credential>
```

# Menggunakan Lensa Amazon ElastiCache Well-Architected

Bagian ini menjelaskan Lensa Amazon ElastiCache Well-Architected, kumpulan prinsip desain dan panduan untuk merancang beban kerja ElastiCache well-architected.

- Lensa ElastiCache adalah aditif untuk AWS Kerangka Kerja Well-Architected.
- Setiap Pilar memiliki serangkaian pertanyaan untuk membantu memulai diskusi seputar Arsitektur ElastiCache.
  - Setiap pertanyaan memiliki sejumlah praktik terkemuka bersama dengan skor mereka untuk pelaporan.
    - Wajib - Diperlukan sebelum mulai ke prod (tidak memenuhinya menimbulkan risiko tinggi)
    - Terbaik - Kemungkinan status terbaik yang bisa dilakukan pelanggan
    - Bagus - Apa yang kami rekomendasikan untuk dimiliki pelanggan (tidak memenuhinya menimbulkan risiko sedang)
- Terminology Well-Architected
  - Komponen — Kode, konfigurasi, dan Sumber Daya AWS yang bersama-sama memenuhi persyaratan. Komponen berinteraksi dengan komponen lain, dan sering disamakan dengan layanan dalam arsitektur microservice.
  - Beban Kerja Seperangkat komponen yang bersama-sama memberikan nilai bisnis. Contoh beban kerja adalah situs web pemasaran, situs web e-commerce, back-end untuk aplikasi seluler, platform analitik, dll.

Topik

- Pilar Optimasi Biaya Lensa Amazon ElastiCache Well-Architected

# Pilar Keunggulan Operasional Lensa Well-Architected Amazon ElastiCache

Pilar keunggulan operasional berfokus untuk menjalankan dan memantau sistem guna memberikan nilai bisnis, dan terus meningkatkan proses dan prosedur. Topik utama yang dibahas meliputi mengotomatiskan perubahan, merespons peristiwa, dan mendefinisikan standar untuk mengelola operasi harian.

Topik

- OE 1: Bagaimana cara memahami dan menanggapi peringatan dan peristiwa yang dipicu oleh klaster ElastiCache Anda?
- OE 2: Kapan dan bagaimana Anda menskalakan klaster ElastiCache yang ada?
- OE 3: Bagaimana cara Anda mengelola sumber daya klaster ElastiCache dan menjaga klaster Anda tetap terbaru?
- OE 4: Bagaimana cara mengelola koneksi klien ke klaster ElastiCache Anda?
- OE 5: Bagaimana cara men-deploy Komponen ElastiCache untuk Beban Kerja?
- OE 6: Bagaimana cara melakukan perencanaan dan mitigasi terhadap kegagalan?
- OE 7: Bagaimana cara memecahkan masalah peristiwa mesin Redis?

## OE 1: Bagaimana cara memahami dan menanggapi peringatan dan peristiwa yang dipicu oleh klaster ElastiCache Anda?

Pengenalan tingkat pertanyaan: Saat Anda mengoperasikan klaster ElastiCache, Anda dapat menerima notifikasi dan peringatan secara opsional saat peristiwa tertentu terjadi. ElastiCache, secara default, mencatat log peristiwa yang berhubungan dengan sumber daya Anda, seperti failover, penggantian simpul, operasi penskalaan, pemeliharaan terjadwal, dan banyak lagi. Setiap peristiwa menyertakan tanggal dan waktu, nama sumber dan jenis sumber, serta deskripsi.

Manfaat tingkat pertanyaan: Dengan memahami dan mengelola alasan mendasar di balik peristiwa yang memicu peringatan yang dihasilkan oleh klaster, Anda dapat beroperasi secara lebih efektif dan merespons peristiwa dengan tepat.

- [Wajib] Tinjau peristiwa yang dihasilkan oleh ElastiCache pada konsol ElastiCache (setelah memilih wilayah Anda) atau menggunakan perintah describe-events Amazon Command Line

Interface (AWS CLI) dan API ElastiCache. Anda dapat mengonfigurasi ElastiCache untuk mengirim notifikasi untuk peristiwa klaster penting menggunakan Amazon Simple Notification Service (Amazon SNS). Penggunaan Amazon SNS dengan klaster Anda memungkinkan Anda mengambil tindakan pada peristiwa ElastiCache secara programatis.

- Ada dua kategori besar peristiwa: peristiwa terkini dan terjadwal. Daftar peristiwa terkini meliputi: pembuatan dan penghapusan sumber daya, operasi penskalaan, failover, boot ulang simpul, pembuatan snapshot, modifikasi parameter klaster, pembaruan sertifikat CA, peristiwa kegagalan (kegagalan penyediaan klaster - VPC atau ENI -, kegagalan penskalaan - ENI -, dan kegagalan snapshot). Daftar peristiwa terjadwal meliputi: simpul yang dijadwalkan untuk penggantian selama periode pemeliharaan dan penggantian simpul yang dijadwalkan ulang.
- Meskipun Anda mungkin tidak perlu segera bereaksi terhadap beberapa peristiwa ini, penting untuk melihat semua peristiwa kegagalan terlebih dahulu:
  - ElastiCache:AddCacheNodeFailed
  - ElastiCache:CacheClusterProvisioningFailed
  - ElastiCache:CacheClusterScalingFailed
  - ElastiCache:CacheNodesRebooted
  - ElastiCache:SnapshotFailed (hanya Redis)
- [Sumber Daya]:
  - Mengelola notifikasi ElastiCache Amazon SNS
  - Notifikasi Peristiwa dan Amazon SNS

- [Terbaik] Untuk mengotomatiskan respons terhadap peristiwa, manfaatkan kemampuan produk dan layanan AWS seperti Fungsi Lambda dan SNS. Ikuti praktik terbaik dengan membuat perubahan yang kecil, sering, dan dapat dikembalikan, sebagai kode untuk mengembangkan operasi Anda dari waktu ke waktu. Anda harus menggunakan metrik Amazon CloudWatch untuk memantau klaster.

  [Sumber Daya]: Pantau titik akhir replika baca Amazon ElastiCache for Redis (mode klaster dinonaktifkan) menggunakan Lambda AWS, Amazon Route 53, dan Amazon SNS untuk kasus penggunaan yang menggunakan Lambda dan SNS.

## OE 2: Kapan dan bagaimana Anda menskalakan klaster ElastiCache yang ada?

Pengenalan tingkat pertanyaan: Rightsizing klaster ElastiCache adalah tindakan penyeimbangan yang perlu dievaluasi setiap kali ada perubahan pada jenis beban kerja yang mendasarinya. Tujuan Anda adalah beroperasi dengan lingkungan yang telah di-rightsizing untuk beban kerja Anda.

Manfaat tingkat pertanyaan: Pemanfaatan sumber daya Anda yang berlebihan dapat mengakibatkan peningkatan latensi dan penurunan performa secara keseluruhan. Kurangnya pemanfaatan, di sisi lain, dapat mengakibatkan sumber daya yang disediakan secara berlebihan dengan optimisasi biaya yang tidak optimal. Dengan melakukan rightsizing lingkungan, Anda dapat mencapai keseimbangan antara efisiensi performa dan optimisasi biaya. Untuk mengatasi pemanfaatan sumber daya Anda yang kurang atau berlebihan, ElastiCache dapat menskalakan dua dimensi ke dalam. Anda dapat menskalakan secara vertikal dengan menambah atau mengurangi kapasitas simpul. Anda juga dapat menskalakan secara horizontal dengan menambahkan dan menghapus simpul.

- [Wajib] Pemanfaatan CPU dan jaringan yang berlebihan pada simpul primer harus diatasi dengan memindahkan dan mengalihkan operasi baca ke simpul replika. Gunakan simpul replika untuk operasi baca guna mengurangi pemanfaatan simpul primer. Hal ini dapat dikonfigurasi di pustaka klien Redis Anda dengan menghubungkan ke titik akhir pembaca ElastiCache untuk mode klaster dinonaktifkan, atau dengan menggunakan perintah Redis READONLY untuk mode klaster diaktifkan.

  [Sumber Daya]:

  - Menemukan titik akhir koneksi
  - Rightsizing Klaster
  - Perintah Redis READONLY
- [Wajib] Pantau pemanfaatan sumber daya klaster penting seperti CPU, memori, dan jaringan. Pemanfaatan sumber daya klaster yang spesifik ini perlu dilacak untuk menentukan keputusan Anda dalam penskalaan, dan jenis operasi penskalaan. Untuk klaster ElastiCache for Redis mode dinonaktifkan, simpul primer dan replika dapat diskalakan secara vertikal. Simpul replika juga dapat diskalakan secara horizontal dari 0 ke 5 simpul. Untuk mode klaster diaktifkan, hal yang sama berlaku dalam setiap serpihan klaster. Selain itu, Anda dapat menambah atau mengurangi jumlah serpihan.

  [Sumber Daya]:

  - Praktik terbaik pemantauan dengan Amazon ElastiCache for Redis menggunakan Amazon CloudWatch
  - Menskalakan Klaster ElastiCache for Redis
  - Menskalakan klaster ElastiCache for Memcached
- [Terbaik] Memantau tren dari waktu ke waktu dapat membantu Anda mendeteksi perubahan beban kerja yang akan luput dari perhatian jika dipantau pada titik waktu tertentu. Untuk mendeteksi tren jangka panjang, gunakan metrik CloudWatch untuk memindai rentang waktu yang lebih lama.

Pembelajaran dari pengamatan metrik CloudWatch dalam jangka waktu lama akan membentuk perkiraan Anda seputar pemanfaatan sumber daya klaster. Titik data dan metrik CloudWatch tersedia untuk rentang waktu hingga 455 hari.

[Sumber Daya]:

- Memantau ElastiCache for Redis dengan Metrik CloudWatch
- Memantau Memcached dengan Metrik CloudWatch
- Praktik terbaik pemantauan dengan Amazon ElastiCache for Redis menggunakan Amazon CloudWatch

- [Terbaik] Jika sumber daya ElastiCache Anda dibuat dengan CloudFormation, praktik terbaiknya adalah melakukan perubahan menggunakan templat CloudFormation untuk menjaga konsistensi operasional dan menghindari perubahan konfigurasi dan penyimpangan tumpukan yang tidak terkelola.

  [Sumber Daya]:

  - Referensi jenis sumber daya ElastiCache untuk CloudFormation

- [Terbaik] Otomatiskan operasi penskalaan Anda menggunakan data operasional klaster dan tentukan ambang batas di CloudWatch untuk menyiapkan alarm. Gunakan CloudWatch Events dan Simple Notification Service (SNS) untuk memicu fungsi Lambda dan menjalankan API ElastiCache untuk menskalakan klaster Anda secara otomatis. Contohnya adalah menambahkan serpihan ke klaster Anda ketika metrik `EngineCPUUtilization` mencapai 80% untuk jangka waktu yang lama. Pilihan lain adalah menggunakan `DatabaseMemoryUsedPercentages` untuk ambang batas berbasis memori.

  [Sumber Daya]:

  - Menggunakan Alarm Amazon CloudWatch
  - Apa itu peristiwa Amazon CloudWatch?
  - Menggunakan AWS Lambda dengan Amazon Simple Notification Service
  - Referensi API ElastiCache

## OE 3: Bagaimana cara Anda mengelola sumber daya klaster ElastiCache dan menjaga klaster Anda tetap terbaru?

Pendahuluan tingkat pertanyaan: Saat beroperasi dalam skala besar, penting bagi Anda untuk dapat menentukan dan mengidentifikasi semua sumber daya ElastiCache Anda. Saat meluncurkan fitur

aplikasi baru, Anda perlu menyamakan versi klaster di semua jenis lingkungan ElastiCache Anda: pengembangan, pengujian, dan produksi. Atribut sumber daya memungkinkan Anda memisahkan lingkungan untuk tujuan operasional yang berbeda-beda, seperti saat meluncurkan fitur baru dan mengaktifkan mekanisme keamanan baru.

Manfaat tingkat pertanyaan: Memisahkan lingkungan pengembangan, pengujian, dan produksi adalah praktik operasional terbaik. Praktik terbaik lainnya adalah klaster dan simpul Anda di seluruh lingkungan memiliki patch perangkat lunak terbaru yang diterapkan menggunakan proses yang dipahami dan terdokumentasi dengan baik. Dengan memanfaatkan fitur ElastiCache native, tim rekayasa Anda dapat fokus pada pemenuhan tujuan bisnis dan bukan pada pemeliharaan ElastiCache.

- [Terbaik] Jalankan klaster pada versi mesin terbaru yang tersedia dan terapkan Pembaruan Layanan Mandiri segera saat tersedia. ElastiCache secara otomatis memperbarui infrastruktur dasarnya selama periode pemeliharaan klaster yang Anda tentukan. Namun, simpul yang berjalan di klaster Anda diperbarui melalui Pembaruan Layanan Mandiri. Pembaruan ini dapat terdiri dari dua jenis: patch keamanan atau pembaruan perangkat lunak minor. Pastikan Anda memahami perbedaan berbagai jenis patch dan waktu penerapannya.

  [Sumber Daya]:

  - Pembaruan Layanan Mandiri di Amazon ElastiCache
  - Halaman Bantuan Pemeliharaan Terkelola dan Pembaruan Layanan Amazon ElastiCache
- [Terbaik] Atur sumber daya ElastiCache Anda menggunakan tag. Gunakan tag pada grup replikasi dan bukan pada simpul individu. Anda dapat mengonfigurasi tag agar ditampilkan saat Anda mengkueri sumber daya dan Anda dapat menggunakan tag untuk melakukan pencarian dan menerapkan filter. Anda harus menggunakan Grup Sumber Daya agar dapat dengan mudah membuat dan memelihara kumpulan sumber daya yang memiliki set tag yang sama.

  [Sumber Daya]:

  - Praktik Terbaik Pemberian Tag
  - Referensi jenis sumber daya ElastiCache untuk CloudFormation
  - Grup Parameter

## OE 4: Bagaimana cara mengelola koneksi klien ke klaster ElastiCache Anda?

Pengenalan tingkat pertanyaan: Saat beroperasi dalam skala besar, Anda perlu memahami bagaimana klien Anda terhubung dengan klaster ElastiCache untuk mengelola aspek operasional aplikasi Anda (seperti waktu respons).

Manfaat tingkat pertanyaan: Dengan memilih mekanisme koneksi yang paling tepat, aplikasi Anda dipastikan tidak akan terputus karena kesalahan konektivitas, seperti batas waktu habis.

- [Wajib] Pisahkan operasi baca dari tulis dan hubungkan ke simpul replika untuk menjalankan operasi baca. Namun, ketahuilah ketika Anda memisahkan operasi baca dan tulis, Anda akan kehilangan kemampuan untuk membaca kunci segera setelah menulisnya karena sifat replikasi Redis yang tidak sinkron. Perintah WAIT dapat dimanfaatkan untuk meningkatkan keamanan data dunia nyata dan memaksa replika mengonfrmasi penulisan sebelum merespons klien, dengan mengorbankan performa secara keseluruhan. Penggunaan simpul replika untuk operasi baca dapat dikonfigurasi di pustaka klien ElastiCache for Redis Anda menggunakan titik akhir pembaca ElastiCache untuk mode klaster dinonaktifkan. Untuk mode klaster diaktifkan, gunakan perintah READONLY ElastiCache for Redis. Untuk banyak pustaka klien ElastiCache for Redis, READONLY ElastiCache for Redis diimplementasikan secara default atau melalui pengaturan konfigurasi.

  [Sumber Daya]:

  - Menemukan titik akhir koneksi
  - READONLY
- [Wajib] Gunakan pooling koneksi. Pembuatan koneksi TCP akan menghabiskan waktu CPU di sisi klien dan server dan pooling memungkinkan Anda menggunakan kembali koneksi TCP.

  Untuk mengurangi overhead koneksi, Anda harus menggunakan pooling koneksi. Dengan pool koneksi, aplikasi Anda dapat menggunakan kembali dan melepaskan koneksi 'sesuka hati', tanpa perlu membuat koneksi. Anda dapat mengimplementasikan pooling koneksi melalui pustaka klien ElastiCache for Redis (jika didukung), dengan Framework yang tersedia untuk lingkungan aplikasi Anda, atau membuatnya dari awal.
- [Terbaik] Pastikan batas waktu soket klien diatur ke setidaknya satu detik (vs. pengaturan default "tidak ada" di beberapa klien).
  - Pengaturan nilai batas waktu yang terlalu rendah dapat menyebabkan kemungkinan batas waktu habis ketika beban server tinggi. Pengaturan yang terlalu tinggi dapat mengakibatkan aplikasi Anda membutuhkan waktu lama untuk mendeteksi masalah koneksi.

- Kendalikan volume koneksi baru dengan menerapkan pooling koneksi di aplikasi klien Anda. Hal ini akan mengurangi latensi dan pemanfaatan CPU yang diperlukan untuk membuka dan menutup koneksi, serta melakukan handshake TLS jika TLS diaktifkan di klaster.

[Sumber Daya]: Konfigurasikan Amazon ElastiCache for Redis untuk ketersediaan yang lebih tinggi

- [Baik] Menggunakan pipelining (jika kasus penggunaan Anda memungkinkannya) dapat meningkatkan performa secara signifikan.
  - Dengan pipelining, Anda akan mengurangi Waktu Pulang Pergi (RTT) antara klien aplikasi dan klaster serta permintaan baru dapat diproses bahkan jika klien belum membaca respons sebelumnya.
  - Dengan pipelining Anda dapat mengirim beberapa perintah ke server tanpa menunggu replies/ack. Kelemahan dari pipelining adalah ketika Anda akhirnya mengambil semua respons secara massal, mungkin ada kesalahan yang tidak akan Anda temukan sampai akhir.
  - Terapkan metode untuk mencoba kembali permintaan ketika ditampilkan kesalahan yang menghilangkan permintaan buruk.

  [Sumber Daya]: Pipelining

## OE 5: Bagaimana cara men-deploy Komponen ElastiCache untuk Beban Kerja?

Pengenalan tingkat pertanyaan: Lingkungan ElastiCache dapat di-deploy secara manual melalui Konsol AWS, atau secara programatis melalui API, CLI, toolkit, dll. Praktik terbaik Keunggulan Operasional menyarankan untuk mengotomatiskan deployment melalui kode jika memungkinkan. Selain itu, klaster ElastiCache dapat diisolasi berdasarkan beban kerja atau digabungkan untuk tujuan optimisasi biaya.

Manfaat tingkat pertanyaan: Dengan memilih mekanisme deployment yang paling tepat untuk lingkungan ElastiCache Anda, Keunggulan Operasional dapat meningkat dari waktu ke waktu. Sebaiknya lakukan operasi sebagai kode jika memungkinkan untuk meminimalkan kesalahan manusia dan meningkatkan pengulangan, fleksibilitas, dan waktu respons terhadap peristiwa.

Dengan memahami persyaratan isolasi beban kerja, Anda dapat memilih untuk memiliki lingkungan ElastiCache khusus per beban kerja atau menggabungkan beberapa beban kerja menjadi satu klaster, atau kombinasinya. Memahami kompromi dapat membantu mencapai keseimbangan antara Keunggulan Operasional dan Optimisasi Biaya

- [Wajib] Pahami opsi deployment yang tersedia untuk ElastiCache, dan otomatiskan prosedur ini jika memungkinkan. Kemungkinan cara otomatisasi termasuk CloudFormation, AWS CLI/SDK, dan API.

  [Sumber Daya]:

  - Referensi jenis sumber daya Amazon ElastiCache
  - elasticache
  - Referensi API Amazon ElastiCache
- [Wajib] Untuk semua beban kerja, tentukan tingkat isolasi klaster yang diperlukan.
  - [Terbaik]: Isolasi Tinggi – pemetaan beban kerja ke klaster 1:1. Memungkinkan kontrol paling terperinci atas akses, ukuran, penskalaan, dan pengelolaan sumber daya ElastiCache berdasarkan per beban kerja.
  - [Lebih Baik]: Isolasi Sedang – M:1 diisolasi berdasarkan tujuan, tetapi mungkin dibagi di beberapa beban kerja (misalnya klaster yang dikhususkan untuk caching beban kerja, dan yang lain dikhususkan untuk pesan).
  - [Baik]: Isolasi Rendah – M:1 semua tujuan, dibagikan sepenuhnya. Direkomendasikan untuk beban kerja di mana akses bersama dapat diterima.

## OE 6: Bagaimana cara melakukan perencanaan dan mitigasi terhadap kegagalan?

Pengenalan tingkat pertanyaan: Keunggulan Operasional mencakup antisipasi kegagalan dengan melakukan latihan "pre-mortem" reguler untuk mengidentifikasi sumber kegagalan potensial sehingga dapat diatasi atau dimitigasi. ElastiCache menawarkan API Failover yang memungkinkan simulasi peristiwa kegagalan simpul, demi tujuan pengujian.

Manfaat tingkat pertanyaan: Dengan menguji skenario kegagalan terlebih dahulu, Anda dapat mempelajari bagaimana pengaruhnya terhadap beban kerja Anda. Hal ini memungkinkan pengujian prosedur respons yang aman dan efektivitasnya, serta membuat tim Anda terbiasa dengan eksekusinya.

[Wajib] Lakukan pengujian failover secara berkala di akun dev/tes. TestFailover

## OE 7: Bagaimana cara memecahkan masalah peristiwa mesin Redis?

Pengenalan tingkat pertanyaan: Keunggulan Operasional membutuhkan kemampuan untuk menyelidiki informasi tingkat layanan dan tingkat mesin untuk menganalisis kondisi dan status klaster

Anda. Amazon ElastiCache for Redis dapat mengirimkan log mesin Redis ke Amazon CloudWatch dan Amazon Kinesis Data Firehose.

Manfaat tingkat pertanyaan: Pengaktifan log mesin Redis di klaster Amazon ElastiCache for Redis akan memberikan wawasan tentang peristiwa yang memengaruhi kondisi dan performa klaster. Log mesin Redis menyediakan data langsung dari mesin Redis yang tidak tersedia melalui mekanisme peristiwa ElastiCache. Melalui pengamatan yang cermat terhadap peristiwa ElastiCache (lihat OE-1 di atas) dan log mesin Redis, urutan peristiwa dapat ditentukan saat memecahkan masalah dari perspektif layanan ElastiCache dan perspektif mesin Redis.

- [Wajib] Pastikan fungsionalitas logging mesin Redis diaktifkan, yang tersedia pada ElastiCache for Redis 6.2 dan yang lebih baru. Hal ini dapat dilakukan selama pembuatan klaster atau dengan memodifikasi klaster setelah pembuatan.
  - Tentukan apakah Log Amazon CloudWatch atau Amazon Kinesis Data Firehose adalah target yang tepat untuk log mesin Redis.
  - Pilih log target yang sesuai dalam CloudWatch atau Kinesis Data Firehose untuk mempertahankan log. Jika Anda memiliki beberapa klaster, pertimbangkan log target yang berbeda untuk setiap klaster karena ini akan membantu mengisolasi data saat pemecahan masalah.

  [Sumber Daya]:
  - Pengiriman log: Pengiriman log
  - Destinasi logging: Log Amazon CloudWatch
  - Pengenalan Log Amazon CloudWatch: Apa itu Log Amazon CloudWatch?
  - Pengenalan Amazon Kinesis Data Firehose: Apa Itu Amazon Kinesis Data Firehose?
- [Terbaik] Jika menggunakan Log Amazon CloudWatch, pertimbangkan untuk memanfaatkan Wawasan Log Amazon CloudWatch untuk mengkueri log mesin Redis untuk informasi penting.

  Sebagai contoh, buat kueri terhadap grup Log CloudWatch yang berisi log mesin Redis yang akan menampilkan peristiwa dengan LogLevel 'WARNING', seperti:

```
fields @timestamp, LogLevel, Message
| sort @timestamp desc
| filter LogLevel = "WARNING"
```

  [Sumber Daya]:Menganalisis data log dengan Wawasan Log CloudWatch

# Pilar Keamanan Lensa Amazon ElastiCache Well-Architected

Pilar keamanan berfokus pada perlindungan informasi dan sistem. Topik utama meliputi kerahasiaan dan integritas data, mengidentifikasi dan mengelola siapa yang dapat melakukan apa dengan manajemen berbasis hak istimewa, melindungi sistem, dan menetapkan kontrol untuk mendeteksi peristiwa keamanan.

Topik



## SEC 1: Langkah apa yang Anda ambil dalam mengontrol akses resmi ke data ElastiCache?

Pengenalan tingkat pertanyaan: Semua klaster ElastiCache dirancang untuk diakses dari instans Amazon Elastic Compute Cloud dalam VPC, fungsi nirserver (AWS Lambda), atau kontainer (Amazon Elastic Container Service). Skenario yang paling sering ditemui adalah mengakses klaster ElastiCache dari instans Amazon Elastic Compute Cloud di Amazon Virtual Private Cloud (Amazon Virtual Private Cloud) yang sama. Sebelum Anda dapat menyambung ke klaster dari instans Amazon EC2, Anda harus memberikan otorisasi pada instans Amazon EC2 untuk mengakses klaster tersebut. Untuk mengakses klaster ElastiCache yang berjalan di VPC, Anda perlu memberikan izin masuk jaringan ke klaster tersebut.

Manfaat tingkat pertanyaan: Masuknya jaringan ke dalam klaster dikendalikan melalui grup keamanan VPC. Grup keamanan bertindak sebagai firewall virtual untuk instans Amazon EC2 Anda untuk mengontrol lalu lintas masuk dan ke luar. Aturan-aturan masuk mengontrol lalu lintas yang masuk ke instans Anda, dan aturan-aturan ke luar mengontrol lalu lintas yang ke luar dari instans Anda. Dalam kasus ElastiCache, saat meluncurkan klaster, diperlukan asosiasi grup keamanan. Ini memastikan bahwa aturan lalu lintas masuk dan keluar berlaku untuk semua simpul yang membentuk

klaster. Selain itu, ElastiCache dikonfigurasi untuk melakukan deploy pada subnet privat secara eksklusif sehingga hanya dapat diakses dari melalui jaringan privat VPC.

- [Wajib] Grup keamanan yang terkait dengan klaster Anda mengontrol masuknya jaringan dan akses ke klaster. Secara default, grup keamanan tidak akan memiliki aturan masuk yang ditentukan dan, oleh karena itu, tidak ada jalur masuk ke ElastiCache. Untuk mengaktifkan ini, konfigurasikan aturan masuk pada grup keamanan yang menentukan rentang/alamat IP sumber, lalu lintas jenis TCP, dan port untuk klaster ElastiCache Anda (misalnya port default 6379 untuk ElastiCache for Redis). Meskipun dimungkinkan untuk mengizinkan serangkaian sumber masuk yang sangat luas, seperti semua sumber daya dalam VPC (0.0.0.0/0), disarankan untuk sedetail mungkin dalam mendefinisikan aturan masuk seperti hanya mengotorisasi akses masuk ke klien Redis yang berjalan di instans Amazon EC2 yang terkait dengan grup keamanan tertentu.

  [Sumber Daya]:

  - Subnet dan grup subnet
  - Mengakses klaster atau grup replikasi Anda
  - Kontrol lalu lintas ke sumber daya menggunakan grup keamanan
  - Grup keamanan Amazon Elastic Compute Cloud untuk instans Linux
- [Wajib] Kebijakan AWS Identity and Access Management dapat ditetapkan ke fungsi AWS Lambda yang memungkinkannya mengakses data ElastiCache. Untuk mengaktifkan fitur ini, buat peran eksekusi IAM dengan izin `AWSLambdaVPCAccessExecutionRole`, lalu tetapkan peran ke fungsi AWS Lambda tersebut.

  [Sumber Daya]: Mengonfigurasi fungsi Lambda untuk mengakses Amazon ElastiCache di Amazon VPC: Tutorial: Mengonfigurasi fungsi Lambda untuk mengakses Amazon ElastiCache di VPC Amazon

## SEC 2: Apakah aplikasi Anda memerlukan otorisasi tambahan untuk ElastiCache di atas dan di atas kontrol berbasis jaringan?

Pengantar tingkat pertanyaan: Dalam skenario di mana perlu untuk membatasi atau mengontrol akses ke ElastiCache untuk klaster Redis pada tingkat klien individu, disarankan untuk mengautentikasi melalui perintah ElastiCache for Redis AUTH. Token autentikasi ElastiCache for Redis, dengan manajemen pengguna dan grup pengguna opsional, memungkinkan ElastiCache for Redis untuk mensyaratkan kata sandi sebelum mengizinkan klien menjalankan perintah dan kunci akses, sehingga meningkatkan keamanan bidang data.

Manfaat tingkat pertanyaan: Untuk membantu menjaga keamanan data Anda, ElastiCache for Redis menyediakan mekanisme untuk melindungi terhadap akses yang tidak sah atas data Anda. Ini termasuk menegakkan Kontrol Akses Berbasis Pesan (RBAC) AUTH, atau token (kata sandi) AUTH yang digunakan oleh klien untuk terhubung ke ElastiCache sebelum melakukan perintah resmi.

- [Terbaik] Untuk ElastiCache for Redis 6.x dan yang lebih tinggi, tentukan kontrol autentikasi dan otorisasi dengan mendefinisikan grup pengguna, pengguna, dan string akses. Tetapkan pengguna ke grup pengguna, lalu tetapkan grup pengguna ke klaster. Untuk memanfaatkan RBAC, itu harus dipilih pada pembuatan klaster, dan enkripsi in-transit harus diaktifkan. Pastikan Anda menggunakan klien Redis yang mendukung TLS untuk dapat memanfaatkan RBAC.

  [Sumber Daya]:

  - Menerapkan RBAC ke Grup Replikasi untuk ElastiCache for Redis
  - Menentukan Izin Menggunakan String Akses
  - ACL
  - Versi ElastiCache for Redis yang Didukung
- [Terbaik] Untuk ElastiCache for Redis versi sebelum 6.x, selain mengatur token/kata sandi yang kuat dan mempertahankan kebijakan kata sandi yang ketat untuk ElastiCache for Redis AUTH, praktik terbaiknya adalah memutar kata sandi/token. ElastiCache dapat mengelola hingga dua (2) token autentikasi pada waktu tertentu. Anda juga dapat memodifikasi klaster untuk secara eksplisit memerlukan penggunaan token autentikasi.

  [Sumber daya]: Memodifikasi token AUTH pada klaster ElastiCache for Redis

## SEC 3: Apakah ada risiko bahwa perintah dapat dijalankan secara tidak sengaja yang menyebabkan kehilangan atau kegagalan data?

Pengantar tingkat pertanyaan: Ada sejumlah perintah Redis yang dapat berdampak buruk pada operasi jika dijalankan secara tidak sengaja atau oleh aktor jahat. Perintah ini dapat memiliki konsekuensi yang tidak diinginkan dari perspektif kinerja dan keamanan data. Misalnya pengembang dapat secara rutin memanggil perintah FLUSHALL di lingkungan pengembangan, dan karena kesalahan mungkin secara tidak sengaja mencoba memanggil perintah ini pada sistem produksi, yang mengakibatkan kehilangan data yang tidak disengaja.

Manfaat tingkat pertanyaan: Dimulai dengan ElastiCache for Redis 5.0.3 di ElastiCache, Anda memiliki kemampuan untuk mengganti nama perintah tertentu yang mungkin mengganggu beban

kerja Anda. Mengganti nama perintah dapat membantu mencegahnya dieksekusi secara tidak sengaja di klaster.

- [Wajib]

  [Sumber Daya]:

  - ElastiCache for Redis versi 5.0.3 (usang, gunakan versi 5.0.6)
  - Perubahan parameter Redis 5.0.3
  - Keamanan Redis

## SEC 4: Bagaimana Anda memastikan enkripsi data diam dengan ElastiCache

Pengantar tingkat pertanyaan: Meskipun ElastiCache for Redis adalah penyimpanan data dalam memori, dimungkinkan untuk mengenkripsi data apa pun yang mungkin disimpan (pada penyimpanan) sebagai bagian dari operasi standar klaster. Ini termasuk pencadangan terjadwal dan manual yang ditulis ke Amazon S3, serta data yang disimpan ke penyimpanan disk sebagai hasil dari operasi sinkronisasi dan swap. Jenis instans dalam keluarga M6g dan R6g juga memiliki fitur enkripsi dalam memori yang selalu aktif.

Manfaat tingkat pertanyaan: ElastiCache for Redis menyediakan enkripsi diam opsional untuk meningkatkan keamanan data.

- [Wajib] Enkripsi diam dapat diaktifkan pada klaster ElastiCache (grup replikasi) hanya pada saat pembuatannya. Klaster yang ada tidak dapat dimodifikasi untuk mulai mengenkripsi data diam. Secara default, ElastiCache akan menyediakan dan mengelola kunci yang digunakan dalam enkripsi diam.

  [Sumber Daya]:

  - Kondisi Enkripsi Diam
  - Mengaktifkan Enkripsi Diam
- [Terbaik] Manfaatkan jenis instans Amazon EC2 yang mengenkripsi data saat berada di memori (seperti M6g atau R6g). Jika memungkinkan, pertimbangkan untuk mengelola kunci Anda sendiri untuk enkripsi diam. Untuk lingkungan keamanan data yang lebih ketat, AWS Key Management Service (KMS) dapat digunakan untuk mengelola sendiri Kunci Utama Pelanggan (CMK). Melalui integrasi ElastiCache dengan AWS Key Management Service, Anda dapat membuat, memiliki, dan mengelola kunci yang digunakan untuk enkripsi data diam untuk klaster ElastiCache for Redis Anda.

[Sumber Daya]:

- Menggunakan kunci yang dikelola pelanggan dari AWS Key Management Service
- AWS Layanan Pengelolaan Kunci
- AWS Konsep KMS

## SEC 5: Bagaimana Anda mengenkripsi data bergerak dengan ElastiCache?

Pengantar tingkat pertanyaan:  Merupakan persyaratan umum untuk mengurangi data yang dikompromikan saat dalam perjalanan. Ini mewakili data dalam komponen sistem terdistribusi, serta antara klien aplikasi dan simpul klaster. ElastiCache for Redis mendukung persyaratan ini dengan memungkinkan enkripsi data bergerak antara klien dan klaster, dan antara simpul klaster itu sendiri. Jenis instans dalam keluarga M6g dan R6g juga memiliki fitur enkripsi dalam memori yang selalu aktif.

Manfaat tingkat pertanyaan: Enkripsi in-transit Amazon ElastiCache adalah fitur opsional yang memungkinkan Anda untuk meningkatkan keamanan data Anda pada titik yang paling rentan, saat data dalam keadaan bergerak dari satu lokasi ke lokasi lain.

- [Wajib] Enkripsi in-transit dapat diaktifkan pada klaster ElastiCache for Redis (grup replikasi) hanya pada saat pembuatannya. Harap dicatat bahwa, karena pemrosesan tambahan yang diperlukan untuk mengenkripsi/mendekripsi data, penerapan enkripsi in-transit akan memiliki beberapa dampak kinerja. Untuk memahami dampaknya, disarankan untuk membandingkan beban kerja Anda sebelum dan sesudah mengaktifkan enkripsi-in-transit.

  [Sumber Daya]:

  - Gambaran umum enkripsi in-transit

## SEC 6: Bagaimana Anda membatasi akses untuk bidang kontrol sumber daya?

Pengantar tingkat pertanyaan: Kebijakan IAM dan ARN memungkinkan kontrol akses berbutir halus untuk ElastiCache for Redis, memungkinkan kontrol yang lebih ketat untuk mengelola pembuatan, modifikasi, dan penghapusan klaster ElastiCache for Redis.

Manfaat tingkat pertanyaan: Pengelolaan sumber daya Amazon ElastiCache, seperti grup replikasi, simpul, dll. Dapat dibatasi ke AWS akun yang memiliki izin khusus berdasarkan kebijakan IAM, meningkatkan keamanan dan keandalan sumber daya.

- [Wajib] Kelola akses ke sumber daya Amazon ElastiCache dengan menetapkan kebijakan AWS Identity and Access Management khusus kepada AWS pengguna, memungkinkan kontrol yang lebih baik atas akun mana yang dapat melakukan tindakan apa pada klaster.

  [Sumber Daya]:

  - Gambaran umum pengelolaan izin akses untuk sumber daya ElastiCache Anda
  - Menggunakan kebijakan berbasis identitas (kebijakan IAM) untuk Amazon ElastiCache

## SEC 7: Bagaimana Anda mendeteksi dan menanggapi peristiwa keamanan?

Pengantar tingkat pertanyaan: ElastiCache, saat dilakukan deploy dengan RBAC aktif, mengekspor metrik CloudWatch untuk memberi tahu pengguna tentang peristiwa keamanan. Metrik ini membantu mengidentifikasi upaya gagal untuk autentikasi, kunci akses, atau menjalankan perintah yang menghubungkan pengguna RBAC tidak diizinkan.

Selain itu, sumber daya produk dan layanan AWS membantu mengamankan beban kerja Anda secara keseluruhan dengan mengotomatiskan deployment dan mencatat log semua tindakan dan modifikasi untuk peninjauan/audit nanti.

Manfaat tingkat pertanyaan: Dengan memantau peristiwa, Anda memungkinkan organisasi Anda untuk merespons sesuai dengan persyaratan, kebijakan, dan prosedur Anda. Mengotomatiskan pemantauan dan respons terhadap peristiwa keamanan ini memperkuat postur keamanan Anda secara keseluruhan.

- [Wajib] Biasakan diri Anda dengan Metrik CloudWatch yang diterbitkan yang berkaitan dengan autentikasi RBAC dan kegagalan otorisasi.
  - AuthenticationFailures = Upaya gagal untuk mengautentikasi ke Redis
  - KeyAuthorizationFailures = Upaya gagal oleh pengguna untuk kunci akses tanpa izin
  - CommandAuthorizationFailures = Upaya gagal oleh pengguna untuk menjalankan perintah tanpa izin

  [Sumber Daya]:

  - Metrik untuk Redis
- [Terbaik] Disarankan untuk mengatur peringatan dan pemberitahuan pada metrik ini dan merespons seperlunya.

  [Sumber Daya]:

  - Menggunakan alarm Amazon CloudWatch
- [Terbaik] Gunakan perintah Redis ACL LOG untuk mengumpulkan rincian lebih lanjut

  [Sumber Daya]:

  - ACL LOG
- [Terbaik] Biasakan diri Anda dengan kemampuan produk dan layanan AWS yang berkaitan dengan pemantauan, pencatatan log, dan analisis deployment serta peristiwa ElastiCache

  [Sumber Daya]:

  - Membuat log panggilan API Amazon ElastiCache dengan CloudTrail AWS
  - elasticache-redis-cluster-automatic-backup-check
  - Memantau penggunaan dengan Metrik CloudWatch

## Pilar Keandalan Lensa Amazon ElastiCache Well-Architected

Topik



### REL 1: Bagaimana Anda mendukung deployment arsitektur ketersediaan tinggi (HA)?

Pengenalan tingkat pertanyaan:  Memahami arsitektur ketersediaan tinggi Amazon ElastiCache akan memungkinkan Anda beroperasi dalam keadaan tangguh selama acara ketersediaan.

Manfaat tingkat pertanyaan: Merancang klaster ElastiCache Anda agar tahan terhadap kegagalan memastikan ketersediaan yang lebih tinggi untuk deployment ElasticCache Anda.

- [Wajib] Tentukan tingkat keandalan yang Anda butuhkan untuk klaster ElastiCache Anda. Beban kerja yang berbeda memiliki standar ketahanan yang berbeda, dari yang sepenuhnya fana hingga beban kerja kritis misi. Tentukan kebutuhan untuk setiap jenis lingkungan yang Anda operasikan seperti dev, test, dan production.

  Mesin caching: Memcached vs ElastiCache for Redis

1. Memcached tidak menyediakan mekanisme replikasi apa pun dan digunakan terutama untuk beban kerja sementara.
2. ElastiCache for Redis menawarkan fitur HA yang dibahas di bawah ini

- [Terbaik] Untuk beban kerja yang membutuhkan HA, gunakan ElastiCache for Redis dalam mode klaster dengan minimal dua replika per serpihan, bahkan untuk beban kerja persyaratan throughput kecil yang hanya membutuhkan satu serpihan.
  1. Untuk mode klaster diaktifkan, Multi-AZ diaktifkan secara otomatis.

     Multi-AZ meminimalkan waktu henti dengan melakukan failover otomatis dari simpul primer ke replika, jika terjadi pemeliharaan yang direncanakan atau tidak direncanakan serta mengurangi kegagalan AZ.
  2. Untuk beban kerja serpihan, minimal tiga serpihan memberikan pemulihan yang lebih cepat selama peristiwa failover karena Protokol Klaster Redis memerlukan mayoritas simpul primer tersedia untuk mencapai kuorum.
  3. Siapkan dua atau lebih replika di seluruh Ketersediaan.

     Memiliki dua replika memberikan peningkatan skalabilitas baca dan juga ketersediaan baca dalam skenario di mana satu replika sedang menjalani pemeliharaan.
  4. Gunakan tipe simpul berbasis Graviton2 (simpul default di sebagian besar wilayah).

     Amazon ElastiCache for Redis telah menambahkan kinerja yang dioptimalkan pada simpul ini. Hasilnya, Anda mendapatkan kinerja replikasi dan sinkronisasi yang lebih baik, sehingga meningkatkan ketersediaan secara keseluruhan.
  5. Monitor dan ukuran yang tepat untuk menangani puncak lalu lintas yang diantisipasi: di bawah beban berat, mesin ElastiCache for Redis mungkin menjadi tidak responsif, yang memengaruhi ketersediaan. `BytesUsedForCache` dan `DatabaseMemoryUsagePercentage` merupakan indikator yang baik dari penggunaan memori Anda, sedangkan `ReplicationLag` merupakan indikator kesehatan replikasi Anda berdasarkan tingkat tulis Anda. Anda dapat menggunakan metrik ini untuk memicu penskalaan klaster.
  6. Pastikan ketahanan sisi klien dengan menguji dengan Failover API sebelum peristiwa failover produksi.

  [Sumber Daya]:

  - Konfigurasikan Amazon ElastiCache for Redis untuk ketersediaan yang lebih tinggi
  - Ketersediaan tinggi menggunakan grup replikasi

## REL 2: Bagaimana Anda memenuhi Sasaran Titik Pemulihan (RPO) Anda dengan ElastiCache?

Pengenalan tingkat pertanyaan: Memahami RPO beban kerja untuk menginformasikan keputusan tentang strategi pencadangan dan pemulihan ElastiCache.

Manfaat tingkat pertanyaan: Memiliki strategi RPO di tempat dapat meningkatkan kelangsungan bisnis jika terjadi skenario pemulihan bencana. Merancang kebijakan pencadangan dan pemulihan dapat membantu Anda memenuhi Sasaran Titik Pemulihan (RPO) untuk data ElastiCache Anda. ElastiCache for Redis menawarkan kemampuan snapshot yang disimpan di Amazon S3, bersama dengan kebijakan retensi yang dapat dikonfigurasi. Snapshot ini diambil selama jendela cadangan yang ditentukan, dan ditangani oleh layanan secara otomatis. Jika beban kerja Anda memerlukan perincian cadangan tambahan, Anda memiliki opsi untuk membuat hingga 20 cadangan manual per hari. Pencadangan yang dibuat secara manual tidak memiliki kebijakan retensi layanan dan dapat disimpan tanpa batas waktu.

- [Wajib] Memahami dan mendokumentasikan RPO deployment ElastiCache Anda.
  - Ketahuilah bahwa Memcached tidak menawarkan proses pencadangan apa pun.
  - Tinjau kemampuan fitur Pencadangan dan Pemulihan Elasticache.
- [Terbaik] Miliki proses yang dikomunikasikan dengan baik untuk membuat cadangan klaster Anda.
  - Memulai pencadangan manual sesuai kebutuhan.
  - Tinjau kebijakan retensi untuk pencadangan otomatis.
  - Perhatikan bahwa cadangan manual akan dipertahankan tanpa batas waktu.
  - Jadwalkan pencadangan otomatis Anda selama periode penggunaan rendah.
  - Lakukan operasi pencadangan terhadap replika baca untuk memastikan Anda meminimalkan dampak pada kinerja klaster.
- [Bagus] Manfaatkan fitur pencadangan terjadwal ElastiCache untuk mencadangkan data Anda secara teratur selama jendela yang ditentukan.
  - Tes secara berkala mengembalikan dari cadangan Anda.
- [Sumber Daya]:
  - Redis
  - Pencadangan dan pemulihan untuk ElastiCache for Redis
  - Membuat cadangan manual
  - Menjadwalkan cadangan otomatis

- Pencadangan dan Pemulihan Klaster Redis ElastiCache

## REL 3: Bagaimana Anda mendukung persyaratan pemulihan bencana (DR)?

Pengenalan tingkat pertanyaan: Pemulihan bencana adalah aspek penting dari setiap perencanaan beban kerja. ElastiCache for Redis menawarkan beberapa opsi untuk menerapkan pemulihan bencana berdasarkan persyaratan ketahanan beban kerja. Dengan Penyimpanan Data Global Amazon ElastiCache for Redis, Anda dapat menulis ke klaster ElastiCache for Redis di satu wilayah dan memiliki data yang tersedia untuk dibaca dari dua klaster replika lintas wilayah lainnya, sehingga memungkinkan pembacaan latensi rendah dan pemulihan bencana di seluruh wilayah.

Manfaat tingkat pertanyaan: Memahami dan merencanakan berbagai skenario bencana dapat memastikan kelangsungan bisnis. Strategi DR harus seimbang dengan biaya, dampak kinerja, dan potensi kehilangan data.

- [Wajib] Kembangkan dan dokumentasikan strategi DR untuk semua komponen ElastiCache Anda berdasarkan persyaratan beban kerja. ElastiCache unik karena beberapa kasus penggunaan sepenuhnya fana dan tidak memerlukan strategi DR apa pun, sedangkan yang lain berada di ujung spektrum yang berlawanan dan memerlukan strategi DR yang sangat kuat. Semua opsi harus ditimbang terhadap Optimalisasi Biaya - ketahanan yang lebih besar membutuhkan jumlah instrastruktur yang lebih besar.

  Memahami opsi DR yang tersedia di tingkat regional dan multi-wilayah.

  - Deployment Multi-AZ direkomendasikan untuk mencegah kegagalan AZ. Pastikan untuk melakukan deploy dengan Klaster-Mode diaktifkan dalam arsitektur Multi-AZ, dengan minimal 3 AZ yang tersedia.
  - Penyimpanan Data Global direkomendasikan untuk menjaga terhadap kegagalan regional.
- [Terbaik] Aktifkan Penyimpanan Data Global untuk beban kerja yang membutuhkan ketahanan tingkat wilayah.
  - Memiliki rencana untuk failover ke wilayah sekunder jika terjadi degradasi primer.
  - Uji proses failover multi-wilayah sebelum failover selesai dalam produksi.
  - Pantau metrik `ReplicationLag` untuk memahami potensi dampak kehilangan data selama peristiwa failover.
- [Sumber Daya]:
  - Mitigasi Kegagalan
  - Replikasi di seluruh AWS Wilayah menggunakan penyimpanan data global

- Memulihkan dari cadangan dengan opsi perubahan ukuran klaster
- Meminimalkan waktu henti di ElastiCache for Redis dengan Multi-AZ

## REL 4: Bagaimana Anda merencanakan failover secara efektif?

Pengenalan tingkat pertanyaan: Mengaktifkan Multi-AZ dengan failover otomatis adalah praktik terbaik ElasticCache. Dalam kasus tertentu, ElastiCache for Redis menggantikan simpul primer sebagai bagian dari operasi layanan. Contohnya termasuk peristiwa pemeliharaan yang direncanakan dan kasus kegagalan simpul atau masalah zona ketersediaan yang tidak mungkin terjadi. Failover yang berhasil bergantung pada ElastiCache dan konfigurasi pustaka klien Anda.

Manfaat tingkat pertanyaan: Mengikuti praktik terbaik untuk failover ElastiCache bersama dengan pustaka klien ElasticCache for Redis spesifik Anda membantu Anda meminimalkan potensi waktu henti selama peristiwa failover.

- [Wajib] Untuk mode klaster dinonaktifkan, gunakan batas waktu sehingga klien Anda mendeteksi jika perlu memutuskan sambungan dari simpul primer lama dan menyambung kembali ke simpul primer baru, menggunakan alamat IP titik akhir primer yang diperbarui. Untuk mode klaster diaktifkan, pustaka klien bertanggung jawab untuk mendeteksi perubahan dalam topologi klaster yang mendasarinya. Ini paling sering dilakukan dengan pengaturan konfigurasi di pustaka klien ElastiCache for Redis, yang juga memungkinkan Anda untuk mengonfigurasi frekuensi dan metode penyegaran. Setiap pustaka klien menawarkan pengaturannya sendiri dan detail lebih lanjut tersedia dalam dokumentasi yang sesuai.

  [Sumber Daya]:

  - Meminimalkan waktu henti di ElastiCache for Redis dengan Multi-AZ
  - Tinjau praktik terbaik pustaka klien ElastiCache for Redis Anda.
- [Wajib] Failover yang berhasil bergantung pada lingkungan replikasi yang sehat antara simpul primer dan replika. Tinjau dan pahami sifat asinkron replikasi Redis, serta metrik CloudWatch yang tersedia untuk melaporkan jeda replikasi antara simpul primer dan replika. Untuk kasus penggunaan yang membutuhkan keamanan data yang lebih besar, manfaatkan perintah Redis WAIT untuk memaksa replika mengakui penulisan sebelum menanggapi klien yang terhubung.

  [Sumber Daya]:

  - Metrik untuk Redis
  - Praktik terbaik pemantauan dengan Amazon ElastiCache for Redis menggunakan Amazon CloudWatch

- [Terbaik] Secara teratur memvalidasi respons aplikasi Anda selama failover menggunakan API ElasticCache Test Failover.

  [Sumber Daya]:

  - Menguji Failover Otomatis ke Replika Baca di Amazon ElastiCache for Redis
  - Menguji failover otomatis

## REL 5: Apakah komponen ElastiCache Anda dirancang untuk menskalakan?

Pengenalan tingkat pertanyaan: Dengan memahami kemampuan penskalaan dan topologi deployment yang tersedia, komponen ElastiCache Anda dapat menyesuaikan dari waktu ke waktu untuk memenuhi persyaratan beban kerja yang berubah. ElastiCache menawarkan penskalaan 4 arah: masuk/keluar (horizontal) serta atas/bawah (vertikal).

Manfaat tingkat pertanyaan: Mengikuti praktik terbaik untuk deployment ElasticCache memberikan fleksibilitas penskalaan terbesar, serta memenuhi prinsip Well Architected dalam penskalaan secara horizontal untuk meminimalkan dampak kegagalan.

- [Wajib] Memahami perbedaan antara topologi Klaster-mode Diaktifkan dan Klaster-mode Dinonaktifkan. Dalam hampir semua kasus, disarankan untuk melakukan deployment dengan mode Klaster diaktifkan karena memungkinkan skalabilitas yang lebih besar dari waktu ke waktu. Komponen mode klaster yang dinonaktifkan terbatas dalam kemampuannya untuk menskalakan secara horizontal dengan menambahkan replika baca.
- [Wajib] Memahami kapan dan bagaimana menskalakan.
  - Untuk READIOPS lainnya: tambahkan replika
  - Untuk WRITEOPS lainnya: tambahkan serpihan (menskalakan ke luar)
  - Untuk IO jaringan lainnya - gunakan instans yang dioptimalkan jaringan, menaikkan skala
- [Terbaik] Deploy komponen ElastiCache Anda dengan mode Klaster diaktifkan, dengan bias terhadap lebih banyak simpul yang lebih kecil daripada simpul yang lebih sedikit dan lebih besar. Ini secara efektif membatasi radius ledakan kegagalan simpul.
- [Terbaik] Sertakan replika di klaster Anda untuk meningkatkan respons selama acara penskalaan
- [Bagus] Untuk mode klaster dinonaktifkan, manfaatkan replika baca untuk meningkatkan kapasitas baca secara keseluruhan. ElastiCache memiliki dukungan hingga 5 replika baca dalam mode klaster dinonaktifkan, serta penskalaan vertikal.
- [Sumber Daya]:

- Penskalaan klaster ElastiCache for Redis
- Menaikkan skala secara online
- Menskalakan klaster ElastiCache for Memcached

# Pilar Efisiensi Kinerja Lensa Amazon ElastiCache yang Dirancang dengan Baik

Pilar efisiensi kinerja berfokus pada penggunaan sumber daya TI dan komputasi secara efisien. Topik utamanya meliputi pemilihan jenis dan ukuran sumber daya yang tepat berdasarkan persyaratan beban kerja, pemantauan kinerja, dan pembuatan keputusan berdasarkan informasi untuk mempertahankan efisiensi seiring dengan berkembangnya kebutuhan bisnis.

Topik

- PE 1: Bagaimana cara memantau kinerja klaster Amazon ElastiCache Anda?
- PE 2: Bagaimana cara mendistribusikan pekerjaan di seluruh Simpul klaster ElastiCache Anda?
- PE 3: Untuk beban kerja caching, bagaimana cara melacak dan melaporkan efektivitas dan kinerja cache Anda?
- PE 4: Bagaimana beban kerja Anda mengoptimalkan penggunaan sumber daya dan koneksi jaringan?
- PE 5: Bagaimana cara mengelola penghapusan kunci dan/atau pengosongan?
- PE 6: Bagaimana Anda memodelkan dan berinteraksi dengan data di ElastiCache?
- PE 7: Bagaimana cara log perintah yang berjalan lambat di klaster Amazon ElastiCache Anda?
- PE8: Bagaimana Penskalaan Otomatis membantu meningkatkan kinerja klaster ElastiCache?

## PE 1: Bagaimana cara memantau kinerja klaster Amazon ElastiCache Anda?

Pengenalan tingkat pertanyaan: Dengan memahami metrik pemantauan yang ada, Anda dapat mengidentifikasi pemanfaatan saat ini. Pemantauan yang tepat dapat membantu mengidentifikasi potensi hambatan yang memengaruhi performa klaster Anda.

Manfaat tingkat pertanyaan: Memahami metrik yang terkait dengan klaster Anda dapat membantu memandu teknik pengoptimalan yang dapat mengurangi latensi dan peningkatan throughput.

- [Wajib] Pengujian kinerja dasar menggunakan subset beban kerja Anda.

  - Anda harus memantau kinerja beban kerja aktual menggunakan mekanisme seperti pengujian beban.
  - Pantau metrik CloudWatch saat menjalankan pengujian ini untuk mendapatkan pemahaman tentang metrik yang tersedia, dan untuk menetapkan garis dasar kinerja.
- [Terbaik] Untuk beban kerja ElastiCache for Redis, ganti nama perintah yang mahal secara komputasi, seperti `KEYS`, untuk membatasi kemampuan pengguna menjalankan perintah pemblokiran pada klaster produksi.
  - Beban kerja ElastiCache for Redis yang menjalankan mesin 6.x, dapat memanfaatkan kontrol akses berbasis peran untuk membatasi perintah tertentu. Akses ke perintah dapat dikontrol dengan membuat Pengguna dan Grup Pengguna dengan Konsol atau CLI AWS, dan mengaitkan Grup Pengguna ke klaster ElastiCache for Redis. Di Redis 6, ketika RBAC diaktifkan, kita dapat menggunakan “- @dangerous" dan itu akan menolak perintah mahal seperti KEYS, MONITOR, SORT, dll. untuk pengguna tersebut.
  - Untuk mesin versi 5.x, ganti nama perintah menggunakan parameter `rename-commands` pada grup parameter klaster Amazon ElastiCache for Redis.
- [Lebih baik] Analisis kueri lambat dan cari teknik pengoptimalan.
  - Untuk beban kerja ElastiCache for Redis, pelajari lebih lanjut kueri Anda dengan menganalisis Log Lambat. Misalnya, Anda dapat menggunakan perintah berikut, `redis-cli slowlog get 10` untuk menampilkan 10 perintah terakhir yang melebihi ambang batas latensi (10 detik secara default).
  - Kueri tertentu dapat dilakukan lebih efisien menggunakan struktur data ElastiCache for Redis. Sebagai contoh, untuk pencarian rentang gaya numerik, aplikasi dapat mengimplementasikan indeks numerik sederhana dengan Set Berurut. Mengelola indeks ini dapat mengurangi pemindaian yang dilakukan pada set data, dan mengembalikan data dengan efisiensi kinerja yang lebih besar.
  - Untuk beban kerja ElastiCache for Redis, `redis-benchmark` menyediakan antarmuka sederhana untuk menguji kinerja perintah yang berbeda menggunakan input yang ditentukan pengguna seperti jumlah klien, dan ukuran data.
  - Karena Memcached hanya mendukung perintah tingkat kunci sederhana, pertimbangkan untuk membangun kunci tambahan sebagai indeks untuk menghindari iterasi melalui ruang kunci untuk melayani kueri klien.
- [Sumber Daya]:
  - Memantau penggunaan dengan Metrik CloudWatch
  - Memantau penggunaan dengan Metrik CloudWatch

- Menggunakan alarm Amazon CloudWatch
- Parameter khusus Redis
- SLOWLOG
- Tolok ukur Redis

## PE 2: Bagaimana cara mendistribusikan pekerjaan di seluruh Simpul klaster ElastiCache Anda?

Pengenalan tingkat pertanyaan: Cara aplikasi Anda tersambung ke simpul Amazon ElastiCache dapat memengaruhi kinerja dan skalabilitas klaster.

Manfaat tingkat pertanyaan: Memanfaatkan simpul yang tersedia di klaster dengan tepat akan memastikan bahwa pekerjaan terdistribusi ke seluruh sumber daya yang tersedia. Teknik-teknik berikut juga membantu menghindari sumber daya yang menganggur.

- [Wajib] Minta klien tersambung ke titik akhir ElastiCache yang tepat.
  - Amazon ElastiCache for Redis mengimplementasikan titik akhir yang berbeda berdasarkan mode klaster yang digunakan. Untuk mode klaster diaktifkan, ElastiCache akan menyediakan titik akhir konfigurasi. Untuk mode cluster dinonaktifkan, ElastiCache menyediakan titik akhir primer, biasanya digunakan untuk menulis, dan titik akhir pembaca untuk menyeimbangkan pembacaan di seluruh replika. Menerapkan titik akhir ini dengan benar akan menghasilkan kinerja yang lebih baik, dan operasi penskalaan yang lebih mudah. Hindari menghubungkan ke titik akhir simpul individu kecuali ada persyaratan khusus untuk melakukannya.
  - Untuk klaster Memcached multisimpul, ElastiCache menyediakan titik akhir konfigurasi yang memungkinkan Penemuan Otomatis. Disarankan untuk menggunakan algoritma hashing untuk mendistribusikan pekerjaan secara merata di seluruh simpul cache. Banyak pustaka klien Memcached menerapkan hashing yang konsisten. Periksa dokumentasi untuk pustaka yang Anda gunakan untuk melihat apakah mendukung hashing yang konsisten dan cara menerapkannya. Anda dapat menemukan informasi selengkapnya tentang penerapan fitur-fitur ini di sini.
- [Lebih baik] Manfaatkan mode klaster ElastiCache for Radis untuk meningkatkan skalabilitas.
  - Klaster ElastiCache for Redis (mode klaster diaktifkan) mendukung operasi penskalaan online (keluar/masuk dan naik/turun) untuk membantu mendistribusikan data secara dinamis di seluruh serpihan. Menggunakan Titik Akhir Konfigurasi akan memastikan klaster Anda yang sadar klien dapat menyesuaikan perubahan dalam topologi klaster.

  - Anda juga dapat menyeimbangkan kembali klaster dengan memindahkan hashslots di antara serpihan yang tersedia di klaster ElastiCache for Redis (mode klaster diaktifkan). Tindakan ini membantu mendistribusikan pekerjaan secara lebih efisien di seluruh serpihan yang tersedia.
- [Lebih baik] Terapkan strategi untuk mengidentifikasi dan memulihkan kunci sibuk dalam beban kerja Anda.
  - Pertimbangkan dampak struktur data Redis multi-dimensi seperti daftar, aliran, set, dll. Struktur data tersebut disimpan dalam Kunci Redis tunggal, yang berada pada satu simpul. Kunci multi-dimensi yang sangat besar memiliki potensi untuk memanfaatkan lebih banyak kapasitas jaringan dan memori daripada jenis data lainnya dan dapat menyebabkan penggunaan simpul yang tidak proporsional. Jika memungkinkan, rancang beban kerja Anda untuk menyebarkan akses data di banyak Kunci diskrit.
  - kunci sibuk dalam beban kerja dapat memengaruhi kinerja simpul yang digunakan. Untuk beban kerja ElastiCache for Redis, Anda dapat mendeteksi kunci sibuk menggunakan `redis-cli --hotkeys` jika kebijakan LFU max-memory diberlakukan.
  - Pertimbangkan untuk mereplikasi kunci sibuk di beberapa simpul untuk mendistribusikan akses ke simpul secara lebih merata. Pendekatan ini mengharuskan klien untuk menulis ke beberapa simpul primer (simpul Redis itu sendiri tidak akan menyediakan fungsi ini) dan untuk mempertahankan daftar nama kunci yang dibaca, selain nama kunci asli.
  - ElastiCache for Redis versi 6 mendukung caching sisi klien yang dibantu server. Hal ini memungkinkan aplikasi untuk menunggu perubahan pada kunci sebelum membuat jaringan memanggil kembali ke ElastiCache.
- [Sumber Daya]:
  - Mengonfigurasi Amazon ElastiCache for Redis untuk ketersediaan yang lebih tinggi
  - Menemukan titik akhir koneksi
  - Praktik terbaik penyeimbangan beban
  - Resharding dan penyeimbangan ulang serpihan secara online untuk Redis (mode klaster diaktifkan)
  - Caching sisi klien di Redis

## PE 3: Untuk beban kerja caching, bagaimana cara melacak dan melaporkan efektivitas dan kinerja cache Anda?

Pengenalan tingkat pertanyaan: Caching adalah beban kerja yang umum ditemui di ElastiCache dan penting bagi Anda untuk memahami cara mengelola efektivitas dan kinerja cache Anda.

Manfaat tingkat pertanyaan: Aplikasi Anda mungkin menunjukkan tanda-tanda kinerja yang lamban. Kemampuan Anda untuk menggunakan metrik khusus cache untuk menginformasikan keputusan Anda tentang cara meningkatkan kinerja aplikasi sangat penting untuk beban kerja cache Anda.

- [Wajib] Ukur dan lacak dari waktu ke waktu rasio hit cache. Efisiensi cache Anda ditentukan oleh 'rasio hit cache'. Rasio hit cache ditentukan oleh total hit kunci dibagi dengan total hit dan gagal. Semakin dekat ke 1 rasionya, semakin efektif cache Anda. Rasio hit cache yang rendah disebabkan oleh volume cache yang gagal. Kegagalan cache terjadi ketika kunci yang diminta tidak ditemukan di cache. Kunci tidak ada dalam cache karena telah dikosongkan atau dihapus, telah kedaluwarsa, atau tidak pernah ada. Pahami mengapa kunci tidak ada dalam cache dan kembangkan strategi yang tepat untuk menyimpan kunci dalam cache.

  [Sumber Daya]:

  - Metrik untuk Redis
- [Wajib] Ukur dan kumpulkan kinerja cache aplikasi Anda bersamaan dengan nilai latensi dan pemanfaatan CPU untuk memahami apakah Anda perlu melakukan penyesuaian pada waktu tayang atau komponen aplikasi lainnya. ElastiCache menyediakan satu set metrik CloudWatch untuk latensi agregat untuk setiap struktur data. Metrik latensi ini dihitung menggunakan statistik commandstats dari perintah INFO ElastiCache for Redis dan tidak menyertakan jaringan dan waktu I/O. Hanya ini waktu yang dihabiskan oleh ElastiCache for Redis untuk memproses operasi.

  [Sumber Daya]:

  - Metrik untuk Redis
  - Praktik terbaik pemantauan dengan Amazon ElastiCache for Redis menggunakan Amazon CloudWatch
- [Terbaik] Pilih strategi caching yang tepat untuk kebutuhan Anda. Rasio hit cache yang rendah disebabkan oleh volume cache yang gagal. Jika beban kerja Anda dirancang untuk memiliki volume kegagalan cache yang rendah (seperti komunikasi waktu nyata), tindakan terbaik adalah melakukan tinjauan strategi caching dan menerapkan resolusi yang paling tepat untuk beban kerja Anda, seperti instrumentasi kueri untuk mengukur memori dan kinerja. Strategi aktual yang Anda gunakan mengisi dan memelihara cache Anda akan tergantung pada data apa yang perlu di-cache oleh klien dan pola akses ke data tersebut. Misalnya, Anda cenderung tidak akan menggunakan strategi yang sama untuk rekomendasi yang dipersonalisasi pada aplikasi streaming, dan untuk berita yang sedang tren.

  [Sumber Daya]:

  - Strategi Pembuatan Cache

- Praktik Terbaik Caching
- Kinerja pada Skala dengan Whitepaper Amazon ElastiCache

## PE 4: Bagaimana beban kerja Anda mengoptimalkan penggunaan sumber daya dan koneksi jaringan?

Pengenalan tingkat pertanyaan: ElastiCache for Redis dan Memcached didukung oleh banyak klien aplikasi, dan implementasinya dapat bervariasi. Anda perlu memahami jaringan dan manajemen koneksi di tempat untuk menganalisis potensi dampak kinerja.

Manfaat tingkat pertanyaan: Penggunaan sumber daya jaringan yang efisien dapat meningkatkan efisiensi kinerja klaster Anda. Rekomendasi berikut dapat mengurangi tuntutan jaringan, dan meningkatkan latensi dan throughput klaster.

- [Wajib] Kelola koneksi secara aktif ke klaster ElastiCache Anda.
  - Pengumpulan koneksi dalam aplikasi mengurangi jumlah overhead pada klaster yang dibuat dengan membuka dan menutup koneksi. Pantau perilaku koneksi di Amazon CloudWatch menggunakan `CurrConnections` dan `NewConnections`.
  - Hindari kebocoran koneksi dengan menutup koneksi klien dengan benar sesuai keperluan. Strategi manajemen koneksi termasuk menutup koneksi yang tidak digunakan dengan benar, dan pengaturan batas waktu koneksi.
  - Untuk beban kerja Memcached, ada jumlah memori yang dapat dikonfigurasi yang disediakan untuk menangani koneksi yang disebut, `memcached_connections_overhead`.
- [Lebih baik] Kompres objek besar untuk mengurangi memori, dan meningkatkan throughput jaringan.
  - Kompresi data dapat mengurangi jumlah throughput jaringan yang diperlukan (Gbps), tetapi meningkatkan jumlah pekerjaan pada aplikasi untuk mengompres dan mendekompresi data.
  - Kompresi juga mengurangi jumlah memori yang dikonsumsi oleh kunci
  - Berdasarkan kebutuhan aplikasi Anda, pertimbangkan pertukaran antara rasio kompresi dan kecepatan kompresi.
- [Sumber Daya]:
  - Amazon ElastiCache for Redis - Penyimpanan Data Global
  - Parameter spesifik Memcached
  - Amazon ElastiCache for Redis 5.0.3 meningkatkan penanganan I/O untuk meningkatkan kinerja

- Metrik untuk Redis
- Mengonfigurasi Amazon ElastiCache for Redis untuk ketersediaan yang lebih tinggi

## PE 5: Bagaimana cara mengelola penghapusan kunci dan/atau pengosongan?

Pengenalan tingkat pertanyaan: Beban kerja memiliki persyaratan yang berbeda, dan perilaku yang diharapkan saat simpul klaster mendekati batas konsumsi memori. Amazon ElastiCache for Redis memiliki kebijakan yang berbeda untuk menangani situasi tersebut.

Manfaat tingkat pertanyaan: Manajemen memori yang tersedia yang tepat, dan pemahaman tentang kebijakan pengosongan akan membantu memastikan kesadaran perilaku klaster ketika batas memori instans terlampaui.

- [Wajib] Instrumen akses data untuk mengevaluasi kebijakan mana yang akan diterapkan. Identifikasi kebijakan max-memory yang sesuai untuk mengontrol apakah dan bagaimana pengosongan dilakukan di klaster.
  - Pengosongan terjadi ketika memori maksimum pada klaster dikonsumsi dan kebijakan diberlakukan untuk memungkinkan pengosongan. Perilaku klaster dalam situasi ini tergantung pada kebijakan pengosongan yang ditentukan. Kebijakan ini dapat dikelola menggunakan `maxmemory-policy` pada grup parameter klaster ElastiCache for Redis.
  - Kebijakan default `volatile-lru` membebaskan memori dengan mengosongkan kunci dengan set waktu kedaluwarsa (nilai TTL). Kebijakan yang paling jarang digunakan (LFU) dan yang paling lama tidak digunakan (LRU) menghapus kunci berdasarkan penggunaan.
  - Untuk beban kerja Memcached, ada kebijakan LRU default yang mengendalikan pengosongan di setiap simpul. Jumlah pengosongan di klaster Amazon ElastiCache Anda dapat dipantau menggunakan metrik Pengosongan di Amazon CloudWatch.
- [Lebih Baik] Lakukan standarisasi perilaku penghapusan untuk mengontrol dampak kinerja pada klaster Anda untuk menghindari kemacetan kinerja yang tidak terduga.
  - Untuk beban kerja ElastiCache for Redis, saat secara eksplisit menghapus kunci dari klaster, `UNLINK` seperti `DEL`: itu menghapus kunci yang ditentukan. Namun, perintah melakukan klaim ulang memori aktual di thread yang berbeda, sehingga tidak memblokir, sementara `DEL` memblokir. Penghapusan yang sebenarnya akan terjadi nanti secara asinkron.
  - Untuk beban kerja ElastiCache for Redis 6.x, perilaku perintah `DEL` dapat dimodifikasi dalam grup parameter menggunakan parameter `lazyfree-lazy-user-del`.
- [Sumber Daya]:

- Mengonfigurasi parameter mesin menggunakan grup parameter
- UNLINK
- Manajemen Keuangan Cloud dengan AWS

## PE 6: Bagaimana Anda memodelkan dan berinteraksi dengan data di ElastiCache?

Pengenalan tingkat pertanyaan: ElastiCache sangat bergantung pada struktur data dan model data yang digunakan, tetapi juga perlu mempertimbangkan penyimpanan data yang mendasarinya (jika ada). Pahami struktur data ElastiCache for Redis yang tersedia dan pastikan Anda menggunakan struktur data yang paling tepat untuk kebutuhan Anda.

Manfaat tingkat pertanyaan: Pemodelan data di ElastiCache memiliki beberapa lapisan, termasuk kasus penggunaan aplikasi, jenis data, dan hubungan antar elemen data. Selain itu, setiap jenis data dan perintah ElastiCache for Redis memiliki kinerja khusus mereka sendiri yang terdokumentasi dengan baik.

- [Terbaik] Praktik terbaiknya adalah mengurangi penimpaan data yang tidak disengaja. Gunakan konvensi penamaan yang meminimalkan tumpang tindihnya nama kunci. Penamaan konvensional struktur data Anda menggunakan metode hierarkis seperti: `APPNAME:CONTEXT:ID`, dan juga `ORDER-APP:CUSTOMER:123`.

  [Sumber Daya]:

  - Penamaan kunci
- [Terbaik] Perintah ElastiCache for Redis memiliki kompleksitas waktu yang ditentukan oleh notasi “Big O”. Kompleksitas waktu perintah ini adalah representasi algoritmik/matematis dari dampaknya. Saat memperkenalkan jenis data baru dalam aplikasi, Anda perlu meninjau kompleksitas waktu perintah terkait dengan cermat. Perintah dengan kompleksitas waktu O(1) bersifat konstan dalam waktu dan tidak bergantung pada ukuran input, namun perintah dengan kompleksitas waktu O(N) bersifat linier dalam waktu dan menyesuaikan ukuran input. Karena desain thread tunggal ElastiCache for Redis, volume besar operasi kompleksitas waktu yang tinggi akan menghasilkan kinerja yang lebih rendah dan potensi waktu habis saat operasi.

  [Sumber Daya]:

  - Perintah
- [Terbaik] Gunakan API untuk mendapatkan visibilitas GUI ke dalam model data di klaster Anda.

  [Sumber Daya]:

- Redis Commander
- Redis Browser
- Redsmin

## PE 7: Bagaimana cara log perintah yang berjalan lambat di klaster Amazon ElastiCache Anda?

Pengenalan tingkat pertanyaan: Manfaat penyetelan kinerja melalui pengambilan, agregasi, dan notifikasi perintah yang berjalan lama. Dengan memahami berapa lama waktu yang dibutuhkan untuk menjalankan perintah, Anda dapat menentukan perintah mana yang menghasilkan kinerja yang buruk serta perintah yang menghalangi kinerja optimal mesin. Amazon ElastiCache for Redis juga memiliki kemampuan untuk meneruskan informasi ini ke Amazon CloudWatch atau Amazon Kinesis Data Firehose.

Manfaat tingkat pertanyaan: Pencatatan log ke lokasi permanen khusus dan menyediakan peristiwa notifikasi untuk perintah lambat dapat membantu analisis kinerja terperinci dan dapat digunakan untuk memicu peristiwa otomatis.

- [Wajib]  Amazon ElastiCache for Redis menjalankan mesin versi 6.0 atau yang lebih baru, grup parameter yang dikonfigurasi dengan benar dan pencatatan SLOWLOG diaktifkan di klaster.
  - Parameter yang diperlukan hanya tersedia ketika kompatibilitas versi mesin diatur ke Redis versi 6.0 atau yang lebih tinggi.
  - Pencatatan log SLOWLOG terjadi ketika waktu eksekusi server dari suatu perintah membutuhkan waktu lebih lama dari nilai yang ditentukan. Perilaku klaster tergantung pada parameter Grup Parameter terkait yaitu `slowlog-log-slower-than` dan `slowlog-max-len`.
  - Perubahan akan diterapkan segera.
- [Terbaik] Manfaatkan kemampuan CloudWatch atau Kinesis Data Firehose.
  - Gunakan kemampuan pemfilteran dan alarm CloudWatch, Wawasan Log CloudWatch, dan Amazon Simple Notification Services untuk mencapai pemantauan performa dan notifikasi peristiwa.
  - Gunakan kemampuan streaming Kinesis Data Firehose untuk mengarsipkan log SLOWLOG ke penyimpanan permanen atau untuk memicu penyetelan parameter klaster otomatis.
  - Tentukan apakah JSON atau format TEXT biasa yang paling sesuai dengan kebutuhan Anda.
  - Berikan izin IAM untuk menerbitkan ke CloudWatch atau Kinesis Data Firehose.
- [Lebih baik] Konfigurasikan `slowlog-log-slower-than` ke nilai selain default.

  - Parameter ini menentukan berapa lama perintah dapat dijalankan dalam mesin Redis sebelum dicatat sebagai perintah berjalan lambat. Nilai defaultnya adalah 10.000 mikrodetik (10 milidetik). Nilai default mungkin terlalu tinggi untuk beberapa beban kerja.
  - Tentukan nilai yang lebih sesuai untuk beban kerja Anda berdasarkan kebutuhan aplikasi dan hasil pengujian; namun, nilai yang terlalu rendah dapat menghasilkan data yang berlebihan.
- [Lebih baik] Biarkan `slowlog-max-len` pada nilai default.
  - Parameter ini menentukan batas atas untuk jumlah perintah yang berjalan lambat yang ditangkap dalam memori Redis pada waktu tertentu. Nilai 0 secara efektif menonaktifkan penangkapan. Semakin tinggi nilainya, semakin banyak entri yang akan disimpan dalam memori, mengurangi kemungkinan informasi penting dihapus sebelum dapat ditinjau. Nilai default-nya adalah 128.
  - Nilai default tersebut sesuai untuk sebagian besar beban kerja. Jika ada kebutuhan untuk menganalisis data dalam jendela waktu yang diperluas dari redis-cli melalui perintah SLOWLOG, pertimbangkan untuk meningkatkan nilai ini. Itu memungkinkan lebih banyak perintah untuk tetap disimpan di memori Redis.

    Jika Anda memancarkan data SLOWLOG ke CloudWatch Logs atau Kinesis Data Firehose, data akan dipertahankan dan dapat dianalisis di luar sistem ElastiCache, mengurangi kebutuhan untuk menyimpan sejumlah besar perintah yang berjalan lambat di memori Redis.
- [Sumber Daya]:
  - Bagaimana cara mengaktifkan log Slow Redis di klaster cache ElastiCache for Redis?
  - Pengiriman log
  - Parameter khusus Redis
  - https://aws.amazon.com/cloudwatch/Amazon CloudWatch
  - Amazon Kinesis Data Firehose

## PE8: Bagaimana Penskalaan Otomatis membantu meningkatkan kinerja klaster ElastiCache?

Pengenalan tingkat pertanyaan: Dengan menerapkan fitur penskalaan otomatis Redis, komponen ElastiCache Anda dapat menyesuaikan dari waktu ke waktu untuk menambah atau mengurangi serpihan atau replika yang diinginkan secara otomatis. Ini dapat dilakukan dengan menerapkan kebijakan pelacakan target atau penskalaan terjadwal.

Manfaat tingkat pertanyaan: Memahami dan merencanakan lonjakan beban kerja dapat memastikan peningkatan kinerja caching dan kelangsungan bisnis. Penskalaan Otomatis ElastiCache for Redis terus memantau pemanfaatan CPU/memori Anda untuk memastikan klaster Anda beroperasi pada tingkat kinerja yang Anda inginkan.

- [Wajib] Saat meluncurkan klaster untuk ElastiCache for Redis:

  1. Pastikan mode Klaster diaktifkan
  2. Pastikan instans berasal dari keluarga jenis dan ukuran tertentu yang mendukung penskalaan otomatis
  3. Pastikan klaster tidak berjalan di penyimpanan data Global, Outposts atau Zona Lokal

  [Sumber Daya]:

  - Penskalaan klaster di Redis (Mode Cluster Diaktifkan)
  - Menggunakan Penskalaan Otomatis dengan serpihan
  - Menggunakan Penskalaan Otomatis dengan replika
- [Terbaik] Identifikasi apakah beban kerja Anda berat baca atau berat tulis untuk menentukan kebijakan penskalaan. Untuk kinerja terbaik, gunakan hanya satu metrik pelacakan. Disarankan untuk menghindari beberapa kebijakan untuk setiap dimensi, karena kebijakan penskalaan otomatis akan menskalakan keluar saat target tercapai, namun menskalakan masuk hanya jika semua kebijakan pelacakan target siap untuk diskalakan.

  [Sumber Daya]:

  - Kebijakan Penskalaan Otomatis
  - Menetapkan kebijakan penskalaan
- [Terbaik] Memantau kinerja dari waktu ke waktu dapat membantu Anda mendeteksi perubahan beban kerja tidak disadari jika dipantau pada titik waktu tertentu. Anda dapat menganalisis metrik CloudWatch yang sesuai untuk pemanfaatan klaster selama periode empat minggu untuk menentukan ambang nilai target. Jika Anda masih tidak yakin nilai apa yang harus dipilih, sebaiknya mulai dengan nilai metrik standar minimum yang didukung.

  [Sumber Daya]:

  - Memantau penggunaan dengan Metrik CloudWatch
- [Lebih baik] Kami menyarankan untuk menguji aplikasi Anda dengan beban kerja minimum dan maksimum yang diharapkan, guna mengidentifikasi jumlah pasti serpihan/replika yang diperlukan klaster untuk mengembangkan kebijakan penskalaan dan mengurangi masalah ketersediaan.

[Sumber Daya]:

- Mendaftarkan Target yang Dapat Diskalakan
- Mendaftarkan Target yang Dapat Diskalakan

## Pilar Optimasi Biaya Lensa Amazon ElastiCache Well-Architected

Pilar optimasi biaya berfokus pada menghindari biaya yang tidak perlu. Topik utama termasuk memahami dan mengendalikan di mana uang dihabiskan, memilih jenis simpul yang paling tepat (menggunakan instans yang mendukung tingkatan data berdasarkan kebutuhan beban kerja), jumlah jenis sumber daya yang tepat (berapa banyak replika baca), menganalisis pengeluaran dari waktu ke waktu, dan penskalaan untuk memenuhi kebutuhan bisnis tanpa pengeluaran berlebihan.

Topik

- BIAYA 1: Bagaimana Anda mengidentifikasi dan memantau biaya yang terkait dengan sumber daya ElastiCache Anda? Bagaimana Anda mengembangkan mekanisme untuk memungkinkan pengguna membuat, mengelola, dan membuang sumber daya yang dibuat?
- BIAYA 2: Bagaimana Anda menggunakan alat pemantauan berkelanjutan untuk membantu Anda mengoptimalkan biaya yang terkait dengan sumber daya ElastiCache Anda?
- BIAYA 3: Haruskah Anda menggunakan tipe instans yang mendukung tingkatan data? Apa keuntungan dari tingkatan data? Kapan sebaiknya tidak menggunakan instans tingkatan data?

### BIAYA 1: Bagaimana Anda mengidentifikasi dan memantau biaya yang terkait dengan sumber daya ElastiCache Anda? Bagaimana Anda mengembangkan mekanisme untuk memungkinkan pengguna membuat, mengelola, dan membuang sumber daya yang dibuat?

Pengenalan tingkat pertanyaan: Memahami metrik biaya memerlukan partisipasi dan kolaborasi di berbagai tim: rekayasa perangkat lunak, manajemen data, pemilik produk, keuangan, dan pimpinan. Mengidentifikasi penggerak biaya utama mengharuskan semua pihak yang terlibat memahami tuas kontrol penggunaan layanan dan kompromi manajemen biaya. Hal ini juga seringkali menjadi perbedaan utama antara upaya pengoptimalan biaya yang berhasil dan kurang berhasil. Memastikan Anda memiliki proses dan alat untuk memantau sumber daya yang dibuat dari pengembangan hingga tahap produksi dan penghentian akan membantu Anda mengelola biaya yang terkait dengan ElastiCache.

Pertanyaan terkait manfaat: Pemantauan berkelanjutan pada semua biaya yang terkait dengan beban kerja Anda memerlukan pemahaman mendalam tentang arsitektur yang mencakup ElastiCache sebagai salah satu komponennya. Selain itu, Anda harus memiliki rencana manajemen biaya untuk mengumpulkan dan membandingkan penggunaan dengan anggaran Anda.

- [Required] Latih Pusat Keunggulan Cloud (CCoE) dengan salah satu charter pendiriannya untuk menentukan, memantau, dan mengambil tindakan pada metrik seputar penggunaan ElastiCache organisasi Anda. Jika CCoE ada dan berfungsi, pastikan ia tahu cara membaca dan melacak biaya yang terkait dengan ElastiCache. Saat sumber daya dibuat, gunakan peran dan kebijakan IAM untuk memvalidasi bahwa hanya tim dan grup tertentu yang dapat membuat instans sumber daya. Hal ini memastikan bahwa biaya yang terkait dengan hasil bisnis dan garis akuntabilitas yang jelas ditetapkan, dari perspektif biaya.

  1. CCoE harus mengidentifikasi, mendefinisikan, dan memublikasikan metrik biaya yang diperbarui secara rutin-bulanan seputar penggunaan ElastiCache utama di seluruh data kategoris seperti:
     a. Jenis simpul yang digunakan dan atributnya: standar vs. memori yang dioptimalkan, sesuai permintaan vs. instans cadangan, wilayah, dan zona ketersediaan
     b. Jenis lingkungan: gratis, developer, pengujian, dan produksi
     c. Strategi penyimpanan dan retensi cadangan
     d. Transfer data di dalam dan lintas wilayah
     e. Instans berjalan di Amazon Outposts
  2. CCoE terdiri dari tim lintas fungsi dengan representasi non-eksklusif dari rekayasa perangkat lunak, manajemen data, tim produk, keuangan, dan tim kepemimpinan di organisasi Anda.

  [Resources]:

  - Buat Cloud Center of Excellence
  - Harga Amazon ElastiCache

- [Required] Gunakan tag alokasi biaya untuk memantau biaya pada tingkat detail yang rendah. Gunakan Manajemen Biaya AWS untuk memvisualisasikan, memahami, dan mengelola biaya AWS dan penggunaan Anda dari waktu ke waktu.

  1. Gunakan tanda untuk mengatur sumber daya Anda, dan tanda alokasi biaya untuk melacak biaya AWS Anda secara mendetail. Setelah Anda mengaktifkan tag alokasi biaya, AWS menggunakan tanda alokasi biaya untuk mengatur biaya sumber daya Anda di laporan alokasi biaya Anda untuk memudahkan Anda mengategorikan dan melacak biaya AWS Anda. AWS menyediakan dua jenis tanda alokasi biaya, tanda yang dihasilkan AWS dan tag yang ditentukan pengguna. AWS mendefinisikan, membuat, dan menerapkan tanda yang dihasilkan AWS untuk

Anda, dan Anda menentukan, membuat, dan menerapkan tanda yang ditentukan pengguna. Anda harus mengaktifkan kedua jenis tanda secara terpisah sebelum tanda tersebut muncul di Manajemen Biaya atau laporan alokasi biaya.

2. Gunakan tanda alokasi biaya untuk mengatur tagihan AWS guna mencerminkan struktur biaya Anda sendiri. Ketika Anda menambahkan tanda alokasi biaya ke sumber daya Anda di Amazon ElastiCache, Anda dapat melacak biaya dengan mengelompokkan pengeluaran pada faktur Anda berdasarkan nilai tanda sumber daya. Anda harus mempertimbangkan menggabungkan tanda untuk melacak biaya dengan tingkat detail yang lebih besar.

[Resources]:

- Menggunakan tanda alokasi biaya AWS
- Memantau biaya dengan tanda alokasi biaya
- AWS Cost Explorer

- [Best] Hubungkan biaya ElastiCache ke metrik yang menjangkau seluruh organisasi.

1. Pertimbangkan metrik bisnis serta metrik operasional seperti latensi - konsep apa dalam model bisnis Anda yang dapat dimengerti di seluruh peran? Metrik perlu dipahami oleh peran dalam organisasi sebanyak mungkin.
2. Contoh - pengguna yang dilayani secara simultan, latensi maks serta rata-rata per operasi dan pengguna, skor keterlibatan pengguna, tingkat pengembalian pengguna/minggu, panjang sesi/pengguna, tingkat pengabaian, laju hit cache, dan kunci yang dilacak

[Resources]:

- Memantau penggunaan dengan Metrik CloudWatch

- [Good]Pertahankan visibilitas arsitektur dan operasional terkini pada metrik dan biaya di seluruh beban kerja yang menggunakan ElastiCache.

1. Memahami seluruh ekosistem solusi Anda, ElastiCache cenderung menjadi bagian dari ekosistem AWS layanan lengkap dalam rangkaian teknologinya, dari klien hingga API Gateway, Redshift, dan QuickSight untuk alat pelaporan (misalnya).
2. Petakan komponen solusi Anda dari klien, koneksi, keamanan, operasi dalam memori, penyimpanan, otomatisasi sumber daya, akses, dan manajemen data, di diagram arsitektur Anda. Setiap lapisan terhubung ke seluruh solusi serta memiliki kebutuhan dan kemampuan sendiri yang menambah dan/atau membantu Anda mengelola biaya keseluruhan.
3. Diagram Anda harus mencakup penggunaan komputasi, jaringan, penyimpanan, kebijakan siklus hidup, pengumpulan metrik serta elemen ElastiCache operasional dan fungsional aplikasi Anda

4. Persyaratan beban kerja Anda cenderung berkembang dari waktu ke waktu dan penting bagi Anda untuk terus memelihara dan mendokumentasikan pemahaman Anda tentang komponen yang mendasarinya serta tujuan fungsional utama Anda agar tetap proaktif dalam manajemen biaya beban kerja Anda.
5. Dukungan eksekutif untuk visibilitas, akuntabilitas, prioritas, dan sumber daya sangat penting bagi Anda untuk memiliki strategi manajemen biaya yang efektif untuk ElastiCache Anda.

## BIAYA 2: Bagaimana Anda menggunakan alat pemantauan berkelanjutan untuk membantu Anda mengoptimalkan biaya yang terkait dengan sumber daya ElastiCache Anda?

Pertanyaan pengenalan: Anda perlu menargetkan keseimbangan yang tepat antara biaya ElastiCache dan metrik performa aplikasi Anda. Amazon CloudWatch menyediakan visibilitas ke dalam metrik operasional utama yang dapat membantu Anda menilai apakah sumber daya ElastiCache Anda terlalu banyak atau kurang digunakan, dibandingkan terhadap kebutuhan Anda. Dari perspektif optimasi biaya, Anda perlu memahami kapan penyediaan Anda berlebih dan dapat mengembangkan mekanisme yang tepat untuk mengubah ukuran sumber daya ElastiCache Anda sambil mempertahankan kebutuhan operasional, ketersediaan, ketahanan, dan kinerja Anda.

Pertanyaan terkait manfaat: Dalam keadaan yang ideal, Anda akan menyediakan sumber daya yang cukup untuk memenuhi kebutuhan operasional beban kerja Anda dan tidak memiliki sumber daya yang kurang dimanfaatkan yang dapat berakibat pada keadaan biaya yang kurang optimal. Anda harus dapat mengidentifikasi dan menghindari pengoperasian sumber daya ElastiCache yang terlalu besar untuk jangka waktu yang lama.

- [Required] Gunakan CloudWatch untuk memantau klaster ElastiCache Anda dan menganalisis bagaimana metrik ini berhubungan dengan dasbor Cost Explorer Anda. AWS
  1. ElastiCache menyediakan metrik tingkat host (misalnya, penggunaan CPU) dan metrik yang khusus untuk perangkat lunak mesin cache (misalnya, pengambilan cache dan kesalahan cache). Metrik ini diukur dan diterbitkan untuk setiap simpul cache dalam interval waktu 60 detik.
  2. Metrik performa ElastiCache (CPUUtilization, EngineUtilization, SwapUsage, CurrConnections, dan Evictions) dapat menunjukkan bahwa Anda perlu menaikkan/menurunkan skala (gunakan tipe simpul cache yang lebih besar/lebih kecil) atau melakukan penskalaan ke dalam/ke keluar (tambahkan lebih banyak/lebih sedikit serpihan). Pahami implikasi biaya dari keputusan penskalaan dengan membuat matriks buku pedoman yang memperkirakan biaya tambahan dan lama waktu min dan maks yang diperlukan untuk memenuhi ambang kinerja aplikasi Anda.

[Resources]:

- Memantau penggunaan dengan Metrik CloudWatch
- Metrik Apa yang Harus Dipantau?
- Harga Amazon ElastiCache

- [Required] Memahami dan mendokumentasikan strategi cadangan dan implikasi biaya Anda.
  1. Dengan ElastiCache, cadangan disimpan di Amazon S3, yang menyediakan penyimpanan yang tahan lama. Anda perlu memahami implikasi biaya dalam kaitannya dengan kemampuan Anda untuk pulih dari kegagalan.
  2. Aktifkan pencadangan otomatis yang akan menghapus file cadangan yang melewati batas retensi.

  [Resources]:

  - Menjadwalkan pencadangan otomatis
  - Harga Amazon Simple Storage Service
- [Best] Gunakan Simpul Cadangan untuk instans Anda sebagai strategi yang disengaja untuk mengelola biaya beban kerja yang dipahami dan didokumentasikan dengan baik. Simpul cadangan dikenakan biaya di muka dan bergantung pada jenis simpul dan lamanya reservasi—satu atau tiga tahun. Biaya ini jauh lebih kecil daripada biaya penggunaan per jam yang dikenakan untuk simpul sesuai permintaan.
  1. Anda mungkin perlu mengoperasikan klaster ElastiCache menggunakan simpul sesuai permintaan hingga Anda mengumpulkan data yang cukup untuk memperkirakan persyaratan instans yang dicadangkan. Rencanakan dan dokumentasikan sumber daya yang dibutuhkan untuk memenuhi kebutuhan Anda dan membandingkan biaya yang diharapkan di seluruh jenis instans (sesuai permintaan vs terpesan)
  2. Evaluasi secara rutin jenis simpul cache baru yang tersedia dan nilai apakah masuk akal, dari perspektif metrik biaya dan operasional, untuk memigrasikan armada instans Anda ke tipe simpul cache baru

## BIAYA 3: Haruskah Anda menggunakan tipe instans yang mendukung tingkatan data? Apa keuntungan dari tingkatan data? Kapan sebaiknya tidak menggunakan instans tingkatan data?

Pertanyaan pengenalan: Memilih jenis instans yang sesuai tidak hanya dapat memiliki dampak kinerja dan tingkat layanan tetapi juga dampak finansial. Jenis instans memiliki biaya berbeda yang

terkait dengannya. Memilih satu atau beberapa jenis instar besar yang dapat mengakomodasi semua kebutuhan penyimpanan dalam memori mungkin merupakan keputusan yang wajar. Namun, hal ini dapat memiliki dampak biaya yang signifikan saat proyek jatuh tempo. Memastikan bahwa jenis instans yang dipilih sudah benar memerlukan pemeriksaan berkala terhadap waktu idle objek ElastiCache.

Pertanyaan manfaat: Anda harus memiliki pemahaman yang jelas tentang bagaimana berbagai jenis instans memengaruhi biaya Anda saat ini dan di masa depan. Perubahan beban kerja marjinal atau periodik seharusnya tidak menyebabkan perubahan biaya yang tidak proporsional. Jika beban kerja mengizinkannya, jenis instans yang mendukung tingkatan data menawarkan harga yang lebih baik per penyimpanan yang tersedia. Karena instans penyimpanan data SSD yang tersedia, instans tingkatan data mendukung kapasitas total data per instance yang jauh lebih tinggi.

- [Required] Memahami batasan instans tingkatan data
  1. Hanya tersedia untuk klaster ElastiCache for Redis.
  2. Hanya tipe instans tertentu yang mendukung tingkatan data.
  3. Hanya ElastiCache for Redis versi 6.2 dan yang lebih baru yang didukung
  4. Item besar tidak ditukar ke SSD. Objek di atas 128 MiB disimpan dalam memori.

  [Resources]:
  - Tingkatan data
  - Harga Amazon ElastiCache
- [Required] Pahami berapa persentase basis data Anda yang secara teratur diakses oleh beban kerja Anda.
  1. Instans tingkatan data ideal untuk beban kerja yang sering mengakses sebagian kecil dari keseluruhan kumpulan data Anda tetapi masih memerlukan akses cepat ke data yang tersisa. Dengan kata lain, rasio data hot dan warm adalah sekitar 20:80.
  2. Kembangkan pemasangan level klaster dari waktu idle objek.
  3. Implementasi besar lebih dari 500 Gb data merupakan kandidat yang baik
- [Wajib] Memahami bahwa instans tingkatan data bukan opsional untuk beban kerja tertentu.
  1. Ada biaya kinerja kecil untuk mengakses objek yang jarang digunakan karena ditukar ke SSD lokal. Jika aplikasi Anda sensitif terhadap waktu respons, uji dampaknya terhadap beban kerja Anda.
  2. Tidak cocok untuk cache yang menyimpan sebagian besar objek berukuran lebih dari 128 MiB.

[Resources]:

- Batasan

- [Best] Jenis instans cadangan mendukung tingkatan data. Hal ini menjamin biaya terendah dalam hal jumlah penyimpanan data per instans.

  1. Anda mungkin perlu mengoperasikan klaster ElastiCache menggunakan instans tingkatan non-data hingga Anda memiliki pemahaman yang lebih baik tentang kebutuhan Anda.
  2. Analisis pola penggunaan data klaster ElastiCache Anda.
  3. Buat pekerjaan otomatis yang secara berkala mengumpulkan waktu idle objek.
  4. Jika Anda melihat bahwa persentase besar (sekitar 80%) objek menganggur untuk jangka waktu yang dianggap sesuai untuk beban kerja Anda, dokumentasikan temuan tersebut dan sarankan untuk memigrasikan klaster ke instans yang mendukung tingkatan data.
  5. Evaluasi secara rutin jenis simpul cache baru yang tersedia dan nilai apakah masuk akal, dari perspektif metrik biaya dan operasional, untuk memigrasikan armada instans Anda ke tipe simpul cache baru.

  [Resources]:

  - OBJECT IDLETIME
  - Harga Amazon ElastiCache

# Langkah-langkah pemecahan masalah umum dan praktik terbaik

Topik

- Masalah koneksi
- Kesalahan klien Redis
- Memecahkan masalah latensi tinggi di Tanpa Server ElastiCache
- Memecahkan masalah pembatasan di Tanpa Server ElastiCache
- Topik-Topik Terkait

## Masalah koneksi

Jika Anda tidak dapat terhubung ke ElastiCache cache Anda, pertimbangkan salah satu dari berikut ini:

1. Menggunakan TLS: Jika Anda mengalami koneksi macet saat mencoba terhubung ke ElastiCache titik akhir Anda, Anda mungkin tidak menggunakan TLS di klien Anda. Jika Anda menggunakan ElastiCache Tanpa Server, enkripsi dalam perjalanan selalu diaktifkan. Pastikan klien Anda menggunakan TLS untuk terhubung ke cache. Pelajari lebih lanjut tentang menghubungkan ke cache yang diaktifkan TLS di sini.
2. VPC: ElastiCache cache hanya dapat diakses dari dalam VPC. Pastikan bahwa instans EC2 dari mana Anda mengakses cache dan cache dibuat dalam VPC yang sama. ElastiCache Atau, Anda harus mengaktifkan peering VPC antara VPC tempat instans EC2 Anda berada dan VPC tempat Anda membuat cache.
3. Grup keamanan: ElastiCache menggunakan grup keamanan untuk mengontrol akses ke cache Anda. Pertimbangkan hal berikut:
   a. Pastikan bahwa grup keamanan yang digunakan oleh ElastiCache cache Anda memungkinkan akses masuk ke sana dari instans EC2 Anda. Lihat di sini untuk mempelajari cara mengatur aturan masuk di grup keamanan Anda dengan benar.
   b. Pastikan bahwa grup keamanan yang digunakan oleh ElastiCache cache Anda memungkinkan akses ke port cache Anda (6379 dan 6380 untuk tanpa server, dan 6379 secara default untuk dirancang sendiri). ElastiCache menggunakan port ini untuk menerima perintah Redis. Pelajari lebih lanjut tentang cara mengatur akses port di sini.

## Kesalahan klien Redis

ElastiCache Tanpa server hanya dapat diakses menggunakan klien Redis yang mendukung protokol mode cluster Redis. Cluster yang dirancang sendiri dapat diakses dari klien Redis dalam mode mana pun, tergantung pada konfigurasi cluster.

Jika Anda mengalami kesalahan Redis di klien Anda, pertimbangkan hal berikut:

1. Mode cluster: Jika Anda mengalami kesalahan atau kesalahan CROSSLOT dengan perintah SELECT Redis, Anda mungkin mencoba mengakses cache Cluster Mode Enabled dengan klien Redis yang tidak mendukung protokol Redis Cluster. ElastiCache Tanpa server hanya mendukung klien Redis yang mendukung protokol cluster Redis. Jika Anda ingin menggunakan Redis di “Cluster Mode Disabled” (CMD), maka Anda harus mendesain cluster Anda sendiri.
2. Kesalahan CROSSLOT: Jika Anda mengalami `ERR CROSSLOT Keys in request don't hash to the same slot` kesalahan, Anda mungkin mencoba mengakses kunci yang tidak termasuk dalam slot yang sama dalam cache mode Cluster. Sebagai pengingat, ElastiCache Serverless selalu beroperasi dalam Mode Cluster. Operasi multi-kunci, transaksi, atau skrip Lua

yang melibatkan beberapa kunci hanya diperbolehkan jika semua kunci yang terlibat berada dalam slot hash yang sama.

Untuk praktik terbaik tambahan seputar mengonfigurasi klien Redis, silakan tinjau posting blog ini.

## Memecahkan masalah latensi tinggi di Tanpa Server ElastiCache

Jika beban kerja Anda tampaknya mengalami latensi tinggi, Anda dapat menganalisis CloudWatch `SuccessfulReadRequestLatency` dan `SuccessfulWriteRequestLatency` metrik untuk memeriksa apakah latensi terkait dengan Tanpa Server. ElastiCache Metrik ini mengukur latensi yang internal ke ElastiCache Tanpa Server - latensi sisi klien dan waktu perjalanan jaringan antara klien Anda dan titik akhir Tanpa ElastiCache Server tidak disertakan.

Beberapa variabilitas dan lonjakan sesekali seharusnya tidak menjadi perhatian. Namun, jika `Average` statistik menunjukkan peningkatan tajam dan berlanjut, Anda harus memeriksa AWS Health Dashboard dan Dashboard Personal Health Anda untuk informasi lebih lanjut. Jika perlu, pertimbangkan untuk membuka kasus dukungan dengan AWS Support.

Pertimbangkan praktik dan strategi terbaik berikut untuk mengurangi latensi:

- Aktifkan Baca dari Replika: Jika aplikasi Anda mengizinkannya, sebaiknya aktifkan fitur “Baca dari Replika” di klien Redis Anda untuk menskalakan pembacaan dan mencapai latensi yang lebih rendah. Saat diaktifkan, ElastiCache Serverless mencoba merutekan permintaan baca Anda ke replika node cache yang berada di Availability Zone (AZ) yang sama dengan klien Anda, sehingga menghindari latensi jaringan lintas-AZ. Perhatikan, bahwa mengaktifkan fitur Baca dari Replika di klien Anda menandakan bahwa aplikasi Anda akhirnya menerima konsistensi data. Aplikasi Anda mungkin menerima data lama untuk beberapa waktu jika Anda mencoba membaca setelah menulis ke kunci.
- Pastikan aplikasi Anda di-deploy dalam AZ yang sama dengan cache Anda: Anda dapat mengamati latensi sisi klien yang lebih tinggi jika aplikasi Anda tidak di-deploy di AZ yang sama dengan cache Anda. Saat Anda membuat cache tanpa server, Anda dapat menyediakan subnet dari mana aplikasi Anda akan mengakses cache, dan ElastiCache Tanpa Server membuat VPC Endpoint di subnet tersebut. Pastikan aplikasi Anda disebarkan di AZ yang sama. Jika tidak, aplikasi Anda mungkin mengalami lompatan lintas-AZ saat mengakses cache yang menghasilkan latensi sisi klien yang lebih tinggi.
- Gunakan kembali koneksi: Permintaan ElastiCache tanpa server dibuat melalui koneksi TCP yang diaktifkan TLS menggunakan protokol RESP. Memulai koneksi (termasuk mengautentikasi koneksi,

jika dikonfigurasi) membutuhkan waktu sehingga latensi permintaan pertama lebih tinggi dari biasanya. Permintaan melalui koneksi yang sudah diinisialisasi memberikan ElastiCache latensi rendah yang konsisten. Untuk alasan ini, Anda harus mempertimbangkan untuk menggunakan penyatuan koneksi atau menggunakan kembali koneksi Redis yang ada.

- Kecepatan penskalaan: ElastiCache Tanpa server secara otomatis menskalakan seiring dengan bertambahnya tingkat permintaan Anda. Peningkatan besar secara tiba-tiba dalam tingkat permintaan, lebih cepat daripada kecepatan skala ElastiCache Tanpa Server, dapat mengakibatkan peningkatan latensi untuk beberapa waktu. ElastiCache Tanpa server biasanya dapat meningkatkan tingkat permintaan yang didukung dengan cepat, membutuhkan waktu hingga 10-12 menit untuk menggandakan tingkat permintaan.
- Periksa perintah yang berjalan lama: Beberapa perintah Redis, termasuk skrip Lua atau perintah pada struktur data besar, dapat berjalan untuk waktu yang lama. Untuk mengidentifikasi perintah ini, ElastiCache menerbitkan metrik tingkat perintah. Dengan ElastiCache Tanpa Server Anda dapat menggunakan metrik. `BasedECPUs`
- Permintaan Terbatas: Saat permintaan dibatasi di ElastiCache Tanpa Server, Anda mungkin mengalami peningkatan latensi sisi klien dalam aplikasi Anda. Saat permintaan dibatasi di ElastiCache Tanpa Server, Anda akan melihat peningkatan metrik Tanpa Server. `ThrottledRequests` ElastiCache Tinjau bagian di bawah ini untuk memecahkan masalah permintaan yang dibatasi.
- Distribusi kunci dan permintaan yang seragam: ElastiCache Untuk Redis, distribusi kunci atau permintaan per slot yang tidak merata dapat menghasilkan slot panas yang dapat mengakibatkan latensi tinggi. ElastiCache Tanpa server mendukung hingga 30.000 ECPUS/detik (90.000 ECPU/detik saat menggunakan Baca dari Replika) pada satu slot, dalam beban kerja yang menjalankan perintah SET/GET sederhana. Sebaiknya evaluasi distribusi kunci dan permintaan Anda di seluruh slot dan memastikan distribusi yang seragam jika tingkat permintaan Anda melebihi batas ini.

## Memecahkan masalah pembatasan di Tanpa Server ElastiCache

Dalam arsitektur berorientasi layanan dan sistem terdistribusi, membatasi kecepatan pemrosesan panggilan API oleh berbagai komponen layanan disebut throttling. Ini menghaluskan lonjakan, mengontrol ketidakcocokan dalam throughput komponen, dan memungkinkan pemulihan yang lebih dapat diprediksi ketika ada peristiwa operasional yang tidak terduga. ElastiCache Tanpa server dirancang untuk jenis arsitektur ini, dan sebagian besar klien Redis memiliki percobaan ulang bawaan untuk permintaan yang dibatasi. Throttling pada tingkat tertentu belum tentu menjadi masalah bagi aplikasi Anda, tetapi throttling yang terus-menerus pada bagian alur kerja data Anda

yang sensitif terhadap latensi dapat berdampak negatif terhadap pengalaman pengguna dan mengurangi efisiensi sistem secara keseluruhan.

Saat permintaan dibatasi di ElastiCache Tanpa Server, Anda akan melihat peningkatan metrik Tanpa Server. `ThrottledRequests` ElastiCache Jika Anda melihat sejumlah besar permintaan terbatas, pertimbangkan hal berikut:

- Kecepatan penskalaan: ElastiCache Tanpa server secara otomatis menskalakan saat Anda menelan lebih banyak data atau meningkatkan tingkat permintaan Anda. Jika aplikasi Anda menskalakan lebih cepat daripada kecepatan skala Tanpa Server, permintaan Anda mungkin terhambat saat ElastiCache Tanpa Server menskalakan untuk mengakomodasi beban kerja Anda. ElastiCache ElastiCache Tanpa server biasanya dapat meningkatkan ukuran penyimpanan dengan cepat, membutuhkan waktu hingga 10-12 menit untuk menggandakan ukuran penyimpanan di cache Anda.
- Distribusi kunci dan permintaan yang seragam: ElastiCache Untuk Redis, distribusi kunci atau permintaan per slot yang tidak merata dapat menghasilkan slot panas. Slot panas dapat mengakibatkan pembatasan permintaan jika tingkat permintaan ke satu slot melebihi 30.000 ECPU/detik, dalam beban kerja yang mengeksekusi perintah SET/GET sederhana.
- Baca dari Replika: Jika aplikasi Anda mengizinkannya, pertimbangkan untuk menggunakan fitur “Baca dari Replika”. Sebagian besar klien Redis dapat dikonfigurasi ke” pembacaan skala “untuk mengarahkan pembacaan ke node replika. Fitur ini memungkinkan Anda untuk menskalakan lalu lintas baca. Selain itu ElastiCache Tanpa Server secara otomatis merutekan pembacaan dari permintaan replika ke node di Availability Zone yang sama dengan aplikasi Anda sehingga latensi lebih rendah. Ketika Read from Replica diaktifkan, Anda dapat mencapai hingga 90.000 ECPU/detik pada satu slot, untuk beban kerja dengan perintah SET/GET sederhana.

## Topik-Topik Terkait

- Langkah pemecahan masalah tambahan
- the section called “Praktik terbaik dan strategi caching”

# Langkah pemecahan masalah tambahan

Item berikut harus diverifikasi saat memecahkan masalah konektivitas persisten dengan: ElastiCache

Topik

- Grup keamanan
- ACL jaringan
- Tabel rute
- Resolusi DNS
- Mengidentifikasi masalah dengan diagnostik sisi server
- Validasi konektivitas jaringan
- Batas terkait jaringan
- Penggunaan CPU
- Koneksi yang dihentikan dari sisi server
- Pemecahan masalah sisi klien untuk instans Amazon EC2
- Membedah waktu yang dibutuhkan untuk menyelesaikan satu permintaan tunggal

## Grup keamanan

Grup Keamanan adalah firewall virtual yang melindungi ElastiCache klien Anda (instans EC2, AWS Lambda fungsi, wadah Amazon ECS, dll.) dan cache. ElastiCache Grup keamanan bersifat stateful, artinya bahwa setelah lalu lintas masuk atau keluar diizinkan, maka tanggapan untuk lalu lintas itu akan secara otomatis mendapat otorisasi dalam konteks grup keamanan tertentu itu.

Fitur stateful mensyaratkan grup keamanan melacak semua koneksi yang diberikan otorisasi, dan terdapat batasan untuk koneksi dilacak. Jika batas itu tercapai, maka koneksi baru akan gagal. Silakan merujuk ke bagian pemecahan masalah untuk bantuan tentang cara mengidentifikasi apakah batas telah tercapai pada klien atau ElastiCache sisi.

Anda dapat memiliki satu grup keamanan yang ditetapkan pada saat yang sama ke klien dan ElastiCache klaster, atau grup keamanan individual untuk masing-masing grup.

Untuk kedua kasus, Anda perlu mengizinkan lalu lintas keluar TCP di ElastiCache port dari sumber dan lalu lintas masuk pada port yang sama ke. ElastiCache Port default adalah 11211 untuk Memcached dan 6379 untuk Redis. Secara default, grup keamanan mengizinkan semua lalu lintas ke luar. Dalam kasus ini, hanya aturan masuk di target grup keamanan yang diperlukan.

Untuk informasi selengkapnya, lihat Pola akses untuk mengakses ElastiCache klaster di VPC Amazon.

# ACL jaringan

Daftar Kontrol Akses (ACL) jaringan adalah aturan yang stateless. Lalu lintas harus diizinkan di kedua arah (masuk dan keluar) agar koneksi berhasil. ACL jaringan ditempatkan ke subnet, bukan untuk sumber daya tertentu. Dimungkinkan untuk memiliki ACL yang sama yang ditugaskan ke ElastiCache dan sumber daya klien, terutama jika mereka berada di subnet yang sama.

Secara default, ACL jaringan mengizinkan semua lalu lintas. Namun, dimungkinkan untuk menyesuaikan ACL agar menolak atau mengizinkan lalu lintas. Selain itu, evaluasi aturan ACL adalah berurutan, yang berarti bahwa aturan dengan nomor terendah yang cocok dengan lalu lintas yang akan mengizinkan atau menolak lalu lintas itu. Konfigurasi minimum untuk mengizinkan lalu lintas Redis adalah:

ACL Jaringan sisi klien:

- Aturan Masuk:
- Nomor aturan: sebaiknya lebih rendah dari semua aturan menolak;
- Jenis: Aturan TCP Khusus;
- Protokol: TCP
- Rentang Port: 1024-65535
- Sumber: 0.0.0.0/0 (atau buat aturan individual untuk subnet cluster) ElastiCache
- Izinkan/Tolak: Izinkan

- Aturan keluar:
- Nomor aturan: sebaiknya lebih rendah dari semua aturan menolak;
- Jenis: Aturan TCP Khusus;
- Protokol: TCP
- Rentang Port: 6379
- Sumber: 0.0.0.0/0 (atau subnet cluster. ElastiCache Ingatlah bahwa menggunakan IP tertentu dapat menimbulkan masalah jika terjadi failover atau penskalaan cluster)
- Izinkan/Tolak: Izinkan

ElastiCache Jaringan ACL:

- Aturan Masuk:

- Nomor aturan: sebaiknya lebih rendah dari semua aturan menolak;
- Jenis: Aturan TCP Khusus;
- Protokol: TCP
- Rentang Port: 6379
- Sumber: 0.0.0.0/0 (atau buat aturan individual untuk subnet cluster) ElastiCache
- Izinkan/Tolak: Izinkan

- Aturan keluar:
- Nomor aturan: sebaiknya lebih rendah dari semua aturan menolak;
- Jenis: Aturan TCP Khusus;
- Protokol: TCP
- Rentang Port: 1024-65535
- Sumber: 0.0.0.0/0 (atau subnet cluster. ElastiCache Ingatlah bahwa menggunakan IP tertentu dapat menimbulkan masalah jika terjadi failover atau penskalaan cluster)
- Izinkan/Tolak: Izinkan

Untuk informasi selengkapnya, silakan lihat ACL Jaringan.

## Tabel rute

Sama seperti ACL Jaringan, setiap subnet dapat memiliki tabel rute yang berbeda. Jika klien dan ElastiCache cluster berada dalam subnet yang berbeda, pastikan bahwa tabel rute mereka memungkinkan mereka untuk mencapai satu sama lain.

Lingkungan yang lebih kompleks, yang melibatkan beberapa VPC, perutean dinamis, atau firewall jaringan, mungkin akan menyulitkan pemecahan masalah. Lihat Validasi konektivitas jaringan untuk mengonfirmasi bahwa pengaturan jaringan Anda sesuai.

## Resolusi DNS

ElastiCache menyediakan titik akhir layanan berdasarkan nama DNS. Titik akhir yang tersedia adalah `Configuration`, `Primary`, `Reader`, dan titik akhir `Node`. Untuk informasi selengkapnya, silakan lihat Menemukan Titik Akhir Koneksi.

Dalam hal failover atau perubahan klaster, alamat yang terkait dengan nama titik akhir mungkin berubah dan akan diperbarui secara otomatis.

Pengaturan DNS khusus (yaitu, tidak menggunakan layanan DNS VPC) mungkin tidak mengetahui nama DNS ElastiCache yang disediakan. Pastikan sistem Anda berhasil menyelesaikan ElastiCache titik akhir menggunakan alat sistem seperti `dig` (seperti yang ditunjukkan berikut) atau`unslookup`.

```
$ dig +short example.xxxxxx.ng.0001.use1.cache.amazonaws.com
example-001.xxxxxx.0001.use1.cache.amazonaws.com.
1.2.3.4
```

Anda juga dapat memaksakan resolusi nama melalui layanan DNS VPC:

```
$ dig +short example.xxxxxx.ng.0001.use1.cache.amazonaws.com @169.254.169.253
example-001.tihewd.0001.use1.cache.amazonaws.com.
1.2.3.4
```

## Mengidentifikasi masalah dengan diagnostik sisi server

CloudWatch metrik dan informasi run-time dari ElastiCache mesin adalah sumber umum atau informasi untuk mengidentifikasi potensi sumber masalah koneksi. Analisis yang baik biasanya dimulai dengan item berikut:

- Penggunaan CPU: Redis adalah aplikasi multi-thread. Akan tetapi, pelaksanaan dari setiap perintah terjadi dalam satu thread tunggal (utama). Untuk alasan ini, ElastiCache berikan metrik `CPUUtilization` dan`EngineCPUUtilization`. `EngineCPUUtilization`menyediakan pemanfaatan CPU yang didedikasikan untuk proses Redis, dan penggunaan `CPUUtilization` di semua vCPU. Simpul dengan lebih dari satu vCPU biasanya memiliki nilai yang berbeda untuk `CPUUtilization` dan `EngineCPUUtilization`, di mana nilai yang kedua umumnya lebih tinggi. `EngineCPUUtilization` yang tinggi dapat disebabkan oleh peningkatan jumlah permintaan atau operasi kompleks yang membutuhkan waktu CPU yang besar untuk diselesaikan. Anda dapat mengidentifikasi keduanya dengan berikut ini:
  - Peningkatan jumlah permintaan: Periksa peningkatan pada metrik lain yang cocok dengan pola `EngineCPUUtilization`. Metrik yang berguna adalah:
    - `CacheHits` dan `CacheMisses`: jumlah permintaan berhasil atau permintaan yang tidak menemukan item yang valid di dalam cache. Jika rasio dari yang meleset lebih tinggi dibandingkan yang berhasil, maka aplikasi memboroskan waktu dan sumber daya dengan permintaan yang tidak membuahkan hasil.
    - `SetTypeCmds` dan `GetTypeCmds`: Kedua metrik ini yang berhubungan dengan `EngineCPUUtilization` dapat membantu untuk memahami jika beban secara signifikan

lebih tinggi untuk permintaan tulis, diukur oleh `SetTypeCmds`, atau lebih tinggi untuk permintaan baca, diukur oleh `GetTypeCmds`. Jika yang lebih banyak adalah beban baca, maka menggunakan beberapa replika baca dapat menyeimbangkan permintaan di beberapa simpul sehingga simpul primer dapat melayani permintaan tulis saja. Dalam cluster yang dinonaktifkan mode cluster, penggunaan replika baca dapat dilakukan dengan membuat konfigurasi koneksi tambahan dalam aplikasi menggunakan endpoint pembaca. ElastiCache Untuk informasi selengkapnya, silakan lihat Menemukan Titik Akhir Koneksi. Operasi baca harus dikirimkan ke koneksi tambahan ini. Operasi tulis akan dilakukan melalui titik akhir primer biasa. Pada mode klaster diaktifkan, sebaiknya menggunakan pustaka yang mendukung replika baca secara asli. Dengan tanda yang tepat, pustaka akan dapat secara otomatis menemukan topologi klaster, simpul replika, mengaktifkan operasi baca melalui perintah Redis READONLY, dan mengirimkan permintaan baca ke replika.

- Jumlah koneksi yang meningkat:
  - `CurrConnections` dan `NewConnections`: `CurrConnection` adalah jumlah koneksi yang dibuat pada saat pengumpulan titik data, sementara `NewConnections` menunjukkan jumlah koneksi yang dibuat pada periode itu.

    Membuat dan menangani koneksi menyebabkan overhead CPU yang cukup besar. Selain itu, proses jabat tangan tiga arah TCP yang diperlukan untuk membuat koneksi baru akan berdampak secara negatif pada waktu respons secara keseluruhan.

    Sebuah ElastiCache node dengan ribuan `NewConnections` per menit menunjukkan bahwa koneksi dibuat dan digunakan hanya dengan beberapa perintah, yang tidak optimal. Menjaga koneksi selalu tersedia dan menggunakan kembali koneksi itu untuk operasi baru adalah praktik terbaik. Hal ini dimungkinkan jika aplikasi klien mendukung dan menerapkan dengan baik pengumpulan koneksi atau koneksi yang persisten. Dengan pengumpulan koneksi, angka `currConnections` tidak akan memiliki variasi besar, dan `NewConnections` seharusnya menjadi serendah mungkin. Redis memberikan kinerja yang optimal dengan currConnections yang kecil. Menjaga currConnection dalam kelompok puluhan atau ratusan meminimalkan penggunaan sumber daya untuk mendukung koneksi tersendiri seperti buffer klien dan siklus CPU untuk melayani koneksi.
- Throughput jaringan:
  - Tentukan bandwidth: ElastiCache node memiliki bandwidth jaringan yang sebanding dengan ukuran node. Karena aplikasi memiliki karakteristik yang berbeda, hasilnya dapat bervariasi sesuai dengan beban kerja. Sebagai contoh, aplikasi dengan permintaan kecil dengan laju yang tinggi cenderung menyebabkan penggunaan CPU lebih besar daripada throughput

jaringan sementara kunci yang lebih besar akan menyebabkan pemanfaatan jaringan yang lebih tinggi. Untuk alasan itu, sebaiknya menguji simpul dengan beban kerja yang sebenarnya untuk pemahaman yang lebih baik mengenai batasan.

Mensimulasi beban dari aplikasi akan memberikan hasil yang lebih akurat. Akan tetapi, alat tolok ukur dapat memberikan gambaran yang baik tentang batasan.

- Untuk kasus di mana permintaan didominasi oleh proses baca, menggunakan replika untuk operasi baca akan mengurangi beban pada simpul primer. Jika kasus penggunaan didominasi proses tulis, digunakannya banyak replika akan memperkuat penggunaan jaringan. Untuk setiap byte yang ditulis ke simpul primer, N byte akan dikirim ke replika, di mana N adalah jumlah replika. Praktik terbaik untuk menulis beban kerja intensif digunakan ElastiCache untuk Redis dengan mode cluster yang diaktifkan sehingga penulisan dapat diseimbangkan di beberapa pecahan, atau ditingkatkan ke tipe node dengan lebih banyak kemampuan jaringan.
- CloudWatchmetrics `NetworkBytesIn`Dan `NetworkBytesOut` memberikan jumlah data yang masuk atau keluar dari node, masing-masing. `ReplicationBytes`adalah lalu lintas yang didedikasikan untuk replikasi data.

Untuk informasi selengkapnya, lihat Batas terkait jaringan.

- Perintah kompleks: Perintah Redis dilayani pada satu thread tunggal, yang berarti bahwa permintaan dilayani secara berurutan. Perintah tunggal yang lambat dapat memengaruhi permintaan lain dan koneksi, yang berpuncak pada terjadinya waktu habis. Penggunaan perintah yang bertindak pada beberapa nilai, kunci, atau jenis data harus dilakukan dengan hati-hati. Koneksi dapat diblokir atau dihentikan tergantung pada jumlah parameter, atau ukuran nilai input atau outputnya.

  Contoh yang terkenal buruk adalah perintah `KEYS`. Perintah ini menyapu seluruh keyspace untuk mencari pola tertentu dan memblokir pelaksanaan dari perintah lain selama pelaksanaannya. Redis menggunakan notasi “Big O” untuk menggambarkan kompleksitas dari perintahnya.

  Perintah keys memiliki O(N) kali kompleksitas, N menjadi jumlah kunci dalam basis data. Oleh karena itu, semakin besar jumlah kunci, semakin lambat perintahnya. `KEYS` dapat menyebabkan masalah dengan cara yang berbeda: Jika tidak menggunakan pola pencarian, perintah ini akan menghasilkan semua nama kunci yang tersedia. Dalam basis data dengan ribuan atau jutaan item, output yang besar akan tercipta dan membanjiri buffer jaringan.

  Jika pola pencarian digunakan, hanya kunci yang cocok dengan pola yang akan dikembalikan ke klien. Namun, mesin masih akan menyapu ke seluruh keyspace untuk mencarinya, dan waktu untuk menyelesaikan perintah akan sama.

Alternatif untuk KEYS adalah perintah SCAN. Perintah ini melakukan iteratesi atas keyspace dan membatasi iterasi dalam jumlah tertentu dari item, menghindari pemblokiran yang berkepanjangan pada mesin.

Scan memiliki parameter COUNT, digunakan untuk mengatur ukuran dari blok iterasi. Nilai defaultnya adalah 10 (10 item per iterasi).

Tergantung pada jumlah item dalam basis data, blok dengan nilai COUNT yang kecil akan membutuhkan lebih banyak iterasi untuk menyelesaikan perintah scan penuh, dan nilai yang lebih besar ini akan membuat mesin sibuk lebih lama di setiap iterasi. Sementara nilai count yang kecil akan membuat SCAN lebih lambat pada basis data yang besar, nilai yang lebih besar dapat menyebabkan masalah yang sama seperti disebutkan untuk KEYS.

Sebagai contoh, menjalankan perintah SCAN dengan nilai count sebesar 10 akan membutuhkan 100.000 pengulangan pada basis data dengan 1 juta kunci. Jika waktu bolak-balik jaringan rata-rata adalah 0,5 milidetik, sekitar 50.000 milidetik (50 detik) akan dihabiskan untuk mengirimkan permintaan ini.

Di sisi lain, jika nilai count adalah 100,0000, iterasi tunggal akan diperlukan dan hanya 0,5 milidetik yang akan dihabiskan untuk mengirimnya. Akan tetapi, mesin akan sepenuhnya terblokir untuk operasi lain sampai perintah itu selesai menyapu semua keyspace.

Selain KEYS, beberapa perintah lain berpotensi membahayakan jika tidak digunakan dengan benar. Untuk melihat daftar semua perintah dan kompleksitas waktunya masing-masing, kunjungi https://redis.io/commands.

Contoh potensi masalah:

- Skrip Lua: Redis menyediakan interpreter Lua tertanam, yang memungkinkan pelaksanaan skrip di sisi server. Skrip Lua pada Redis dijalankan di tingkat mesin dan bersifat atom menurut definisi, yang berarti bahwa tidak ada perintah atau skrip lain akan diizinkan untuk bekerja sementara skrip itu dijalankan. Skrip Lua menyediakan kemungkinan untuk menjalankan beberapa perintah, algoritma pengambilan keputusan, penguraian data, dan lain-lain secara langsung pada mesin Redis. Meskipun sifat atom dari skrip dan kemungkinan melepaskan aplikasi cukup menarik, skrip harus digunakan dengan hati-hati dan untuk operasi yang kecil. ElastiCacheAktif, waktu eksekusi skrip Lua dibatasi hingga 5 detik. Skrip yang belum ditulis ke keyspace akan secara otomatis dihentikan setelah periode 5 detik. Untuk menghindari kerusakan dan inkonsistensi data, simpul akan melakukan failover jika pelaksanaan skrip belum selesai dalam 5 detik dan tetap menjalankan proses tulis apapun selama pelaksanaan

skrip. Perintah transactions adalah alternatif untuk menjamin konsistensi dari perubahan beberapa kunci yang terkait di Redis. Perintah transaction memungkinkan eksekusi dari suatu blok perintah, memperhatikan perubahan pada kunci yang telah ada. Jika salah satu kunci yang diperhatikan berubah sebelum selesainya transaksi, maka semua perubahan akan dibuang.

- Penghapusan item secara massal: perintah `DEL` menerima beberapa parameter, yang merupakan nama kunci yang akan dihapus. Operasi penghapusan bersifat sinkron dan akan mengambil waktu CPU yang signifikan jika daftar parameter besar, atau berisi daftar yang panjang, set, set berurutan, atau hash yang besar (struktur data memegang beberapa sub-item). Dengan kata lain, bahkan penghapusan satu kunci pun dapat memakan waktu lama jika kunci itu memiliki banyak elemen. Alternatif untuk `DEL` adalah `UNLINK`, yang merupakan perintah asinkron yang tersedia sejak Redis 4. `UNLINK` seharusnya lebih dipilih daripada `DEL` jika dimungkinkan. Dimulai ElastiCache untuk Redis 6x, `lazyfree-lazy-user-del` parameter membuat `DEL` perintah berperilaku seperti `UNLINK` saat diaktifkan. Untuk informasi selengkapnya, lihat Perubahan Parameter Redis 6.0.
- Perintah yang bekerja pada beberapa kunci: `DEL` disebutkan sebelumnya sebagai perintah yang menerima beberapa argumen dan waktu pelaksanaannya akan berbanding lurus dengan itu. Namun, Redis menyediakan lebih banyak perintah yang bekerja dengan cara yang sama. Sebagai contoh, `MSET` dan `MGET` memungkinkan penyisipan atau pengambilan beberapa kunci String sekaligus. Penggunaannya mungkin bermanfaat untuk mengurangi latensi jaringan yang terjadi pada beberapa perintah tersendiri `SET` atau `GET`. Akan tetapi, daftar parameter yang panjang akan memengaruhi penggunaan CPU.

  Sementara pemanfaatan CPU sendiri bukan penyebab untuk masalah konektivitas, menghabiskan terlalu banyak waktu untuk memproses satu atau beberapa perintah atas beberapa kunci dapat menyebabkan kegagalan permintaan lain dan meningkatkan pemanfaatan CPU secara keseluruhan.

  Jumlah kunci dan ukurannya akan mempengaruhi kompleksitas perintah dan karena itu berpengaruh pada waktu penyelesaian.

  Contoh lain dari perintah yang dapat bertindak atas beberapa kunci:`HMGET`, `HMSET`, `MSETNX`, `PFCOUNT`, `PFMERGE`, `SDIFF`, `SDIFFSTORE`, `SINTER`, `SINTERSTORE`, `SUNION`, `SUNIONSTORE`, `TOUCH`, `ZDIFF`, `ZDIFFSTORE`, `ZINTER` atau `ZINTERSTORE`.
- Perintah yang bekerja pada beberapa tipe data: Redis juga menyediakan perintah yang bekerja pada satu atau beberapa kunci, terlepas dari tipe datanya. ElastiCache untuk

Redis menyediakan metrik `KeyBasedCmds` untuk memantau perintah tersebut. Metrik ini menjumlahkan eksekusi dari perintah berikut pada periode yang dipilih:

- Kompleksitas O(N):
  - `KEYS`
- O(1)
  - `EXISTS`
  - `OBJECT`
  - `PTTL`
  - `RANDOMKEY`
  - `TTL`
  - `TYPE`
  - `EXPIRE`
  - `EXPIREAT`
  - `MOVE`
  - `PERSIST`
  - `PEXPIRE`
  - `PEXPIREAT`
  - `UNLINK (O(N)` untuk mengeklaim kembali memori. Namun tugas mengeklaim kembali memori itu terjadi di thread yang terpisah dan tidak memblokir mesin
- Waktu kompleksitas yang berbeda akan tergantung pada jenis data:
  - `DEL`
  - `DUMP`
  - `RENAME` dianggap perintah dengan kompleksitas O(1), tetapi menjalankan `DEL` secara internal. Waktu pelaksanaan akan bervariasi tergantung pada ukuran kunci yang diganti namanya.
  - `RENAMENX`
  - `RESTORE`
  - `SORT`
- Hash besar: Hash adalah jenis data yang memungkinkan satu kunci dengan beberapa sub-item nilai kunci. Setiap hash dapat menyimpan 4.294.967.295 item dan operasi pada hash yang besar dapat menjadi mahal. Sama dengan `KEYS`, hash mempunyai perintah `HKEYS`

dengan kompleksitas waktu O(N), N adalah jumlah item dalam hash. `HSCAN` seharusnya lebih dipilih dibandingkan `HKEYS` untuk menghindari perintah yang berjalan lama. `HDEL`, `HGETALL`, `HMGET`, `HMSET` dan `HVALS` adalah perintah yang harus digunakan dengan hati-hati pada hash besar.

- Struktur big data lainnya: Selain hash, struktur data lainnya dapat memakan waktu CPU. Sets, Lists, Sorted Sets, dan Hyperloglogs juga dapat memakan waktu yang lama untuk dimanipulasi tergantung pada ukuran dan perintah yang digunakan. Untuk informasi selengkapnya tentang perintah tersebut, lihat https://redis.io/commands.

## Validasi konektivitas jaringan

Setelah meninjau konfigurasi jaringan yang berkaitan dengan resolusi DNS, grup keamanan, ACL jaringan, dan tabel rute, konektivitas dapat divalidasi dengan VPC Reachability Analyzer dan alat sistem.

Reachability Analyzer akan menguji konektivitas jaringan dan mengonfirmasi jika semua persyaratan dan izin terpenuhi. Untuk tes di bawah ini Anda akan memerlukan ID ENI (Elastic Network Interface Identification) dari salah satu ElastiCache node yang tersedia di VPC Anda. Anda dapat menemukannya dengan melakukan hal berikut:

1. Kunjungi https://console.aws.amazon.com/ec2/v2/home?#NIC:
2. Filter daftar antarmuka dengan nama ElastiCache cluster Anda atau alamat IP yang didapat dari validasi DNS sebelumnya.
3. Tuliskan atau simpan ENI ID. Jika beberapa antarmuka ditampilkan, tinjau deskripsi untuk mengonfirmasi bahwa mereka termasuk dalam ElastiCache cluster yang tepat dan pilih salah satunya.
4. Lanjutkan ke langkah berikutnya.
5. Buat jalur analisis di https://console.aws.amazon.com/vpc/home? # ReachabilityAnalyzer dan pilih opsi berikut:

   - Jenis Sumber: Pilih instance jika ElastiCache klien Anda berjalan pada instans Amazon EC2 atau Antarmuka Jaringan jika menggunakan layanan lain, seperti AWS Fargate Amazon ECS dengan jaringan awsvpc, dll) AWS Lambda, dan ID sumber daya masing-masing (instans EC2 atau ID ENI);
   - Jenis Tujuan: Pilih Antarmuka Jaringan dan pilih ElastiCache ENI dari daftar.

- Port tujuan: tentukan 6379 untuk Redis atau 11211 ElastiCache untuk untuk Memcached. ElastiCache Itu adalah port yang ditetapkan dengan konfigurasi default dan contoh ini mengasumsikan bahwa port tersebut tidak berubah.
- Protokol: TCP

Buat jalur analisis dan tunggu beberapa saat untuk hasilnya. Jika status tidak terjangkau, buka detail analisis dan tinjau Penjelajah analisis untuk detail di mana permintaan diblokir.

Jika tes penjangkauan sudah lulus, lanjutkan ke verifikasi di tingkat OS.

Untuk memvalidasi konektivitas TCP pada port ElastiCache layanan: Di Amazon Linux, `Nping` tersedia dalam paket `nmap` dan dapat menguji konektivitas TCP pada ElastiCache port, serta menyediakan waktu pulang-pergi jaringan untuk membuat koneksi. Gunakan ini untuk memvalidasi konektivitas jaringan dan latensi saat ini ke ElastiCache cluster, seperti yang ditunjukkan berikut:

```
$ sudo nping --tcp -p 6379 example.xxxxxx.ng.0001.use1.cache.amazonaws.com

Starting Nping 0.6.40 ( http://nmap.org/nping ) at 2020-12-30 16:48 UTC
SENT (0.0495s) TCP ...
(Output suppressed )

Max rtt: 0.937ms | Min rtt: 0.318ms | Avg rtt: 0.449ms
Raw packets sent: 5 (200B) | Rcvd: 5 (220B) | Lost: 0 (0.00%)
Nping done: 1 IP address pinged in 4.08 seconds
```

Secara default, `nping` mengirimkan 5 paket penyelidikan dengan waktu tunda 1 detik di antara paket itu. Anda dapat menggunakan opsi “-c” untuk menambah jumlah paket penyelidikan dan “--delay “ untuk mengubah waktu untuk mengirim pengujian baru.

Jika pengujian dengan `nping` gagal dan pengujian VPC Reachability Analyzer lulus, mintalah administrator sistem Anda untuk meninjau kemungkinan aturan firewall berbasis Host, aturan perutean asimetris, atau pembatasan lain yang dimungkinkan di tingkat sistem operasi.

Di ElastiCache konsol, periksa apakah Enkripsi dalam transit diaktifkan di detail ElastiCache klaster Anda. Jika enkripsi bergerak diaktifkan, lakukan konfirmasi jika sesi TLS dapat dibuat dengan perintah berikut:

```
openssl s_client -connect example.xxxxxx.use1.cache.amazonaws.com:6379
```

Output yang panjang akan keluar jika koneksi dan negosiasi TLS berhasil. Periksa kode yang dihasilkan yang terdapat di baris terakhir, nilainya harus `0 (ok)`. Jika openssl menghasilkan sesuatu yang berbeda, periksa alasan untuk kesalahan itu di https://www.openssl.org/docs/man1.0.2/man1/verify.html#DIAGNOSTICS.

Jika semua pengujian infrastruktur dan sistem operasi lulus tetapi aplikasi Anda masih tidak dapat terhubung ElastiCache, periksa apakah konfigurasi aplikasi sesuai dengan pengaturan. ElastiCache Kesalahan yang umum adalah:

- Aplikasi Anda tidak mendukung mode ElastiCache klaster, dan ElastiCache mengaktifkan mode cluster;
- Aplikasi Anda tidak mendukung TLS/SSL, dan ElastiCache memiliki enkripsi dalam transit yang diaktifkan;
- Aplikasi mendukung TLS/SSL tetapi tidak memiliki tanda konfigurasi atau otoritas sertifikasi tepercaya yang tepat;

## Batas terkait jaringan

- Jumlah maksimum koneksi: Ada batas keras untuk koneksi secara serentak. Setiap ElastiCache node memungkinkan hingga 65.000 koneksi simultan di semua klien. Batas ini dapat dipantau melalui `CurrConnections` metrik pada. CloudWatch Namun, klien juga memiliki batas untuk koneksi keluar. Pada Linux, periksa rentang port ephemeral yang diizinkan dengan perintah:

  ```
  # sysctl net.ipv4.ip_local_port_range
  net.ipv4.ip_local_port_range = 32768 60999
  ```

  Pada contoh sebelumnya, 28231 koneksi akan diizinkan dari sumber yang sama, ke IP tujuan (ElastiCache node) dan port yang sama. Perintah berikut menunjukkan berapa banyak koneksi yang ada untuk ElastiCache node tertentu (IP 1.2.3.4):

  ```
  ss --numeric --tcp state connected "dst 1.2.3.4 and dport == 6379" | grep -vE
   '^State' | wc -l
  ```

  Jika jumlahnya terlalu tinggi, sistem Anda mungkin menjadi kelebihan beban mencoba untuk memproses permintaan koneksi. Sebaiknya mempertimbangkan menerapkan teknik seperti pengumpulan koneksi atau koneksi persisten untuk menangani koneksi itu dengan lebih baik. Jika memungkinkan, lakukan konfigurasi kumpulan koneksi untuk membatasi jumlah maksimum

koneksi hanya beberapa ratus. Selain itu, logika back-off untuk menangani masalah waktu habis atau pengecualian koneksi lain juga dianjurkan untuk menghindari masalah koneksi churn.

- Batas lalu lintas jaringan: Periksa CloudWatch metrik berikut untuk Redis untuk mengidentifikasi kemungkinan batas jaringan yang terkena pada ElastiCache node:
  - `NetworkBandwidthInAllowanceExceeded` / `NetworkBandwidthOutAllowanceExceeded`: Paket jaringan yang ditunda karena throughput melebihi batas agregasi bandwidth.

    Penting untuk dicatat bahwa setiap byte yang ditulis ke simpul primer akan direplikasi ke N replika, di mana N adalah jumlah replika. Klaster dengan jenis simpul kecil, beberapa replika, dan permintaan tulis intensif mungkin tidak mampu mengatasi backlog replikasi. Untuk kasus seperti itu, praktik terbaik adalah menaikkan skala (mengubah jenis simpul), menskalakan ke luar (menambahkan serpihan dalam klaster dengan mode klaster diaktifkan), mengurangi jumlah replika, atau meminimalkan jumlah proses tulis.
  - `NetworkConntrackAllowanceExceeded`: Paket yang ditunda karena telah terlampauinya jumlah maksimum koneksi yang dilacak di seluruh grup keamanan yang ditetapkan ke simpul. Koneksi baru kemungkinan akan gagal selama periode ini.
  - `NetworkPackets PerSecondAllowanceExceeded`: Jumlah maksimum paket per detik terlampaui. Beban kerja berdasarkan laju yang tinggi dari permintaan yang sangat kecil dapat mencapai batas ini sebelum bandwidth mencapai maksimum.

  Metrik di atas adalah cara ideal untuk mengonfirmasi simpul yang mencapai batas jaringannya. Namun, batas juga dapat diidentifikasi dengan bentuk plato pada metrik jaringan.

  Jika dataran tinggi teramati untuk waktu yang lama, kemungkinan akan diikuti oleh jeda replikasi, peningkatan byte yang Digunakan untuk cache, penurunan memori yang dapat dikosongkan, swap tinggi, dan penggunaan CPU. Instans Amazon EC2 juga memiliki batas jaringan yang dapat dilacak melalui metrik driver ENA. Instans Linux dengan dukungan jaringan yang ditingkatkan dan penggerak ENA 2.2.10 atau yang lebih baru dapat meninjau penghitung batas dengan perintah:

```
# ethtool -S eth0 | grep "allowance_exceeded"
```

## Penggunaan CPU

Metrik penggunaan CPU adalah titik awal penyelidikan, dan item berikut dapat membantu mempersempit kemungkinan masalah di ElastiCache samping:

- Redis SlowLogs: Konfigurasi ElastiCache default mempertahankan 128 perintah terakhir yang membutuhkan waktu lebih dari 10 milidetik untuk diselesaikan. Riwayat perintah lambat disimpan selama runtime mesin dan akan hilang jika terjadi kegagalan atau mulai ulang. Jika daftar mencapai 128 entri, maka peristiwa lama akan dihapus untuk memberikan tempat untuk peristiwa baru. Ukuran dari daftar peristiwa lambat dan waktu pelaksanaan yang dianggap lambat dapat diubah melalui parameter `slowlog-max-len` dan `slowlog-log-slower-than` di dalam grup parameter khusus. Daftar slowlogs dapat diambil dengan menjalankan `SLOWLOG GET 128` pada mesin, 128 adalah 128 perintah lambat terakhir yang dilaporkan. Setiap entri memiliki bidang berikut:

```
1) 1) (integer) 1 -----------> Sequential ID
   2) (integer) 1609010767 --> Timestamp (Unix epoch time)of the Event
   3) (integer) 4823378 -----> Time in microseconds to complete the command.
   4) 1) "keys" -------------> Command
      2) "*" ----------------> Arguments
   5) "1.2.3.4:57004"-> Source
```

Peristiwa di atas terjadi pada tanggal 26 Desember pukul 19:26:07 UTC, membutuhkan 4,8 detik (4,823ms) untuk diselesaikan dan disebabkan oleh perintah KEYS yang diminta dari klien 1.2.3.4.

Di Linux, stempel waktu dapat dikonversi dengan perintah tanggal:

```
$ date --date='@1609010767'
Sat Dec 26 19:26:07 UTC 2020
```

Pada Python:

```
>>> from datetime import datetime
>>> datetime.fromtimestamp(1609010767)
datetime.datetime(2020, 12, 26, 19, 26, 7)
```

Atau di Windows dengan PowerShell:

```
PS D:\Users\user> [datetimeoffset]::FromUnixTimeSeconds('1609010767')
DateTime      : 12/26/2020 7:26:07 PM
UtcDateTime
                : 12/26/2020 7:26:07 PM
LocalDateTime : 12/26/2020 2:26:07 PM
Date          : 12/26/2020 12:00:00 AM
```

```
Day           : 26
DayOfWeek
                : Saturday
DayOfYear     : 361
Hour          : 19
Millisecond   : 0
Minute        : 26
Month
                : 12
Offset        : 00:00:00Ticks         : 637446075670000000
UtcTicks
                : 637446075670000000
TimeOfDay     : 19:26:07
Year          : 2020
```

Banyak perintah lambat dalam waktu singkat (menit yang sama atau kurang) menjadi hal yang dikhawatirkan. Tinjau sifat dari perintah dan bagaimana perintah itu dapat dioptimalkan (lihat contoh sebelumnya). Jika perintah dengan kompleksitas waktu O(1) sering dilaporkan, periksa faktor lain yang menyebabkan penggunaan CPU tinggi yang disebutkan sebelumnya.

- Metrik latensi: ElastiCache untuk Redis menyediakan CloudWatch metrik untuk memantau latensi rata-rata untuk berbagai kelas perintah. Titik data dihitung dengan membagi jumlah pelaksanaan perintah dalam kategori dengan total waktu pelaksanaan pada periode tersebut. Penting untuk dipahami bahwa hasil metrik latensi merupakan kumpulan dari beberapa perintah. Satu perintah dapat menyebabkan hasil tidak terduga, seperti waktu habis, tanpa dampak signifikan pada metrik. Untuk kasus seperti ini, peristiwa slowlog akan menjadi sumber informasi yang lebih akurat. Daftar berikut berisi metrik latensi yang tersedia dan perintah terkait yang memengaruhinya.
  - EvalBasedCmdsLatency: terkait dengan perintah Lua Script,`eval`,`evalsha`;
  - GeoSpatialBasedCmdsLatency: `geodist`, `geohash`, `geopos`, `georadius`, `georadiusbymember`, `geoadd`;
  - GetTypeCmdsLatency: Baca perintah, terlepas dari tipe data;
  - HashBasedCmdsLatency: `hexists`, `hget`, `hgetall`, `hkeys`, `hlen`, `hmget`, `hvals`, `hstrlen`, `hdel`, `hincrby`, `hincrbyfloat`, `hmset`, `hset`, `hsetnx`;
  - HyperLogLogBasedCmdsLatency: `pfselftest`, `pfcount`, `pfdebug`, `pfadd`, `pfmerge`;
  - KeyBasedCmdsLatency: Perintah yang dapat bertindak atas tipe data yang berbeda: dump `existskeys`,`object`,`pttl`,,`randomkey`,`ttl`,`type`,`del`,`expire`,`expireat`,`move`,`persist`,`pexpire`
  - ListBasedCmdsLatency: lindex, llen, lrange, blpop, brpop, brpoplpush, linsert, lpop, lpush, lpushx, lrem, lset, ltrim, rpop, rpoplpush, rpush, rpushx;

  - PubSubBasedCmdsLatency: psubscribe, publish, pubsub, punsubscribe, subscribe, unsubscribe;
  - SetBasedCmdsLatency: `scard`, `sdiff`, `sinter`, `sismember`, `smembers`, `srandmember`, `sunion`, `sadd`, `sdiffstore`, `sinterstore`, `smove`, `spop`, `srem`, `sunionstore`;
  - SetTypeCmdsLatency: Menulis perintah, terlepas dari tipe data;
  - SortedSetBasedCmdsLatency: `zcard`, `zcount`, `zrange`, `zrangebyscore`, `zrank`, `zrevrange`, `zrevrangebyscore`, `zrevrank`, `zscore`, `zrangebylex`, `zrevrangebylex`, `zlexcount`, `zadd`. `zincrby`, `zinterstore`, `zrem`, `zremrangebyrank`, `zremrangebyscore`, `zunionstore`, `zremrangebylex`, `zpopmax`, `zpopmin`, `bzpopmin`, `bzpopmax`;
  - StringBasedCmdsLatency: `bitcount`, `get`, `getbit`, `getrange`, `mget`, `strlen`, `substr`, `bitpos`, `append`, `bitop`, `bitfield`, `decr`, `decrby`, `getset`, `incr`, `incrby`, `incrbyfloat`, `mset`, `msetnx`, `psetex`, `set`, `setbit`, `setex`, `setnx`, `setrange`;
  - StreamBasedCmdsLatency: `xrange`, `xrevrange`, `xlen`, `xread`, `xpending`, `xinfo`, `xadd`, `xgroup`, `readgroup`, `xack`, `xclaim`, `xdel`, `xtrim`, `xsetid`;
- Perintah runtime Redis:
  - info commandstats: Menyediakan daftar perintah yang dilaksanakan sejak mesin Redis dimulai, jumlah pelaksanaan kumulatifnya, jumlah waktu pelaksanaan, dan rata-rata waktu pelaksanaan per perintah;
  - client list: Menyediakan daftar dari klien yang saat ini tersambung dan informasi yang relevan seperti penggunaan buffer, perintah yang dilaksanakan terakhir, dll;
- Backup dan replikasi: ElastiCache untuk versi Redis lebih awal dari 2.8.22 gunakan proses bercabang untuk membuat cadangan dan memproses sinkronisasi penuh dengan replika. Metode ini mungkin menyebabkan overhead memori yang besar untuk kasus penggunaan proses tulis intensif.

  Dimulai dengan ElastiCache Redis 2.8.22, AWS memperkenalkan metode pencadangan dan replikasi tanpa garpu. Metode yang baru dapat menunda proses tulis untuk mencegah kegagalan. Kedua metode dapat menyebabkan periode pemanfaatan CPU yang lebih tinggi, yang menyebabkan waktu respons lebih tinggi dan akibatnya menyebabkan klien mengalami waktu habis selama melaksanakan perintah. Selalu periksa apakah kegagalan klien terjadi selama jendela pencadangan atau metrik `SaveInProgress` bernilai 1 pada periode tersebut. Sebaiknya menjadwalkan jendela pencadangan untuk periode pemanfaatan yang rendah untuk meminimalkan kemungkinan masalah dengan klien atau kegagalan proses pencadangan.

# Koneksi yang dihentikan dari sisi server

Default ElastiCache untuk konfigurasi Redis membuat koneksi klien dibuat tanpa batas waktu. Akan tetapi, dalam beberapa kasus penghentian koneksi mungkin diinginkan. Misalnya:

- Bug dalam aplikasi klien dapat menyebabkan koneksi dilupakan dan tetap tersambung dengan keadaan diam. Ini disebut “kebocoran koneksi “ dan konsekuensinya adalah peningkatan yang tetap pada jumlah koneksi tersambung yang diamati pada metrik `CurrConnections`. Perilaku ini dapat mengakibatkan kejenuhan pada klien atau ElastiCache sisi. Ketika perbaikan langsung tidak dimungkinkan dari sisi klien, beberapa administrator menetapkan nilai” batas waktu “dalam grup ElastiCache parameter mereka. Waktu habis adalah waktu dalam detik yang diizinkan untuk koneksi diam untuk bertahan. Jika klien tidak mengirimkan permintaan apapun dalam periode itu, mesin Redis akan mengakhiri koneksi itu segera setelah koneksi itu mencapai nilai waktu habis. Nilai waktu habis yang kecil dapat mengakibatkan pemutusan koneksi yang tidak perlu dan klien akan perlu menangani ini dengan tepat dan menyambungkan kembali, yang menyebabkan penundaan.
- Memori yang digunakan untuk menyimpan kunci dibagikan dengan buffer klien. Klien lambat dengan permintaan atau tanggapan besar mungkin menuntut sejumlah besar memori untuk menangani buffer. Default ElastiCache untuk konfigurasi Redis tidak membatasi ukuran buffer keluaran klien biasa. Jika batas `maxmemory` tercapai, mesin akan mencoba untuk melakukan pengosongan item untuk memenuhi penggunaan buffer. Dalam kondisi memori yang sangat rendah, ElastiCache untuk Redis mungkin memilih untuk memutuskan klien yang mengkonsumsi buffer keluaran klien besar untuk membebaskan memori dan mempertahankan kesehatan cluster.

  Dimungkinkan untuk membatasi ukuran buffer klien dengan konfigurasi khusus dan klien yang mencapai batas ini akan terputus. Namun, klien harus dapat menangani pemutusan yang tidak terduga. Parameter untuk menangani ukuran buffer untuk klien biasa adalah sebagai berikut:

  - client-query-buffer-limit: Ukuran maksimum permintaan input tunggal;
  - client-output-buffer-limit-normal-soft-limit: Batas lunak untuk koneksi klien. Koneksi akan dihentikan jika tetap di atas batas lunak selama lebih dari waktu dalam detik yang ditentukan pada client-output-buffer-limit - normal-soft-seconds atau jika mencapai batas keras;
  - client-output-buffer-limit-normal-soft-seconds: Waktu yang diizinkan untuk koneksi melebihi client-output-buffer-limit -normal-soft-limit;
  - client-output-buffer-limit-normal-hard-limit: Koneksi yang mencapai batas ini akan segera dihentikan.

Selain buffer klien biasa, pilihan berikut mengontrol buffer untuk simpul replika dan klien Pub/Sub (Penerbitan/Berlangganan):

- client-output-buffer-limit-replica-hard-limit;
- client-output-buffer-limit-replica-soft-seconds;
- client-output-buffer-limit-replica-hard-limit;
- client-output-buffer-limit-pubsub-soft-limit;
- client-output-buffer-limit-pubsub-soft-seconds;
- client-output-buffer-limit-pubsub-hard-limit;

## Pemecahan masalah sisi klien untuk instans Amazon EC2

Beban dan daya tanggap di sisi klien juga dapat memengaruhi permintaan. ElastiCache Batas dari instans EC2 dan sistem operasi harus ditinjau dengan hati-hati sambil melakukan pemecahan masalah konektivitas yang putus-putus atau masalah waktu habis. Beberapa poin penting untuk diamati:

- CPU
  - Penggunaan CPU instans EC2: Pastikan CPU belum jenuh atau mendekati 100 persen. Analisis historis dapat dilakukan melalui CloudWatch, namun perlu diingat bahwa perincian titik data adalah 1 menit (dengan pemantauan terperinci diaktifkan) atau 5 menit;
  - Jika menggunakan instans EC2 burstable, pastikan bahwa saldo kredit CPU belum habis. Informasi ini tersedia pada `CPUCreditBalance` CloudWatch metrik.
  - Periode singkat penggunaan CPU yang tinggi dapat menyebabkan batas waktu tanpa mencerminkan pemanfaatan 100 persen. CloudWatch Kasus seperti itu memerlukan pemantauan waktu nyata dengan peralatan sistem operasi seperti `top`, `ps`, dan `mpstat`.
- Jaringan
  - Periksa apakah throughput Jaringan berada di bawah nilai yang dapat diterima sesuai dengan kemampuan instans. Untuk informasi selengkapnya, lihat Tipe Instans Amazon EC2.
  - Pada instans dengan `ena` penggerak Jaringan yang Ditingkatkan, periksa statistik ena untuk batas waktu habis atau batas yang terlampaui. Statistik berikut berguna untuk mengonfirmasi kejenuhan batas jaringan:
    - `bw_in_allowance_exceeded` / `bw_out_allowance_exceeded`: jumlah paket yang ditunda karena kelebihan throughput masuk atau keluar;

- `conntrack_allowance_exceeded`: jumlah paket yang tidak diteruskan karena batas pelacakan koneksi dari grup keamanan. Koneksi baru akan gagal saat batas ini jenuh;
- `linklocal_allowance_exceeded`: jumlah paket yang tidak diteruskan karena permintaan berlebihan untuk metadata dari instans, NTP melalui DNS VPC. Batasnya adalah 1024 paket per detik untuk semua layanan;
- `pps_allowance_exceeded`: jumlah paket yang tidak diteruskan karena rasio paket per detik yang berlebihan. Batas PPS dapat dicapai ketika lalu lintas jaringan terdiri dari ribuan atau jutaan permintaan yang sangat kecil per detik. ElastiCache Lalu lintas dapat dioptimalkan untuk memanfaatkan paket jaringan dengan lebih baik melalui saluran pipa atau perintah yang melakukan beberapa operasi sekaligus seperti `MGET` alih-alih. `GET`

## Membedah waktu yang dibutuhkan untuk menyelesaikan satu permintaan tunggal

- Di jaringan: `Tcpdump` dan `Wireshark` (tshark pada baris perintah) adalah alat yang berguna untuk memahami berapa banyak waktu yang dibutuhkan permintaan untuk melakukan perjalanan jaringan, menekan ElastiCache mesin dan mendapatkan pengembalian. Contoh berikut menyorot satu permintaan tunggal yang dibuat dengan perintah berikut:

```
$ echo ping | nc example.xxxxxx.ng.0001.use1.cache.amazonaws.com 6379
+PONG
```

Sejajar dengan perintah di atas, tcpdump dijalankan dan menghasilkan:

```
$ sudo tcpdump -i any -nn port 6379 -tt
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
1609428918.917869 IP 172.31.11.142.40966
    > 172.31.11.247.6379: Flags [S], seq 177032944, win 26883, options [mss
 8961,sackOK,TS val 27819440 ecr 0,nop,wscale 7], length 0
1609428918.918071 IP 172.31.11.247.6379 > 172.31.11.142.40966: Flags [S.], seq
 53962565, ack 177032945, win
    28960, options [mss 1460,sackOK,TS val 3788576332 ecr 27819440,nop,wscale 7],
 length 0
1609428918.918091 IP 172.31.11.142.40966 > 172.31.11.247.6379: Flags [.], ack 1, win
 211, options [nop,nop,TS val 27819440 ecr 3788576332], length 0
1609428918.918122
```

```
    IP 172.31.11.142.40966 > 172.31.11.247.6379: Flags [P.], seq 1:6, ack 1, win 211,
 options [nop,nop,TS val 27819440 ecr 3788576332], length 5: RESP "ping"
1609428918.918132 IP 172.31.11.142.40966 > 172.31.11.247.6379: Flags [F.], seq 6, ack
    1, win 211, options [nop,nop,TS val 27819440 ecr 3788576332], length 0
1609428918.918240 IP 172.31.11.247.6379 > 172.31.11.142.40966: Flags [.], ack 6, win
 227, options [nop,nop,TS val 3788576332 ecr 27819440], length 0
1609428918.918295
    IP 172.31.11.247.6379 > 172.31.11.142.40966: Flags [P.], seq 1:8, ack 7, win 227,
 options [nop,nop,TS val 3788576332 ecr 27819440], length 7: RESP "PONG"
1609428918.918300 IP 172.31.11.142.40966 > 172.31.11.247.6379: Flags [.], ack 8, win
    211, options [nop,nop,TS val 27819441 ecr 3788576332], length 0
1609428918.918302 IP 172.31.11.247.6379 > 172.31.11.142.40966: Flags [F.], seq 8, ack
 7, win 227, options [nop,nop,TS val 3788576332 ecr 27819440], length 0
1609428918.918307
    IP 172.31.11.142.40966 > 172.31.11.247.6379: Flags [.], ack 9, win 211, options
 [nop,nop,TS val 27819441 ecr 3788576332], length 0
^C
10 packets captured
10 packets received by filter
0 packets dropped by kernel
```

Dari output di atas kita dapat mengonfirmasi bahwa jabat tangan tiga arah TCP diselesaikan dalam 222 mikrodetik (918091 - 917869) dan perintah ping dikirim dan dikembalikan dalam 173 mikrodetik (918295 - 918122).

Dibutuhkan waktu 438 mikrodetik (918307 - 917869) mulai dari membuat permintaan hingga koneksi ditutup. Hasil tersebut akan memastikan bahwa waktu respons jaringan dan mesin adalah baik dan penyelidikan dapat berfokus pada komponen lainnya.

- Pada sistem operasi: `Strace` dapat membantu mengidentifikasi kesenjangan waktu pada tingkat OS. Analisis aplikasi aktual akan jauh lebih luas dan dianjurkan menggunakan alat profil khusus untuk aplikasi atau debugger. Contoh berikut hanya menunjukkan jika komponen sistem operasi dasar bekerja seperti yang diharapkan, jika tidak, penyelidikan lebih lanjut mungkin diperlukan. Menggunakan Redis yang sama, perintah `PING` dengan `strace` kita mendapatkan:

```
$ echo ping | strace -f -tttt -r -e trace=execve,socket,open,recvfrom,sendto
 nc example.xxxxxx.ng.0001.use1.cache.amazonaws.com (http://
example.xxxxxx.ng.0001.use1.cache.amazonaws.com/)
    6379
1609430221.697712 (+ 0.000000) execve("/usr/bin/nc", ["nc",
 "example.xxxxxx.ng.0001.use"..., "6379"], 0x7fffede7cc38 /* 22 vars */) = 0
```

```
1609430221.708955 (+ 0.011231) socket(AF_UNIX, SOCK_STREAM|SOCK_CLOEXEC|
SOCK_NONBLOCK, 0) = 3
1609430221.709084
    (+ 0.000124) socket(AF_UNIX, SOCK_STREAM|SOCK_CLOEXEC|SOCK_NONBLOCK, 0) = 3
1609430221.709258 (+ 0.000173) open("/etc/nsswitch.conf", O_RDONLY|O_CLOEXEC) = 3
1609430221.709637 (+ 0.000378) open("/etc/host.conf", O_RDONLY|O_CLOEXEC) = 3
1609430221.709923
    (+ 0.000286) open("/etc/resolv.conf", O_RDONLY|O_CLOEXEC) = 3
1609430221.711365 (+ 0.001443) open("/etc/hosts", O_RDONLY|O_CLOEXEC) = 3
1609430221.713293 (+ 0.001928) socket(AF_INET, SOCK_DGRAM|SOCK_CLOEXEC|SOCK_NONBLOCK,
 IPPROTO_IP) = 3
1609430221.717419
    (+ 0.004126) recvfrom(3, "\362|
\201\200\0\1\0\2\0\0\0\0\rnotls20201224\6tihew"..., 2048, 0, {sa_family=AF_INET,
 sin_port=htons(53), sin_addr=inet_addr("172.31.0.2")}, [28->16]) = 155
1609430221.717890 (+ 0.000469) recvfrom(3,
 "\204\207\201\200\0\1\0\1\0\0\0\0\rnotls20201224\6tihew"...,
    65536, 0, {sa_family=AF_INET, sin_port=htons(53),
 sin_addr=inet_addr("172.31.0.2")}, [28->16]) = 139
1609430221.745659 (+ 0.027772) socket(AF_INET, SOCK_STREAM, IPPROTO_TCP) = 3
1609430221.747548 (+ 0.001887) recvfrom(0, 0x7ffcf2f2ca50, 8192,
    0, 0x7ffcf2f2c9d0, [128]) = -1 ENOTSOCK (Socket operation on non-socket)
1609430221.747858 (+ 0.000308) sendto(3, "ping\n", 5, 0, NULL, 0) = 5
1609430221.748048 (+ 0.000188) recvfrom(0, 0x7ffcf2f2ca50, 8192, 0, 0x7ffcf2f2c9d0,
 [128]) = -1 ENOTSOCK
    (Socket operation on non-socket)
1609430221.748330 (+ 0.000282) recvfrom(3, "+PONG\r\n", 8192, 0, 0x7ffcf2f2c9d0,
 [128->0]) = 7
+PONG
1609430221.748543 (+ 0.000213) recvfrom(3, "", 8192, 0, 0x7ffcf2f2c9d0, [128->0]) = 0
1609430221.752110
    (+ 0.003569) +++ exited with 0 +++
```

Pada contoh di atas, perintah membutuhkan waktu sekitar 54 milidetik untuk diselesaikan (752110 - 697712 = 54398 mikrodetik).

Lama waktu yang signifikan, sekitar 20 ms, dibutuhkan untuk membuat instans nc dan melakukan resolusi nama (dari 697712 hingga 717890), setelah itu, waktu 2 ms diperlukan untuk membuat soket TCP (745659 hingga 747858), dan waktu 0,4 ms (747858 hingga 748330) dibutuhkan untuk mengirimkan dan menerima respons untuk permintaan.

# Keamanan di Amazon ElastiCache

Keamanan cloud di AWS adalah prioritas tertinggi. Sebagai AWS pelanggan, Anda mendapat manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan organisasi yang paling sensitif terhadap keamanan.

Keamanan adalah tanggung jawab bersama antara Anda AWS dan Anda. Model tanggung jawab bersama menjelaskan hal ini sebagai keamanan dari cloud dan keamanan di cloud:

- Keamanan cloud — AWS bertanggung jawab untuk melindungi infrastruktur yang menjalankan AWS layanan di AWS Cloud. AWS juga memberi Anda layanan yang dapat Anda gunakan dengan aman. Auditor pihak ketiga secara berkala menguji dan memverifikasi efektivitas keamanan kami sebagai bagian dari AWS program kepatuhan. Untuk mempelajari tentang program kepatuhan yang berlaku untuk Amazon ElastiCache, lihat AWS Layanan dalam Lingkup berdasarkan Program Kepatuhan.
- Keamanan di cloud — Tanggung jawab Anda ditentukan oleh AWS layanan yang Anda gunakan. Anda juga bertanggung jawab atas faktor lain, yang mencakup sensitivitas data Anda, persyaratan perusahaan Anda, serta undang-undang dan peraturan yang berlaku.

Dokumentasi ini membantu Anda memahami cara menerapkan model tanggung jawab bersama saat menggunakan Amazon ElastiCache. Topik berikut menunjukkan cara mengonfigurasi Amazon ElastiCache untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga mempelajari cara menggunakan AWS layanan lain yang membantu Anda memantau dan mengamankan ElastiCache sumber daya Amazon Anda.

Topik

# Perlindungan data di Amazon ElastiCache

Model tanggung jawab bersama AWS diterapkan ke perlindungan data di AWS ElastiCache (ElastiCache). Sebagaimana diuraikan dalam model ini, AWS bertanggung jawab untuk melindungi infrastruktur global yang menjalankan AWS Cloud secara keseluruhan. Anda harus bertanggung jawab untuk memelihara kendali atas konten yang di-hosting di infrastruktur ini. Konten ini meliputi konfigurasi keamanan dan tugas-tugas pengelolaan untuk berbagai layanan AWS yang Anda gunakan. Untuk informasi selengkapnya tentang privasi data, lihat FAQ privasi data.

Untuk tujuan perlindungan data, sebaiknya Anda melindungi kredensial akun AWS dan menyiapkan akun individual dengan AWS Identity and Access Management (IAM). Dengan cara ini, setiap pengguna hanya diberi izin yang diperlukan untuk memenuhi tanggung jawab tugas mereka. Kami juga merekomendasikan agar Anda mengamankan data Anda dengan cara-cara berikut ini:

- Gunakan autentikasi multi-faktor (MFA) pada setiap akun.
- Gunakan TLS untuk melakukan komunikasi dengan sumber daya AWS.
- Siapkan API dan logging aktivitas pengguna dengan AWS CloudTrail.
- Gunakan solusi enkripsi AWS, bersama dengan semua kontrol keamanan default dalam layanan AWS.
- Gunakan layanan keamanan terkelola lanjutan seperti Amazon Macie, yang membantu menemukan dan mengamankan data pribadi yang disimpan di Amazon S3.

Sebaiknya jangan pernah memasukkan informasi identitas yang sensitif, seperti nomor rekening pelanggan Anda, ke dalam bidang isian bebas seperti bidang Nama. Hal ini termasuk ketika Anda menangani ElastiCache atau layanan AWS lainnya menggunakan konsol, API, AWS CLI, atau AWS SDK. Setiap data yang Anda masukkan ke dalam ElastiCache atau layanan lain mungkin akan diambil untuk disertakan dalam log diagnostik. Saat Anda memberikan URL ke server eksternal, jangan sertakan informasi kredensial di URL untuk memvalidasi permintaan Anda ke server tersebut.

Topik

- Keamanan data di Amazon ElastiCache

## Keamanan data di Amazon ElastiCache

Untuk membantu menjaga keamanan data Anda, Amazon ElastiCache dan Amazon EC2 menyediakan mekanisme untuk mencegah akses yang tidak sah terhadap data Anda di server.

Amazon ElastiCache for Redis menyediakan fitur enkripsi untuk data pada cache yang menjalankan Redis versi 3.2.6 (dijadwalkan untuk EOL, lihat Jadwal akhir masa aktif Redis), 4.0.10 atau yang lebih baru:

- Enkripsi data bergerak mengenkripsi data Anda setiap kali data bergerak dari satu tempat ke tempat lain, misalnya antara simpul di klaster atau antara cache dan aplikasi Anda.
- Enkripsi data diam mengenkripsi data pada disk Anda selama operasi sinkronisasi dan pencadangan.

Amazon ElastiCache for Redis juga mendukung autentikasi pengguna dengan IAM atau Redis AUTH, dan mengotorisasi operasi pengguna menggunakan Kontrol Akses Berbasis Peran (RBAC).

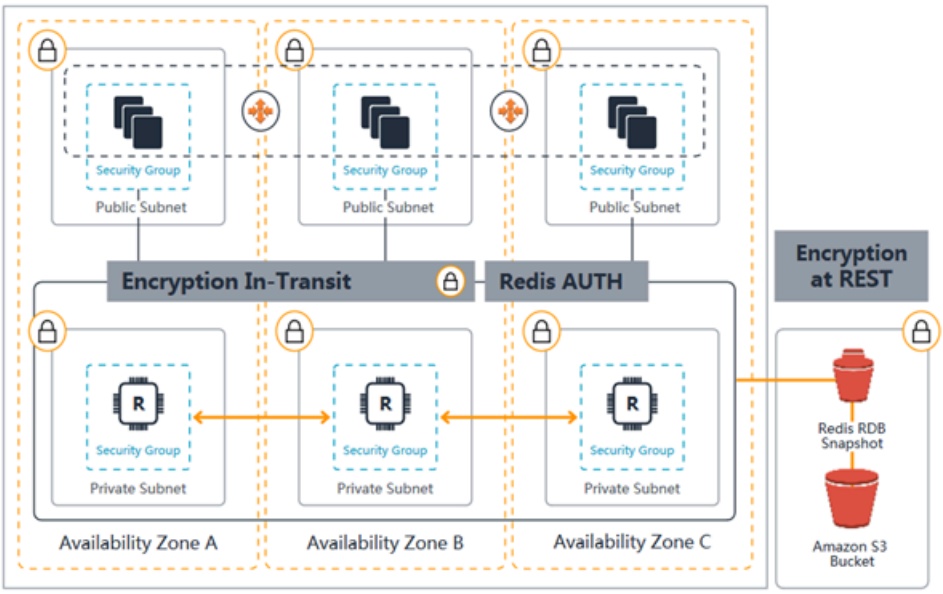



Diagram Keamanan ElastiCache for Redis

Topik

- Enkripsi bergerak ElastiCache (TLS)
- Enkripsi Diam di ElastiCache
- Autentikasi dan Otorisasi

## Enkripsi bergerak ElastiCache (TLS)

Untuk membantu menjaga keamanan data Anda, Amazon ElastiCache dan Amazon EC2 menyediakan mekanisme untuk mencegah akses yang tidak sah terhadap data Anda di server.

Dengan menyediakan kemampuan enkripsi bergerak, ElastiCache menyediakan alat untuk membantu Anda melindungi data saat bergerak dari satu lokasi ke lokasi lain.

Semua cache nirserver memiliki enkripsi bergerak yang aktif. Untuk klaster yang dirancang sendiri, Anda dapat mengaktifkan enkripsi bergerak pada grup replikasi dengan menetapkan parameter `TransitEncryptionEnabled` ke `true` (CLI: `--transit-encryption-enabled`) saat Anda membuat grup replikasi. Anda dapat melakukannya terlepas dari apakah Anda membuat grup replikasi menggunakan AWS Management Console, AWS CLI, atau API ElastiCache.

Topik



## Gambaran umum enkripsi bergerak

Enkripsi bergerak Amazon ElastiCache adalah fitur opsional yang memungkinkan Anda meningkatkan keamanan data Anda pada titik yang paling rentan—saat data dalam keadaan bergerak dari satu lokasi ke lokasi lain. Karena diperlukan beberapa pemrosesan untuk mengenkripsi dan mendekripsi data di titik akhir, mengaktifkan enkripsi bergerak dapat memberikan beberapa dampak pada performa. Anda harus melakukan tolok ukur terhadap data Anda dengan dan tanpa enkripsi bergerak untuk menentukan dampak terhadap performa pada kasus penggunaan Anda.

Enkripsi bergerak ElastiCache menerapkan fitur berikut:

- Koneksi klien terenkripsi—koneksi klien ke simpul cache dienkripsi TLS.
- Koneksi server terenkripsi—data yang bergerak antarsimpul dalam klaster dienkripsi.
- Autentikasi server—Klien dapat mengautentikasi bahwa koneksinya dilakukan ke server yang benar.

- Autentikasi klien—menggunakan fitur AUTH Redis, server dapat mengautentikasi klien.

### Kondisi enkripsi bergerak

Batasan berikut pada enkripsi bergerak Amazon ElastiCache harus Anda ingat saat merencanakan implementasi klaster yang dirancang sendiri:

- Enkripsi bergerak didukung pada grup replikasi yang menjalankan Redis versi 3.2.6, 4.0.10, atau yang lebih baru.
- Modifikasi pada pengaturan enkripsi bergerak, untuk klaster yang sudah ada, didukung pada grup replikasi yang menjalankan Redis versi 7 atau yang lebih baru.
- Enkripsi bergerak didukung hanya untuk grup replikasi yang berjalan di Amazon VPC.
- Enkripsi bergerak hanya didukung untuk grup replikasi yang menjalankan jenis simpul berikut.
  - R6g, R5, R4, R3
  - M6g, M5, M4, M3
  - T4g, T3, T2

  Untuk informasi selengkapnya, lihat Tipe simpul yang didukung.
- Enkripsi bergerak diaktifkan dengan menetapkan parameter `TransitEncryptionEnabled` ke `true` secara eksplisit.
- Pastikan klien cache Anda mendukung konektivitas TLS dan Anda telah mengaktifkannya dalam konfigurasi klien.

### Praktik terbaik enkripsi bergerak

- Karena pemrosesan diperlukan untuk mengenkripsi dan mendekripsi data di titik akhir, penerapan enkripsi bergerak dapat mengurangi performa. Lakukan tolok ukur terhadap enkripsi bergerak dibandingkan dengan tanpa enkripsi pada data Anda sendiri untuk menentukan dampaknya terhadap performa untuk implementasi Anda.
- Karena membuat koneksi baru bisa berbiaya mahal, Anda dapat mengurangi dampak enkripsi bergerak pada performa dengan mempersistensi koneksi TLS Anda.

### Lihat juga

- Enkripsi Diam di ElastiCache

- Autentikasi dengan perintah AUTH Redis
- Mengautentikasi Pengguna dengan Kontrol Akses Berbasis Peran (RBAC)
- Amazon VPC dan keamanan ElastiCache
- Manajemen Identitas dan Akses untuk Amazon ElastiCache

### Mengaktifkan enkripsi bergerak

Semua cache nirserver memiliki enkripsi bergerak yang aktif. Pada klaster yang dirancang sendiri, Anda dapat mengaktifkan enkripsi bergerak menggunakan AWS Management Console, AWS CLI, atau API ElastiCache.

#### Mengaktifkan enkripsi bergerak menggunakan AWS Management Console

##### Mengaktifkan enkripsi bergerak pada klaster baru yang dirancang sendiri menggunakan AWS Management Console

Saat merancang klaster Anda sendiri, konfigurasi 'dev/tes' dan 'Produksi' dengan metode 'Mudah dibuat' akan memiliki enkripsi bergerak yang aktif. Saat memilih konfigurasi sendiri, buat pilihan berikut:

- Pilih versi mesin 3.2.6, 4.0.10 atau yang lebih baru.
- Klik kotak centang di samping Aktifkan untuk opsi Enkripsi saat transit.

Untuk proses langkah demi langkah, lihat yang berikut ini:

- Membuat klaster Redis (Mode Klaster Dinonaktifkan) (Konsol)
- Membuat klaster Redis (mode klaster diaktifkan) (Konsol)

##### Mengaktifkan enkripsi bergerak pada klaster yang dirancang sendiri yang sudah ada menggunakan AWS Management Console

Pengaktifan enkripsi bergerak adalah proses dua langkah. Anda harus terlebih dahulu mengatur mode enkripsi bergerak ke `preferred`. Mode ini memungkinkan klien Redis Anda terhubung menggunakan koneksi terenkripsi dan tidak terenkripsi. Setelah memigrasikan semua klien Redis untuk menggunakan koneksi terenkripsi, Anda kemudian dapat mengubah konfigurasi klaster untuk mengatur mode enkripsi bergerak ke `required`. Pengaturan mode enkripsi bergerak ke `required`

akan menghentikan semua koneksi yang tidak terenkripsi dan hanya akan mengizinkan koneksi terenkripsi.

Langkah 1: Atur Mode enkripsi transit ke Pilihan

1. Masuk ke AWS Management Console dan buka konsol Amazon ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Pilih Cache Redis dari Sumber Daya ElastiCache yang tercantum di panel navigasi, yang terdapat di sebelah kiri.
3. Pilih Cache Redis yang ingin Anda perbarui.
4. Pilih drop-down Tindakan, lalu pilih Ubah.
5. Pilih Aktifkan pada Enkripsi saat transit di bagian Keamanan.
6. Klik Pilihan sebagai Mode enkripsi transit.
7. Pilih Pratinjau perubahan dan simpan perubahan Anda.

Setelah memigrasikan semua klien Redis Anda untuk menggunakan koneksi terenkripsi:

Langkah 2: Atur Mode enkripsi transit ke Wajib

1. Masuk ke AWS Management Console dan buka konsol Amazon ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Pilih Cache Redis dari Sumber Daya ElastiCache yang tercantum di panel navigasi, yang terdapat di sebelah kiri.
3. Pilih Cache Redis yang ingin Anda perbarui.
4. Pilih drop-down Tindakan, lalu pilih Ubah.
5. Pilih Wajib sebagai Mode enkripsi transit, di bagian Keamanan.
6. Pilih Pratinjau perubahan dan simpan perubahan Anda.

Mengaktifkan enkripsi bergerak menggunakan AWS CLI

Untuk mengaktifkan enkripsi bergerak saat membuat grup replikasi Redis menggunakan AWS CLI, gunakan parameter `transit-encryption-enabled`.

Mengaktifkan enkripsi bergerak pada klaster baru yang dirancang sendiri untuk Redis (Mode Klaster Dinonaktifkan) (CLI)

Gunakan operasi `create-replication-group` AWS CLI dan parameter berikut untuk membuat grup replikasi Redis dengan replika yang memiliki enkripsi bergerak yang aktif:

Parameter kunci:

- **`--engine`**—Harus `redis`.
- **`--engine-version`**—Harus 3.2.6, 4.0.10, atau yang lebih baru.
- **`--transit-encryption-enabled`**—Wajib. Jika Anda mengaktifkan enkripsi bergerak, Anda juga harus menyediakan nilai untuk parameter `--cache-subnet-group`.
- **`--num-cache-clusters`**—Minimal bernilai 1. Nilai maksimum untuk parameter ini adalah enam.

Untuk informasi selengkapnya tentang IAM, lihat hal berikut:

- Membuat grup replikasi Redis (Mode Klaster Dinonaktifkan) dari awal (AWS CLI)
- create-replication-group

Mengaktifkan enkripsi bergerak pada klaster baru yang dirancang sendiri untuk Redis (Mode Klaster Diaktifkan) (CLI)

Gunakan operasi `create-replication-group` AWS CLI dan parameter berikut untuk membuat grup replikasi Redis (mode klaster diaktifkan) yang memiliki enkripsi bergerak yang aktif:

Parameter kunci:

- **`--engine`**—Harus `redis`.
- **`--engine-version`**—Harus 3.2.6, 4.0.10, atau yang lebih baru.
- **`--transit-encryption-enabled`**—Wajib. Jika Anda mengaktifkan enkripsi bergerak, Anda juga harus menyediakan nilai untuk parameter `--cache-subnet-group`.
- Gunakan salah satu set parameter berikut untuk menentukan konfigurasi grup simpul dari grup replikasi:
  - **`--num-node-groups`**—Menentukan jumlah serpihan (grup simpul) dalam grup replikasi ini. Nilai maksimum untuk parameter ini adalah 500.

**`--replicas-per-node-group`**—Menentukan jumlah simpul replika di setiap grup simpul. Nilai yang ditentukan di sini berlaku untuk semua serpihan di dalam grup replikasi ini. Nilai maksimum untuk parameter ini adalah 5.

- **`--node-group-configuration`**—Menentukan konfigurasi dari setiap serpihan secara independen.

Untuk informasi selengkapnya tentang IAM, lihat hal berikut:

- Membuat grup replikasi di Redis (Mode Klaster Diaktifkan) dari awal (AWS CLI)
- create-replication-group

Mengaktifkan enkripsi bergerak untuk klaster yang sudah ada menggunakan AWS CLI

Pengaktifan enkripsi bergerak adalah proses dua langkah. Anda harus terlebih dahulu mengatur mode enkripsi bergerak ke `preferred`. Mode ini memungkinkan klien Redis Anda terhubung menggunakan koneksi terenkripsi dan tidak terenkripsi. Setelah memigrasikan semua klien Redis untuk menggunakan koneksi terenkripsi, Anda kemudian dapat mengubah konfigurasi klaster untuk mengatur mode enkripsi bergerak ke `required`. Pengaturan mode enkripsi bergerak ke `required` akan menghentikan semua koneksi yang tidak terenkripsi dan hanya akan mengizinkan koneksi terenkripsi.

Gunakan operasi `modify-replication-group` AWS CLI dan parameter berikut untuk memperbarui grup replikasi Redis (mode klaster diaktifkan) yang memiliki enkripsi bergerak yang nonaktif.

Untuk mengaktifkan enkripsi bergerak

1. Atur transit-encryption-mode ke `preferred`, menggunakan parameter berikut
   - **`--transit-encryption-enabled`**—Wajib.
   - **`--transit-encryption-mode`**—Harus diatur ke `preferred`.
2. Atur transit-encryption-mode ke `required`, menggunakan parameter berikut:
   - **`--transit-encryption-enabled`**—Wajib.
   - **`--transit-encryption-mode`**—Harus diatur ke `required`.

### Menghubungkan ke Amazon ElastiCache untuk Redis dengan enkripsi dalam perjalanan menggunakan redis-cli

Untuk mengakses data dari ElastiCache cache Redis yang diaktifkan dengan enkripsi dalam perjalanan, Anda menggunakan klien yang bekerja dengan Secure Socket Layer (SSL). Anda juga dapat menggunakan redis-cli dengan TLS/SSL di Amazon Linux dan Amazon Linux 2. Jika klien Anda tidak mendukung TLS, Anda dapat menggunakan perintah `stunnel` di host klien Anda untuk membuat tunnel SSL ke simpul Redis.

#### Koneksi terenkripsi dengan Linux

Untuk menggunakan redis-cli untuk terhubung ke kluster Redis yang diaktifkan dengan enkripsi dalam transit di Amazon Linux 2023, Amazon Linux 2, atau Amazon Linux, ikuti langkah-langkah berikut.

1. Unduh dan kompilasi utilitas redis-cli. Utilitas ini disertakan dalam distribusi perangkat lunak Redis.
2. Pada prompt perintah instans EC2 Anda, ketikkan perintah yang sesuai untuk versi Linux yang Anda gunakan.

   Amazon Linux 2023

   Jika menggunakan Amazon Linux 2023, masukkan ini:

   ```
   sudo yum install redis6 -y
   ```

   Kemudian ketik perintah berikut, ganti titik akhir cluster dan port Anda dengan apa yang ditampilkan dalam contoh ini.

   ```
   redis-cli -h Primary or Configuration Endpoint --tls -p 6379
   ```

   Untuk informasi selengkapnya tentang cara mencari titik akhir, lihat Menemukan Titik Akhir Simpul Anda.

   Amazon Linux 2

   Jika menggunakan Amazon Linux 2, masukkan ini:

   ```
   sudo yum -y install openssl-devel gcc
   wget http://download.redis.io/redis-stable.tar.gz
   ```

```
tar xvzf redis-stable.tar.gz
cd redis-stable
make distclean
make redis-cli BUILD_TLS=yes
sudo install -m 755 src/redis-cli /usr/local/bin/
```

Amazon Linux

Jika menggunakan Amazon Linux, masukkan ini:

```
sudo yum install gcc jemalloc-devel openssl-devel tcl tcl-devel clang wget
wget http://download.redis.io/redis-stable.tar.gz
tar xvzf redis-stable.tar.gz
cd redis-stable
make redis-cli CC=clang BUILD_TLS=yes
sudo install -m 755 src/redis-cli /usr/local/bin/
```

Di Amazon Linux, Anda mungkin perlu menjalankan langkah tambahan berikut:

```
sudo yum install clang
CC=clang make
sudo make install
```

3. Setelah Anda mengunduh dan menginstal utilitas redis-cli, Anda disarankan untuk menjalankan perintah opsional. `make-test`
4. Untuk terhubung ke cluster dengan enkripsi dan otentikasi diaktifkan, masukkan perintah ini:

```
redis-cli -h Primary or Configuration Endpoint --tls -a 'your-password' -p 6379
```

> **Note**
>
> Jika Anda menginstal redis6 di Amazon Linux 2023, Anda sekarang dapat menggunakan `redis6-cli` perintah alih-alih: `redis-cli`
>
> ```
> redis6-cli -h Primary or Configuration Endpoint --tls -p 6379
> ```

### Koneksi terenkripsi dengan stunnel

Untuk menggunakan redis-cli untuk terhubung ke kluster Redis yang diaktifkan dengan enkripsi dalam transit menggunakan stunnel, ikuti langkah-langkah berikut.

1. Gunakan SSH untuk menyambungkan ke klien Anda dan instal `stunnel`.

   ```
   sudo yum install stunnel
   ```

2. Jalankan perintah berikut untuk membuat dan mengedit file `'/etc/stunnel/redis-cli.conf'` secara bersamaan untuk menambahkan titik akhir cluster Redis untuk satu atau beberapa parameter koneksi, menggunakan output yang disediakan di bawah ini sebagai template. ElastiCache

   ```
   vi /etc/stunnel/redis-cli.conf


   fips = no
   setuid = root
   setgid = root
   pid = /var/run/stunnel.pid
   debug = 7
   delay = yes
   options = NO_SSLv2
   options = NO_SSLv3
   [redis-cli]
      client = yes
      accept = 127.0.0.1:6379
      connect = primary.ssltest.wif01h.use1.cache.amazonaws.com:6379
   [redis-cli-replica]
      client = yes
      accept = 127.0.0.1:6380
      connect = ssltest-02.ssltest.wif01h.use1.cache.amazonaws.com:6379
   ```

   Dalam contoh ini, file config memiliki dua koneksi, `redis-cli` dan `redis-cli-replica`. Parameternya ditetapkan sebagai berikut:

   - client ditetapkan ke ya untuk menentukan bahwa instans stunnel ini adalah klien.
   - accept ditetapkan ke IP klien. Pada contoh ini, primer ditetapkan ke default Redis 127.0.0.1 pada port 6379. Replika harus memanggil port yang berbeda dan ditetapkan ke 6380. Anda

dapat menggunakan port sementara 1024–65535. Untuk informasi selengkapnya, lihat Port sementara di Panduan Pengguna Amazon VPC.

- connect ditetapkan ke titik akhir server Redis. Untuk informasi selengkapnya, lihat Menemukan titik akhir koneksi.

3. Mulai `stunnel`.

```
sudo stunnel /etc/stunnel/redis-cli.conf
```

Gunakan perintah `netstat` untuk mengonfirmasi bahwa tunnel dimulai.

```
sudo netstat -tulnp | grep -i stunnel

tcp        0      0 127.0.0.1:6379              0.0.0.0:*                   LISTEN
     3189/stunnel
tcp        0      0 127.0.0.1:6380              0.0.0.0:*                   LISTEN
     3189/stunnel
```

4. Menghubungkan ke simpul Redis terenkripsi menggunakan titik akhir terowongan lokal.

- Jika tidak ada kata sandi AUTH yang digunakan ElastiCache selama pembuatan klaster Redis, contoh ini menggunakan redis-cli untuk terhubung ke server for Redis menggunakan jalur lengkap ElastiCache untuk redis-cli, di Amazon Linux:

```
/home/ec2-user/redis-stable/src/redis-cli -h localhost -p 6379
```

Jika kata sandi AUTH digunakan selama pembuatan klaster Redis, contoh ini menggunakan redis-cli untuk menyambung ke server Redis menggunakan jalur lengkap untuk redis-cli, di Amazon Linux:

```
 /home/ec2-user/redis-stable/src/redis-cli -h localhost -p 6379 -a my-secret-
password
```

ATAU

- Ubah direktori ke redis-stable dan lakukan hal berikut:

Jika tidak ada kata sandi AUTH yang digunakan ElastiCache selama pembuatan klaster Redis, contoh ini menggunakan redis-cli untuk terhubung ke server for Redis menggunakan jalur lengkap ElastiCache untuk redis-cli, di Amazon Linux:

```
src/redis-cli -h localhost -p 6379
```

Jika kata sandi AUTH digunakan selama pembuatan klaster Redis, contoh ini menggunakan redis-cli untuk terhubung ke server Redis menggunakan jalur lengkap untuk redis-cli, di Amazon Linux:

```
src/redis-cli -h localhost -p 6379 -a my-secret-password
```

Contoh ini menggunakan Telnet untuk terhubung ke server Redis.

```
telnet localhost 6379

Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
auth MySecretPassword
+OK
get foo
$3
bar
```

5. Untuk menghentikan dan menutup terowongan SSL, `pkill` proses stunnel.

```
sudo pkill stunnel
```

## Mengaktifkan enkripsi data bergerak pada Klaster Redis yang dirancang sendiri menggunakan Python

Panduan berikut menunjukkan cara mengaktifkan enkripsi data bergerak pada klaster Redis 7.0 yang awalnya dibuat dengan enkripsi data bergerak nonaktif. Klien TCP dan TLS akan terus berkomunikasi dengan klaster selama proses ini tanpa waktu henti.

Boto3 akan mendapatkan kredensial yang dibutuhkannya (`aws_access_key_id`,`aws_secret_access_key`, dan`aws_session_token`) dari variabel lingkungan. Kredensial tersebut akan ditempelkan terlebih dahulu di terminal bash yang sama di tempat kita menjalankan `python3` untuk memproses kode Python yang ditunjukkan dalam panduan ini. Kode dalam contoh di bawah ini adalah proses dari instans EC2 yang diluncurkan di VPC yang sama yang akan digunakan untuk membuat Klaster Redis ElastiCache di dalamnya.

> **Note**
>
> - Contoh berikut menggunakan SDK boto3 untuk operasi pengelolaan ElastiCache (klaster atau pembuatan pengguna) dan redis-py/redis-py-cluster untuk penanganan data.
> - Anda harus menggunakan setidaknya boto3 versi (=~) 1.26.39 untuk menggunakan migrasi TLS online dengan API modifikasi klaster.
> - ElastiCache mendukung migrasi TLS online hanya untuk Klaster Redis dengan versi 7.0 atau lebih tinggi. Jadi, jika Anda memiliki klaster yang menjalankan versi Redis lebih lama dari 7.0, Anda harus memutakhirkan versi Redis klaster Anda. Untuk informasi selengkapnya tentang perbedaa versi, lihat Perilaku versi utama dan perbedaan kompatibilitas.

Topik



### Tentukan konstanta string yang akan meluncurkan Klaster Redis ElastiCache

Pertama, mari kita mendefinisikan beberapa konstanta string Python sederhana yang akan berisi nama-nama entitas AWS yang diperlukan untuk membuat klaster ElastiCache seperti `security-group`, `Cache Subnet group`, dan `default parameter group`. Semua entitas AWS ini harus dibuat terlebih dahulu di akun AWS Anda di Wilayah yang ingin Anda gunakan.

```
#Constants definitions
```

```
SECURITY_GROUP = "sg-0492aa0a29c558427"
CLUSTER_DESCRIPTION = "This cluster has been launched as part of the online TLS
 migration user guide"
EC_SUBNET_GROUP = "client-testing"
DEFAULT_PARAMETER_GROUP_REDIS_7_CLUSTER_MODE_ENABLED = "default.redis7.cluster.on"
```

Tentukan kelas untuk konfigurasi klaster

Sekarang, mari kita definisikan beberapa kelas Python sederhana yang akan mewakili konfigurasi klaster, yang akan menyimpan metadata tentang klaster seperti versi Redis, jenis instans, dan apakah enkripsi data bergerak (TLS) diaktifkan atau dinonaktifkan.

```
#Class definitions

class Config:
    def __init__(
        self,
        instance_type: str = "cache.t4g.small",
        version: str = "7.0",
        multi_az: bool = True,
        TLS: bool = True,
        name: str = None,
    ):
        self.instance_type = instance_type
        self.version = version
        self.multi_az = multi_az
        self.TLS = TLS
        self.name = name or f"tls-test"

    def create_base_launch_request(self):
        return {
            "ReplicationGroupId": self.name,
            "TransitEncryptionEnabled": self.TLS,
            "MultiAZEnabled": self.multi_az,
            "CacheNodeType": self.instance_type,
            "Engine": "redis",
            "EngineVersion": self.version,
            "CacheSubnetGroupName": EC_SUBNET_GROUP ,
            "CacheParameterGroupName":
 DEFAULT_PARAMETER_GROUP_REDIS_7_CLUSTER_MODE_ENABLED ,
            "ReplicationGroupDescription": CLUSTER_DESCRIPTION,
            "SecurityGroupIds": [SECURITY_GROUP],
        }
```

```
class ConfigCME(Config):
    def __init__(
        self,
        instance_type: str = "cache.t4g.small",
        version: str = "7.0",
        multi_az: bool = True,
        TLS: bool = True,
        name: str = None,
        num_shards: int = 2,
        num_replicas_per_shard: int = 1,
    ):
        super().__init__(instance_type, version, multi_az, TLS, name)
        self.num_shards = num_shards
        self.num_replicas_per_shard = num_replicas_per_shard

    def create_launch_request(self) -> dict:
        launch_request = self.create_base_launch_request()
        launch_request["NumNodeGroups"] = self.num_shards
        launch_request["ReplicasPerNodeGroup"] = self.num_replicas_per_shard
        return launch_request
```

Mendefinisikan kelas yang akan mewakili klaster itu sendiri

Sekarang, mari kita mendefinisikan beberapa kelas Python sederhana yang akan mewakili Klaster Redis ElastiCache itu sendiri. Kelas ini akan memiliki bidang klien yang akan menampung klien boto3 untuk operasi manajemen ElastiCache seperti pembuatan klaster dan kueri API ElastiCache.

```
import botocore.config
import boto3

# Create boto3 client
def init_client(region: str = "us-east-1"):
    config = botocore.config.Config(retries={"max_attempts": 10, "mode": "standard"})
    init_request = dict()
    init_request["config"] = config
    init_request["service_name"] = "elasticache"
    init_request["region_name"] = region
    return boto3.client(**init_request)


class ElastiCacheClusterBase:
    def __init__(self, name: str):
```

```
        self.name = name
        self.elasticache_client = init_client()

    def get_first_replication_group(self):
        return self.elasticache_client.describe_replication_groups(
        ReplicationGroupId=self.name
        )["ReplicationGroups"][0]

    def get_status(self) -> str:
        return self.get_first_replication_group()["Status"]

    def get_transit_encryption_enabled(self) -> bool:
        return self.get_first_replication_group()["TransitEncryptionEnabled"]

    def is_available(self) -> bool:
        return self.get_status() == "available"

    def is_modifying(self) -> bool:
        return self.get_status() == "modifying"

    def wait_for_available(self):
        while True:
            if self.is_available():
                break
            else:
                time.sleep(5)

    def wait_for_modifying(self):
        while True:
            if self.is_modifying():
                break
            else:
                time.sleep(5)

    def delete_cluster(self) -> bool:
        self.elasticache_client.delete_replication_group(
            ReplicationGroupId=self.name, RetainPrimaryCluster=False
        )

    def modify_transit_encryption_mode(self, new_transit_encryption_mode: str):
        # generate api call to migrate the cluster to TLS preffered or to TLS required
            self.elasticache_client.modify_replication_group(
                ReplicationGroupId=self.name,
                TransitEncryptionMode=new_transit_encryption_mode,
```

```
                TransitEncryptionEnabled=True,
                ApplyImmediately=True,
            )
        self.wait_for_modifying()

 class ElastiCacheClusterCME(ElastiCacheClusterBase):
    def __init__(self, name: str):
        super().__init__(name)

    @classmethod
    def launch(cls, config: ConfigCME = None) -> ElastiCacheClusterCME:
        config = config or ConfigCME()
        print(config)
        new_cluster = ElastiCacheClusterCME(config.name)
        launch_request = config.create_launch_request()
        new_cluster.elasticache_client.create_replication_group(**launch_request)
        new_cluster.wait_for_available()
        return new_cluster

    def get_configuration_endpoint(self) -> str:
        return self.get_first_replication_group()["ConfigurationEndpoint"]["Address"]

#Since the code can throw exceptions, we define this class to make the code more
 readable and
#so we won't forget to delete the cluster
class ElastiCacheCMEManager:
    def __init__(self, config: ConfigCME = None):
        self.config = config or ConfigCME()

    def __enter__(self) -> ElastiCacheClusterCME:
        self.cluster = ElastiCacheClusterCME.launch(self.config)
        return self.cluster

    def __exit__(self, exc_type, exc_val, exc_tb):
        self.cluster.delete_cluster()
```

(Opsional) Membuat kelas pembungkus untuk demo koneksi klien ke klaster Redis

Sekarang, mari kita buat kelas pembungkus untuk klien `redis-py-cluster`. Kelas pembungkus ini akan mendukung pra-pengisian klaster dengan beberapa kunci lalu melakukan perintah `get` acak berulang.

> **Note**
>
> Ini adalah langkah opsional tetapi menyederhanakan kode fungsi utama yang muncul pada langkah selanjutnya.

```
import redis
improt random
from time import perf_counter_ns, time


class DowntimeTestClient:
    def __init__(self, client):
        self.client = client

        # num of keys prefilled
        self.prefilled = 0
        # percent of get above prefilled
        self.percent_get_above_prefilled = 10 # nil result expected when get hit above
 prefilled
        # total downtime in nano seconds
        self.downtime_ns = 0
        # num of success and fail operations
        self.success_ops = 0
        self.fail_ops = 0
        self.connection_errors = 0
        self.timeout_errors = 0


    def replace_client(self, client):
        self.client = client

    def prefill_data(self, timelimit_sec=60):
        end_time = time() + timelimit_sec
        while time() < end_time:
            self.client.set(self.prefilled, self.prefilled)
            self.prefilled += 1

    # unsuccesful operations throw exceptions
    def _exec(self, func):
        try:
            start_ns = perf_counter_ns()
```

```
            func()
            self.success_ops += 1
            elapsed_ms = (perf_counter_ns() - start_ns) // 10 ** 6
            # upon succesful execution of func
            # reset random_key to None so that the next command
            # will use a new random key
            self.random_key = None

        except Exception as e:
            elapsed_ns = perf_counter_ns() - start_ns
            self.downtime_ns += elapsed_ns
            # in case of failure- increment the relevant counters so that we will keep
track
            # of how many connection issues we had while trying to communicate with
            # the cluster.
            self.fail_ops += 1
            if e.__class__ is redis.exceptions.ConnectionError:
                self.connection_errors += 1
            if e.__class__ is redis.exceptions.TimeoutError:
                self.timeout_errors += 1

    def _repeat_exec(self, func, seconds):
        end_time = time() + seconds
        while time() < end_time:
            self._exec(func)

    def _new_random_key_if_needed(self, percent_above_prefilled):
        if self.random_key is None:
            max = int((self.prefilled * (100 + percent_above_prefilled)) / 100)
            return random.randint(0, max)
        return self.random_key

    def _random_get(self):
        key = self._new_random_key_if_needed(self.percent_get_above_prefilled)
        result = self.client.get(key)
        # we know the key was set for sure only in the case key < self.prefilled
        if key < self.prefilled:
            assert result.decode("UTF-8") == str(key)


    def repeat_get(self, seconds=60):
        self._repeat_exec(self._random_get, seconds)

    def get_downtime_ms(self) -> int:
```

```
        return self.downtime_ns // 10 ** 6


    def do_get_until(self, cond_check):
        while not cond_check():
            self.repeat_get()
        # do one more get cycle once condition is met
        self.repeat_get()
```

Membuat fungsi utama yang mendemonstrasikan proses pengubahan konfigurasi enkripsi data bergerak

Sekarang, mari kita definisikan fungsi utama, yang akan melakukan hal berikut:

1. Buat cluster menggunakan klien boto3 ElastiCache.
2. Inisialisasi klien `redis-py-cluster` yang akan terhubung ke klaster dengan koneksi TCP yang jelas tanpa TLS.
3. klien `redis-py-cluster` mengisi klaster dengan beberapa data.
4. Klien boto3 akan memicu migrasi TLS dari tanpa TLS ke TLS diutamakan.
5. Sementara klaster sedang dimigrasikan ke TLS `Preferred`, klien `redis-py-cluster` TCP akan mengirim operasi `get` berulang ke klaster sampai migrasi selesai.
6. Setelah migrasi ke TLS `Preferred` selesai, kita akan memastikan bahwa klaster mendukung enkripsi data bergerak. Setelah itu, kita akan membuat klien `redis-py-cluster` yang akan tersambung ke klaster dengan TLS.
7. Kami akan mengirim beberapa perintah `get` menggunakan klien TLS baru dan klien TCP lama.
8. Klien boto3 akan memicu migrasi TLS dari TLS `Preferred` ke TLS diperlukan.
9. Sementara klaster sedang dimigrasikan ke TLS diperlukan, klien TLS redis-py-cluster akan mengirim operasi `get` berulang ke klaster hingga migrasi selesai.

```
import redis

def init_cluster_client(
    cluster: ElastiCacheClusterCME, prefill_data: bool, TLS: bool = True) ->
 DowntimeTestClient:
    # we must use for the host name the cluster configuration endpoint.
    redis_client = redis.RedisCluster(
        host=cluster.get_configuration_endpoint(), ssl=TLS, socket_timeout=0.25,
 socket_connect_timeout=0.1
```

```
    )
    test_client = DowntimeTestClient(redis_client)
    if prefill_data:
        test_client.prefill_data()
    return test_client

if __name__ == '__main__':
    config = ConfigCME(TLS=False, instance_type="cache.m5.large")

    with ElastiCacheCMEManager(config) as cluster:
        # create a client that will connect to the cluster with clear tcp connection
        test_client_tcp = init_cluster_client(cluster, prefill_data=True, TLS=False)

       # migrate the cluster to TLS Preferred
        cluster.modify_transit_encryption_mode(new_transit_encryption_mode="preferred")

        # do repeated get commands until the cluster finishes the migration to TLS
 Preferred
        test_client_tcp.do_get_until(cluster.is_available)

      # verify that in transit encryption is enabled so that clients will be able to
 connect to the cluster with TLS
        assert cluster.get_transit_encryption_enabled() == True

      # create a client that will connect to the cluster with TLS connection.
        # we must first make sure that the cluster indeed supports TLS
        test_client_tls = init_cluster_client(cluster, prefill_data=True, TLS=True)

        # by doing get commands with the tcp client for 60 more seconds
      # we can verify that the existing tcp connection to the cluster still works
        test_client_tcp.repeat_get(seconds=60)

        # do get commands with the new TLS client for 60 more seconds
        test_client_tcp.repeat_get(seconds=60)

      # migrate the cluster to TLS required
        cluster.modify_transit_encryption_mode(new_transit_encryption_mode="required")

      # from this point the tcp clients will be disconnected and we must not use them
 anymore.
      # do get commands with the TLS client until the cluster finishes migartion to
 TLS required mode.
        test_client_tls.do_get_until(cluster.is_available)
```

Praktik terbaik saat mengaktifkan enkripsi bergerak

Sebelum mengaktifkan enkripsi bergerak: pastikan Anda memiliki penanganan catatan DNS yang tepat

> **Note**
>
> Kita akan mengubah dan menghapus titik akhir lama selama proses ini. Penggunaan titik akhir yang salah dapat mengakibatkan klien Redis menggunakan titik akhir yang lama dan sudah dihapus sehingga akan mencegah koneksi ke klaster.

Saat klaster sedang dimigrasikan dari mode tanpa TLS ke mode TLS pilihan, catatan DNS per simpul yang lama akan dipertahankan dan catatan DNS per simpul yang baru akan dihasilkan dalam format yang berbeda. Klaster yang memiliki TLS aktif menggunakan format catatan DNS yang berbeda dari klaster yang tidak mendukung TLS. ElastiCache akan menyimpan kedua catatan DNS ini saat klaster dikonfigurasi dalam mode enkripsi: Pilihan sehingga Aplikasi dan Klien Redis lain dapat beralih di antara keduanya. Perubahan dalam catatan DNS berikut terjadi selama proses migrasi TLS:

Deskripsi perubahan dalam catatan DNS yang terjadi saat mengaktifkan enkripsi bergerak

Untuk klaster CME

Ketika sebuah klaster diatur ke 'mode enkripsi transit: pilihan':

- Titik akhir klaster asli untuk klaster tanpa TLS aktif akan tetap aktif. Tidak akan ada waktu henti ketika klaster dikonfigurasi ulang dari mode enkripsi TLS 'tidak ada' ke 'pilihan'.
- Titik akhir Redis TLS baru akan dihasilkan saat klaster diatur ke mode TLS pilihan. Titik akhir baru ini akan diresolusi ke IP yang sama dengan yang lama (non-TLS).
- Titik akhir konfigurasi Redis TLS yang baru akan diekspos di Konsol ElastiCache dan sebagai respons terhadap API `describe-replication-group`.

Ketika sebuah klaster diatur ke 'mode enkripsi transit: wajib':

- Titik akhir lama tanpa TLS aktif akan dihapus. Tidak akan ada waktu henti pada titik akhir klaster TLS.
- Anda dapat mengambil `cluster-configuration-endpoint` yang baru dari Konsol ElastiCache atau dari API `describe-replication-group`.

Untuk klaster CMD dengan Failover Otomatis aktif atau Failover Otomatis nonaktif

Ketika grup replikasi diatur ke 'mode enkripsi transit: pilihan':

- Titik akhir primer asli dan titik akhir pembaca untuk klaster tanpa TLS aktif akan tetap aktif.
- Titik akhir primer dan pembaca TLS baru akan dihasilkan saat klaster diatur ke mode TLS `Preferred`. Titik akhir baru ini akan diresolusi ke IP yang sama dengan yang lama (non-TLS).
- Titik akhir primer dan titik akhir pembaca yang baru akan diekspos di Konsol ElastiCache dan sebagai respons terhadap API `describe-replication-group`.

Ketika grup replikasi diatur ke 'mode enkripsi transit: wajib':

- Titik akhir primer dan titik akhir pembaca yang asli untuk klaster tanpa TLS aktif akan tetap aktif.
- Titik akhir primer dan pembaca non-TLS lama akan dihapus. Tidak akan ada waktu henti pada titik akhir klaster TLS.
- Anda dapat mengambil titik akhir primer dan pembaca yang baru dari Konsol ElastiCache atau dari API `describe-replication-group`.

Penggunaan catatan DNS yang disarankan

Untuk klaster CME

- Gunakan titik akhir konfigurasi klaster, bukan catatan DNS per simpul dalam kode aplikasi Anda. Penggunaan nama DNS per simpul secara langsung tidak disarankan karena nama DNS tersebut mungkin akan berubah saat menambahkan atau menghapus serpihan.
- Jangan melakukan hardcoding terhadap titik akhir konfigurasi klaster di aplikasi Anda karena titik akhir tersebut akan berubah selama proses ini.
- Memiliki titik akhir konfigurasi klaster yang di-hardcoding di aplikasi Anda adalah praktik yang buruk karena titik akhir tersebut dapat berubah selama proses ini. Setelah enkripsi bergerak selesai, kueri titik akhir konfigurasi klaster dengan API `describe-replication-group` (seperti yang ditunjukkan di atas (dalam cetak tebal)) dan selanjutnya, gunakan DNS yang Anda dapatkan dalam respons kueri ini.

Untuk klaster CMD dengan Failover Otomatis aktif

- Gunakan titik akhir primer dan titik akhir pembaca, bukan nama DNS per simpul dalam kode aplikasi Anda karena nama DNS per simpul yang lama dihapus dan yang baru akan dihasilkan

saat memigrasikan klaster dari mode tanpa TLS ke mode TLS pilihan. Penggunaan nama DNS per simpul secara langsung tidak direkomendasikan karena Anda mungkin akan menambahkan replika ke klaster Anda nanti. Selain itu, saat Failover Otomatis diaktifkan, peran klaster primer dan replika diubah secara otomatis oleh layanan ElastiCache, dan penggunaan titik akhir primer dan titik akhir pembaca disarankan untuk membantu Anda melacak perubahan tersebut. Terakhir, penggunaan titik akhir pembaca akan membantu Anda mendistribusikan proses baca Anda di antara replika dalam klaster secara merata.

- Memiliki titik akhir primer dan titik akhir pembaca yang di-hardcoding di aplikasi Anda adalah praktik yang buruk karena titik akhir tersebut dapat berubah selama proses migrasi TLS. Setelah perubahan migrasi ke TLS pilihan selesai, kueri titik akhir primer dan titik akhir pembaca dengan API describe-replication-group dan selanjutnya, gunakan DNS yang Anda dapatkan dalam respons kueri ini. Dengan cara ini, Anda dapat melacak perubahan titik akhir secara dinamis.

Untuk klaster CMD dengan Failover Otomatis nonaktif

- Gunakan titik akhir primer dan titik akhir pembaca, bukan nama DNS per simpul dalam kode aplikasi Anda. Saat Failover Otomatis dinonaktifkan, penskalaan, patching, failover, dan prosedur lain yang dikelola secara otomatis oleh layanan ElastiCache saat Failover Otomatis diaktifkan perlu Anda lakukan secara manual. Hal ini memudahkan Anda untuk melacak titik akhir yang berbeda-beda secara manual. Karena nama DNS per simpul yang lama dihapus dan yang baru akan dihasilkan saat memigrasikan klaster dari mode tanpa TLS ke mode TLS pilihan, jangan gunakan nama DNS per simpul secara langsung. Hal ini wajib dilakukan agar klien dapat terhubung ke klaster selama migrasi TLS. Selain itu, Anda akan mendapatkan manfaat dengan menyebarkan permintaan baca secara merata di antara replika saat menggunakan titik akhir pembaca dan melacak catatan DNS saat menambahkan atau menghapus replika dari klaster.
- Memiliki titik akhir konfigurasi klaster yang di-hardcoding di aplikasi Anda adalah praktik yang buruk karena titik akhir tersebut dapat berubah selama proses migrasi TLS.

Selama enkripsi bergerak: perhatikan kapan proses migrasi selesai

Perubahan mode enkripsi bergerak tidak diterapkan segera dan dapat memakan waktu. Hal ini terutama berlaku untuk klaster besar. Hanya setelah menyelesaikan migrasi ke mode TLS pilihan, klaster ini dapat menerima dan melayani koneksi TCP dan TLS. Oleh karena itu, Anda sebaiknya tidak membuat klien yang akan mencoba membuat koneksi TLS ke klaster sampai enkripsi bergerak selesai.

Ada beberapa cara untuk mendapatkan notifikasi ketika enkripsi bergerak berhasil diselesaikan atau gagal: (Tidak ditampilkan dalam contoh kode di atas):

- Menggunakan layanan SNS untuk mendapatkan notifikasi saat enkripsi selesai
- Menggunakan API `describe-events` yang akan menerbitkan peristiwa saat enkripsi selesai
- Melihat pesan di Konsol ElastiCache bahwa enkripsi selesai

Anda juga dapat menerapkan logika dalam aplikasi Anda untuk mengetahui apakah enkripsi selesai. Pada contoh di atas, kita melihat beberapa cara untuk memastikan klaster menyelesaikan migrasi:

- Menunggu hingga proses migrasi dimulai (status klaster berubah menjadi "mengubah"), dan menunggu hingga modifikasi selesai (status klaster berubah kembali ke "tersedia")
- memastikan bahwa klaster telah mengatur `transit_encryption_enabled` ke True dengan mengkueri API `describe-replication-group`.

Setelah mengaktifkan enkripsi bergerak: pastikan klien yang Anda gunakan dikonfigurasi dengan benar

Saat klaster berada dalam mode TLS pilihan, aplikasi Anda harus membuka koneksi TLS ke klaster dan hanya menggunakan koneksi tersebut. Dengan begitu, aplikasi Anda tidak akan mengalami waktu henti saat sedang mengaktifkan enkripsi bergerak. Anda dapat memastikan bahwa tidak ada koneksi TCP yang lebih jelas ke mesin Redis menggunakan perintah info Redis di bagian SSL.

```
# SSL
ssl_enabled:yes
ssl_current_certificate_not_before_date:Mar 20 23:27:07 2017 GMT
ssl_current_certificate_not_after_date:Feb 24 23:27:07 2117 GMT
ssl_current_certificate_serial:D8C7DEA91E684163
tls_mode_connected_tcp_clients:0   (should be zero)
tls_mode_connected_tls_clients:100
```

## Enkripsi Diam di ElastiCache

Untuk membantu menjaga keamanan data Anda, Amazon ElastiCache dan Amazon S3 menyediakan beberapa cara untuk membatasi akses ke data di dalam cache Anda. Untuk informasi selengkapnya, lihat Amazon VPC dan keamanan ElastiCache dan Manajemen Identitas dan Akses untuk Amazon ElastiCache.

Enkripsi diam ElastiCache adalah fitur untuk meningkatkan keamanan data dengan mengenkripsi data di disk. Fitur ini selalu diaktifkan di cache nirserver. Saat diaktifkan, fitur ini mengenkripsi aspek-aspek berikut:

- Disk selama operasi sinkronisasi, pencadangan, dan swap
- Cadangan yang disimpan di Amazon S3

Data yang disimpan di SSD (solid-state drive) dalam klaster yang mengaktifkan tingkatan data selalu dienkripsi.

ElastiCache menawarkan enkripsi diam secara default (dikelola layanan), dan juga kemampuan untuk menggunakan kunci AWS KMS Anda sendiri yang simetris dan dikelola pelanggan di AWS Key Management Service (KMS). Saat cache dicadangkan, di bagian opsi enkripsi, pilih apakah akan menggunakan kunci enkripsi default atau kunci yang dikelola pelanggan. Untuk informasi selengkapnya, lihat Mengaktifkan Enkripsi Diam.

> **Note**
>
> Enkripsi default (dikelola layanan) adalah satu-satunya pilihan yang tersedia di Wilayah GovCloud (US).

> **Important**
>
> Mengaktifkan Enkripsi Diam di klaster Redis yang dirancang sendiri yang ada melibatkan penghapusan grup replikasi Anda yang ada, setelah menjalankan pencadangan dan pemulihan pada grup replikasi.

Enkripsi diam dapat diaktifkan di cache hanya pada saat pembuatannya. Karena diperlukan beberapa pemrosesan untuk mengenkripsi dan mendekripsi data, mengaktifkan enkripsi diam dapat

berdampak pada performa selama operasi ini. Anda harus membandingkan data Anda menggunakan dan tidak menggunakan enkripsi diam untuk menentukan dampaknya terhadap performa untuk kasus penggunaan Anda.

Topik



## Kondisi Enkripsi Diam

Batasan berikut pada enkripsi diam ElastiCache harus diperhatikan saat Anda merencanakan penerapan enkripsi diam ElastiCache:

- Enkripsi diam didukung pada grup replikasi yang menjalankan versi Redis (3.2.6 dijadwalkan untuk EOL, lihat jadwal akhir masa pakai versi Redis), 4.0.10 atau yang lebih baru.
- Enkripsi diam didukung hanya untuk grup replikasi yang berjalan di Amazon VPC.
- Enkripsi diam hanya didukung untuk grup replikasi yang menjalankan jenis simpul berikut.
  - R6gd, R6g, R5, R4, R3
  - M6g, M5, M4, M3
  - T4g, T3, T2

  Untuk informasi selengkapnya, lihat Tipe simpul yang didukung
- Enkripsi diam diaktifkan dengan menetapkan parameter `AtRestEncryptionEnabled` ke `true` secara eksplisit.
- Anda dapat mengaktifkan enkripsi diam pada grup replikasi hanya saat membuat grup replikasi. Anda tidak dapat mengaktifkan dan menonaktifkan enkripsi diam dengan mengubah grup replikasi. Untuk informasi tentang cara menerapkan enkripsi diam pada grup replikasi yang telah ada, lihat Mengaktifkan Enkripsi Diam.
- Jika klaster menggunakan jenis simpul dari keluarga r6gd, data yang disimpan di SSD dienkripsi baik apakah enkripsi diam diaktifkan atau tidak.
- Opsi untuk menggunakan kunci yang dikelola pelanggan untuk enkripsi diam tidak tersedia di Wilayah AWS GovCloud (us-gov-east-1 dan us-gov-west-1).

- Jika sebuah klaster menggunakan jenis simpul dari keluarga r6gd, data yang disimpan di SSD dienkripsi dengan kunci AWS KMS yang dikelola pelanggan yang dipilih (atau enkripsi yang dikelola layanan di Wilayah AWS GovCloud).

Menerapkan enkripsi diam dapat menurunkan performa selama operasi pencadangan dan sinkronisasi simpul. Bandingkan enkripsi diam dengan tanpa enkripsi pada data Anda sendiri untuk menentukan dampaknya terhadap performa untuk implementasi Anda.

Menggunakan kunci yang dikelola pelanggan dari AWS KMS

ElastiCache mendukung kunci AWS KMS (kunci KMS) yang simetris dan dikelola pelanggan untuk enkripsi diam. Kunci KMS yang dikelola pelanggan adalah kunci enkripsi yang Anda buat, miliki, dan kelola di akun AWS Anda. Untuk informasi selengkapnya, lihat Kunci AWS KMS di Panduan Developer untuk AWS Key Management Service. Kunci harus dibuat di AWS KMS sebelum dapat digunakan dengan ElastiCache.

Untuk mempelajari cara membuat kunci root AWS KMS, lihat Buat Kunci di Panduan Developer AWS Key Management Service.

ElastiCache memungkinkan Anda berintegrasi dengan AWS KMS. Untuk informasi selengkapnya, lihat Menggunakan Grant di Panduan Developer AWS Key Management Service. Tidak diperlukan tindakan pelanggan untuk mengaktifkan integrasi Amazon ElastiCache dengan AWS KMS.

Kunci kondisi `kms:ViaService` membatasi penggunaan dari kunci AWS KMS (kunci KMS) untuk meminta dari layanan AWS yang ditentukan. Untuk menggunakan `kms:ViaService` dengan ElastiCache, masukkan kedua nama ViaService ke dalam nilai kunci kondisi: `elasticache.AWS_region.amazonaws.com` dan `dax.AWS_region.amazonaws.com`. Untuk informasi selengkapnya, lihat kms:ViaService.

Anda dapat menggunakan AWS CloudTrail untuk melacak permintaan yang dikirim Amazon ElastiCache ke AWS Key Management Service atas nama Anda. Semua panggilan API ke AWS Key Management Service yang terkait dengan kunci yang dikelola pelanggan memiliki log CloudTrail yang sesuai. Anda juga dapat melihat grant yang dibuat oleh ElastiCache dengan memanggil API KMS ListGrants.

Setelah grup replikasi dienkripsi menggunakan kunci yang dikelola pelanggan, semua cadangan untuk grup replikasi akan dienkripsi sebagai berikut:

- Cadangan harian otomatis dienkripsi menggunakan kunci yang dikelola pelanggan yang terkait dengan klaster.

- Cadangan akhir yang dibuat saat grup replikasi dihapus, juga dienkripsi menggunakan kunci yang dikelola pelanggan yang terkait dengan grup replikasi.
- Cadangan yang dibuat secara manual dienkripsi secara default untuk menggunakan kunci KMS yang terkait dengan grup replikasi. Anda dapat mengabaikan ini dengan memilih kunci dikelola pelanggan yang lain.
- Menyalin cadangan akan secara default menerapkan penggunaan kunci yang dikelola pelanggan yang terkait dengan cadangan sumber. Anda dapat mengabaikan ini dengan memilih kunci dikelola pelanggan yang lain.

> **Note**
>
> - Kunci yang dikelola pelanggan tidak dapat digunakan saat mengekspor cadangan ke bucket Amazon S3 pilihan Anda. Namun, semua cadangan yang diekspor ke Amazon S3 akan dienkripsi menggunakan Enkripsi sisi server. Anda dapat memilih untuk menyalin file cadangan ke objek S3 baru dan mengenkripsi menggunakan kunci KMS yang dikelola pelanggan, menyalin file ke bucket S3 lain yang diatur dengan enkripsi default menggunakan kunci KMS atau mengubah opsi enkripsi di dalam file itu sendiri.
> - Anda juga dapat menggunakan kunci yang dikelola pelanggan untuk mengenkripsi cadangan yang dibuat secara manual untuk grup replikasi yang tidak menggunakan kunci dikelola pelanggan untuk enkripsi. Dengan opsi ini, file cadangan yang disimpan di Amazon S3 akan dienkripsi menggunakan kunci KMS, meskipun data tersebut tidak dienkripsi pada grup replikasi yang asli.
>
> Memulihkan dari cadangan memungkinkan Anda memilih opsi enkripsi yang tersedia, mirip dengan pilihan enkripsi yang tersedia saat membuat grup replikasi baru.

- Jika Anda menghapus kunci atau menonaktifkan kunci dan mencabut grant untuk kunci yang digunakan untuk mengenkripsi cache, cache menjadi tidak dapat dipulihkan. Dengan kata lain, kunci tidak dapat diubah atau dipulihkan setelah kegagalan perangkat keras. AWS KMS menghapus kunci root hanya setelah masa tunggu setidaknya tujuh hari. Setelah kunci dihapus, Anda dapat menggunakan kunci dikelola pelanggan yang berbeda untuk membuat cadangan untuk tujuan pengarsipan.
- Rotasi kunci otomatis mempertahankan properti kunci root AWS KMS Anda, sehingga rotasi tidak berpengaruh pada kemampuan Anda untuk mengakses data ElastiCache. Cache Amazon

ElastiCache terenkripsi tidak mendukung rotasi kunci manual, yang melibatkan pembuatan kunci root baru dan pembaruan referensi apa pun ke kunci lama. Untuk mempelajari lebih lanjut, lihat Merotasikan kunci AWS KMS di Panduan Developer AWS Key Management Service.

- Mengenkripsi cache ElastiCache menggunakan kunci KMS memerlukan satu grant per cache. Grant ini digunakan sepanjang masa pakai cache. Selain itu, satu grant per cadangan digunakan selama pembuatan cadagan. Grant ini dipensiunkan setelah backup dibuat.
- Untuk informasi selengkapnya tentang grant dan batas AWS KMS, lihat bagian Batas di Panduan Developer AWS Key Management Service.

### Mengaktifkan Enkripsi Diam

Semua cache nirserver mengaktifkan enkripsi diam.

Saat membuat klaster yang dirancang sendiri, Anda dapat mengaktifkan enkripsi diam dengan menyetel parameter `AtRestEncryptionEnabled` ke `true`. Anda tidak dapat mengaktifkan enkripsi diam di grup replikasi yang ada.

Anda dapat mengaktifkan enkripsi diam saat Anda membuat cache ElastiCache. Anda dapat melakukannya menggunakan AWS Management Console, AWS CLI, atau API ElastiCache.

Saat membuat cache, Anda dapat memilih salah satu opsi berikut:

- Default – Opsi ini menggunakan enkripsi diam yang dikelola layanan.
- Kunci yang dikelola pelanggan – Opsi ini memungkinkan Anda menyediakan Kunci ID/ARN dari AWS KMS untuk enkripsi diam.

Untuk mempelajari cara membuat kunci root AWS KMS, lihat Membuat Kunci di Panduan Developer AWS Key Management Service

Daftar Isi



### Mengaktifkan Enkripsi Diam di Klaster Redis yang Ada

Anda hanya dapat mengaktifkan enkripsi diam saat Anda membuat grup replikasi Redis. Jika Anda memiliki grup replikasi yang ada tempat Anda ingin mengaktifkan enkripsi diam, lakukan hal berikut.

Untuk mengaktifkan enkripsi diam pada grup replikasi yang ada

1. Membuat cadangan manual dari grup replikasi yang ada. Untuk informasi selengkapnya, lihat Mengambil cadangan manual.
2. Membuat grup replikasi baru dengan memulihkan dari cadangan. Pada grup replikasi baru, aktifkan enkripsi diam. Untuk informasi selengkapnya, lihat Melakukan pemulihan dari cadangan ke dalam cache baru.
3. Memperbarui titik akhir dalam aplikasi Anda untuk mengarah ke grup replikasi baru.
4. Hapus grup replikasi lama. Untuk informasi lebih lanjut, lihat Menghapus klaster atau Menghapus grup replikasi.

### Mengaktifkan Enkripsi Diam Menggunakan AWS Management Console

#### Mengaktifkan Enkripsi Diam di Klaster Nirserver (Konsol)

Semua cache nirserver mengaktifkan enkripsi diam. Secara default, kunci KMS yang dimiliki AWS digunakan untuk mengenkripsi data. Untuk memilih kunci AWS KMS Anda sendiri, buat pilihan berikut:

- Perluas bagian Pengaturan default.
- Pilih Sesuaikan pengaturan default di bagian Pengaturan default.
- Pilih Sesuaikan pengaturan keamanan Anda di bagian Keamanan.
- Pilih CMK terkelola pelanggan di bagian Pengaturan kunci enkripsi.
- Pilih kunci di bagian pengaturan kunci AWS KMS.

#### Mengaktifkan Enkripsi Diam di Klaster yang Dirancang Sendiri (Konsol)

Saat mendesain cache Anda sendiri, konfigurasi 'Dev/Test' dan 'Production' dengan metode 'Easy create' mengaktifkan enkripsi diam menggunakan kunci Default. Saat memilih konfigurasi sendiri, buat pilihan berikut:

- Pilih versi 3.2.6, 4.0.10 atau yang lebih baru sebagai versi mesin Anda.
- Klik kotak centang di sebelah Aktifkan untuk opsi Enkripsi diam.
- Pilih salah satu kunci Default atau CMK yang dikelola Pelanggan.

Untuk prosedur langkah demi langkah, lihat berikut ini:

- Membuat klaster Redis (Mode Klaster Dinonaktifkan) (Konsol)
- Membuat klaster Redis (mode klaster diaktifkan) (Konsol)

### Mengaktifkan Enkripsi Diam Menggunakan AWS CLI

Untuk mengaktifkan enkripsi diam saat membuat klaster Redis menggunakan AWS CLI, gunakan parameter --at-rest-encryption-enabled saat membuat grup replikasi.

#### Mengaktifkan Enkripsi Diam di Klaster Redis (Mode Klaster Dinonaktifkan) (CLI)

Operasi berikut membuat grup replikasi Redis (mode klaster dinonaktifkan) `my-classic-rg` dengan tiga simpul (--num-cache-clusters), satu primer dan dua replika baca. Enkripsi diam diaktifkan untuk grup replikasi ini (—at-rest-encryption-enabled).

Parameter berikut dan nilainya diperlukan untuk mengaktifkan enkripsi pada grup replikasi ini:

Parameter Kunci

- **`--engine`**—Harus berupa `redis`.
- **`--engine-version`**—Harus 3.2.6, 4.0.10 atau yang lebih baru.
- **`--at-rest-encryption-enabled`**—Diperlukan untuk mengaktifkan enkripsi diam.

Example 1: Klaster Redis (Mode Klaster Dinonaktifkan) dengan Replika

Untuk Linux, macOS, atau Unix:

```
aws elasticache create-replication-group \
    --replication-group-id my-classic-rg \
    --replication-group-description "3 node replication group" \
    --cache-node-type cache.m4.large \
    --engine redis \
    --at-rest-encryption-enabled \
    --num-cache-clusters 3
```

Untuk Windows:

```
aws elasticache create-replication-group ^
    --replication-group-id my-classic-rg ^
    --replication-group-description "3 node replication group" ^
```

```
    --cache-node-type cache.m4.large ^
    --engine redis ^
    --at-rest-encryption-enabled ^
    --num-cache-clusters 3 ^
```

Untuk informasi tambahan, lihat hal berikut:

- Membuat grup replikasi Redis (Mode Klaster Dinonaktifkan) dari awal (AWS CLI)
- create-replication-group

Mengaktifkan Enkripsi Diam di Klaster untuk Redis (Mode Klaster Diaktifkan) (CLI)

Operasi berikut membuat grup replikasi Redis (mode klaster diaktifkan) `my-clustered-rg` dengan tiga grup simpul atau serpihan (--num-node-groups). Masing-masing memiliki tiga simpul, satu primer dan dua replika baca (—replika-per-simpul-grup). Enkripsi diam diaktifkan untuk grup replikasi ini (—at-rest-encryption-enabled).

Parameter berikut dan nilainya diperlukan untuk mengaktifkan enkripsi pada grup replikasi ini:

Parameter Kunci

- **`--engine`**—Harus berupa `redis`.
- **`--engine-version`**—Harus 4.0.10 atau yang lebih baru.
- **`--at-rest-encryption-enabled`**—Diperlukan untuk mengaktifkan enkripsi diam.
- **`--cache-parameter-group`**—Harus `default-redis4.0.cluster.on` atau yang berasal dari itu untuk membuat grup replikasi dengan pengaktifan mode klaster.

Example 2: Klaster untuk Redis (Mode Klaster Diaktifkan)

Untuk Linux, macOS, atau Unix:

```
aws elasticache create-replication-group \
   --replication-group-id my-clustered-rg \
   --replication-group-description "redis clustered cluster" \
   --cache-node-type cache.m3.large \
   --num-node-groups 3 \
```

```
    --replicas-per-node-group 2 \
    --engine redis \
    --engine-version 6.2 \
    --at-rest-encryption-enabled \
    --cache-parameter-group default.redis6.x.cluster.on
```

Untuk Windows:

```
aws elasticache create-replication-group ^
    --replication-group-id my-clustered-rg ^
    --replication-group-description "redis clustered cluster" ^
    --cache-node-type cache.m3.large ^
    --num-node-groups 3 ^
    --replicas-per-node-group 2 ^
    --engine redis ^
    --engine-version 6.2 ^
    --at-rest-encryption-enabled ^
    --cache-parameter-group default.redis6.x.cluster.on
```

Untuk informasi tambahan, lihat hal berikut:

- Membuat grup replikasi di Redis (Mode Klaster Diaktifkan) dari awal (AWS CLI)
- create-replication-group

Lihat Juga

- Amazon VPC dan keamanan ElastiCache
- Manajemen Identitas dan Akses untuk Amazon ElastiCache

## Autentikasi dan Otorisasi

ElastiCache mendukung autentikasi pengguna menggunakan IAM dan perintah Redis AUTH, dan otorisasi operasi pengguna menggunakan Role-Based Access Control (RBAC).

Topik

- Kontrol Akses Berbasis Peran (RBAC)
- Autentikasi dengan perintah AUTH Redis
- Menonaktifkan kontrol akses pada cache Redis ElastiCache

## Kontrol Akses Berbasis Peran (RBAC)

Alih-alih melakukan autentikasi pengguna dengan perintah AUTH Redis seperti yang dijelaskan di Autentikasi dengan perintah AUTH Redis, di Redis 6.0 dan yang lebih baru, Anda dapat menggunakan fitur yang disebut Kontrol Akses Berbasis Peran (RBAC). RBAC juga merupakan satu-satunya cara untuk mengontrol akses ke cache nirserver.

Tidak seperti AUTH Redis yang memungkinkan semua klien terautentikasi memiliki akses cache penuh jika tokennya terautentikasi, RBAC memungkinkan Anda mengontrol akses cache melalui grup pengguna. Grup pengguna ini dirancang sebagai cara untuk mengatur akses ke cache.

Dengan RBAC, Anda membuat pengguna dan memberi mereka izin tertentu menggunakan string akses, seperti yang dijelaskan berikut. Anda menempatkan pengguna ke grup pengguna yang diselaraskan dengan peran tertentu (administrator, sumber daya manusia), lalu di-deploy ke satu atau beberapa cache ElastiCache for Redis. Dengan melakukan hal ini, Anda dapat menetapkan batas keamanan antara klien menggunakan cache atau cache Redis yang sama dan mencegah klien mengakses data masing-masing.

RBAC dirancang untuk mendukung pengenalan ACL Redis di Redis 6. Saat Anda menggunakan RBAC dengan cache ElastiCache for Redis Anda, ada beberapa batasan:

- Anda tidak dapat menentukan kata sandi dalam string akses. Anda mengatur kata sandi dengan memanggil CreateUser atau ModifyUser.
- Untuk hak pengguna, Anda meneruskan on dan `off` sebagai bagian dari string akses. Jika tidak ditentukan dalam string akses, pengguna ditetapkan ke `off` dan tidak memiliki hak akses ke cache.
- Anda tidak dapat menggunakan perintah terlarang dan yang berganti nama. Jika Anda menentukan perintah terlarang atau yang berganti nama, pengecualian program akan terjadi. Jika Anda ingin menggunakan daftar kontrol akses (ACL) untuk perintah yang berganti nama, tentukan nama asli dari perintah tersebut.
- Anda tidak dapat menggunakan perintah `reset` sebagai bagian dari string akses. Anda menentukan kata sandi dengan parameter API dan ElastiCache for Redis akan mengelola kata sandi. Dengan demikian, Anda tidak dapat menggunakan `reset` karena akan menghapus semua kata sandi untuk pengguna.
- Redis 6 memperkenalkan perintah ACL LIST. Perintah ini mengembalikan daftar pengguna beserta aturan ACL yang berlaku untuk setiap pengguna. ElastiCache for Redis mendukung perintah `ACL LIST`, tetapi tidak menyertakan dukungan untuk hash kata sandi seperti yang dilakukan

Redis. Dengan ElastiCache for Redis, Anda dapat menggunakan operasi describe-users untuk mendapatkan informasi serupa, termasuk aturan yang terkandung dalam string akses. Namun, describe-users tidak mengambil kata sandi pengguna.

Perintah hanya-baca lainnya yang didukung oleh ElastiCache for Redis termasuk ACL WHOAMI, ACL USERS, dan ACL CAT. ElastiCache for Redis tidak mendukung perintah ACL berbasis tulis lainnya.

- Batasan berikut berlaku:

| Sumber daya | Maksimum yang diperbolehkan |
|---|---|
| Pengguna per grup pengguna | =100. |
| Jumlah pengguna | 1000* |
| Jumlah grup pengguna | =100. |

Penggunaan RBAC dengan ElastiCache for Redis dijelaskan secara lebih terperinci berikut ini.

Topik

- Menentukan Izin Menggunakan String Akses
- Menerapkan RBAC ke Cache untuk ElastiCache for Redis
- Bermigrasi dari AUTH Redis ke RBAC
- Bermigrasi dari RBAC ke AUTH Redis
- Memutar kata sandi untuk pengguna secara otomatis
- Autentikasi dengan IAM

### Menentukan Izin Menggunakan String Akses

Untuk menentukan izin pada cache ElastiCache for Redis, Anda membuat string akses dan menetapkannya ke pengguna, menggunakan AWS CLI atau AWS Management Console.

String akses didefinisikan sebagai daftar aturan yang dipisahkan spasi yang diterapkan pada pengguna. String akses menentukan perintah yang dapat dijalankan oleh pengguna dan kunci yang dapat dioperasikan oleh pengguna. Agar dapat menjalankan perintah, pengguna harus memiliki akses ke perintah yang dijalankan dan semua kunci yang diakses oleh perintah tersebut. Aturan

diterapkan dari kiri ke kanan secara kumulatif, dan string yang lebih sederhana dapat digunakan sebagai pengganti yang disediakan jika ada kelebihan dalam string yang disediakan.

Untuk informasi tentang sintaks aturan ACL, lihat ACL.

Pada contoh berikut, string akses mewakili pengguna aktif dengan akses ke semua kunci dan perintah yang tersedia.

```
on ~* +@all
```

Sintaks string akses dirinci sebagai berikut:

- `on` – Pengguna adalah pengguna yang aktif.
- `~*` – Akses diberikan ke semua kunci yang tersedia.
- `+@all` – Akses diberikan ke semua perintah yang tersedia.

Pengaturan yang mendahului adalah yang paling membatasi. Anda dapat mengubah pengaturan ini untuk membuatnya lebih aman.

Pada contoh berikut, string akses mewakili pengguna dengan akses yang dibatasi untuk akses baca pada kunci yang dimulai dengan keyspace “app::”

```
on ~app::* -@all +@read
```

Anda dapat mempersempit izin ini lebih lanjut dengan mencantumkan perintah yang dapat diakses pengguna:

`+`*`command1`* – Akses pengguna ke perintah dibatasi pada *`command1`*.

`+@category` – Akses pengguna dibatasi pada kategori perintah.

Untuk informasi tentang cara menetapkan string akses ke pengguna, lihat Membuat Pengguna dan Grup Pengguna dengan Konsol dan CLI.

Jika Anda memigrasikan beban kerja yang ada ke ElastiCache, Anda dapat mengambil string akses dengan memanggil `ACL LIST`, dengan mengecualikan pengguna dan hash kata sandi.

Untuk Redis versi 6.2 dan yang lebih baru, sintaks string akses berikut juga didukung:

- `&*` – Akses diberikan ke semua saluran yang tersedia.

Untuk Redis versi 7.0 dan yang lebih baru, sintaks string akses berikut juga didukung:

- `|` – Dapat digunakan untuk memblokir subperintah (misalnya “-config|set”).
- `%R~<pattern>` – Menambahkan pola kunci baca yang ditentukan. Sintaks ini berperilaku mirip dengan pola kunci biasa tetapi hanya memberikan izin untuk membaca dari kunci yang cocok dengan pola yang diberikan. Lihat izin utama untuk informasi selengkapnya.
- `%W~<pattern>` – Menambahkan pola kunci tulis yang ditentukan. Sintaks ini berperilaku mirip dengan pola kunci biasa tetapi hanya memberikan izin untuk menulis dari kunci yang cocok dengan pola yang diberikan. Lihat izin utama untuk informasi selengkapnya.
- `%RW~<pattern>` – Alia untuk `~<pattern>`.
- `(<rule list>)` – Membuat pemilih baru untuk mencocokkan aturan. Pemilih dievaluasi setelah izin pengguna, dan dievaluasi sesuai dengan urutan yang ditentukan. Perintah akan diizinkan jika cocok dengan izin pengguna atau pemilih apa pun. Lihat pemilih ACL untuk informasi lebih lanjut.
- `clearselectors` – Menghapus semua pemilih yang melekat pada pengguna.

### Menerapkan RBAC ke Cache untuk ElastiCache for Redis

Untuk menggunakan RBAC ElastiCache for Redis, lakukan beberapa langkah berikut:

1. Buat satu atau beberapa pengguna.
2. Buat grup pengguna dan tambahkan pengguna ke grup tersebut.
3. Tentukan grup pengguna ke cache yang mengaktifkan enkripsi dalam transit.

Langkah ini dijelaskan secara mendetail di bagian berikut.

Topik

- Membuat Pengguna dan Grup Pengguna dengan Konsol dan CLI
- Mengelola Grup Pengguna dengan Konsol dan CLI
- Menetapkan Grup Pengguna ke Cache Nirserver
- Menempatkan Grup Pengguna ke Grup Replikasi

### Membuat Pengguna dan Grup Pengguna dengan Konsol dan CLI

Informasi pengguna untuk pengguna RBAC adalah ID pengguna, nama pengguna, serta secara opsional, kata sandi dan string akses. String akses menyediakan tingkat izin pada kunci dan perintah. ID pengguna bersifat unik untuk pengguna, dan nama pengguna adalah data yang diteruskan ke mesin.

Pastikan bahwa izin pengguna yang Anda berikan sesuai dengan tujuan yang dimaksud oleh grup pengguna. Misalnya, jika Anda membuat grup pengguna bernama `Administrators`, setiap pengguna yang Anda tambahkan ke grup tersebut akan memiliki string akses yang ditetapkan menjadi akses penuh ke kunci dan perintah. Untuk pengguna di grup pengguna `e-commerce`, Anda dapat mengatur string aksesnya menjadi akses hanya-baca.

ElastiCache secara otomatis membuat konfigurasi pengguna default dengan ID pengguna dan nama pengguna `"default"` serta menambahkannya ke semua grup pengguna. Anda tidak dapat mengubah atau menghapus pengguna ini. Pengguna ini dimaksudkan untuk kompatibilitas dengan perilaku default dari versi Redis sebelumnya dan memiliki string akses yang mengizinkannya untuk memanggil semua perintah dan mengakses semua kunci.

Untuk menambahkan kontrol akses yang tepat ke cache, ganti pengguna default ini dengan pengguna baru yang tidak diaktifkan atau menggunakan kata sandi yang kuat. Untuk mengubah pengguna default, buat pengguna baru dengan nama pengguna yang ditetapkan ke `default`. Anda kemudian dapat menukarkannya dengan pengguna default yang asli.

Prosedur berikut menunjukkan cara untuk menukar pengguna `default` asli dengan pengguna `default` lain yang memiliki string akses yang sudah diubah.

Untuk mengubah pengguna default di konsol

1. Masuk ke AWS Management Console dan buka konsol Amazon ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Pilih Manajemen grup pengguna dari panel navigasi.
3. Untuk ID grup pengguna, pilih ID yang ingin Anda ubah. Pastikan bahwa Anda memilih tautan, bukan kotak centang.
4. Pilih Ubah.
5. Di jendela Ubah, pilih Kelola dan plih pengguna yang Anda inginkan sebagai pengguna default dengan Nama pengguna sebagai default.
6. Pilih Tutup.
7. Pilih Ubah. Saat Anda melakukannya, koneksi apa pun yang ada ke cache yang dimiliki oleh pengguna default asli akan dihentikan.

Untuk mengubah pengguna default dengan AWS CLI

1. Buat pengguna baru dengan nama pengguna `default` menggunakan perintah berikut.

Untuk Linux, macOS, atau Unix:

```
aws elasticache create-user \
 --user-id "new-default-user" \
 --user-name "default" \
 --engine "REDIS" \
 --passwords "a-str0ng-pa))word" \
 --access-string "off +get ~keys*"
```

Untuk Windows:

```
aws elasticache create-user ^
 --user-id "new-default-user" ^
 --user-name "default" ^
 --engine "REDIS" ^
 --passwords "a-str0ng-pa))word" ^
 --access-string "off +get ~keys*"
```

2. Buat grup pengguna dan tambahkan pengguna yang telah Anda buat sebelumnya.

   Untuk Linux, macOS, atau Unix:

   ```
   aws elasticache create-user-group \
     --user-group-id "new-group-2" \
     --engine "REDIS" \
     --user-ids "new-default-user"
   ```

   Untuk Windows:

   ```
   aws elasticache create-user-group ^
     --user-group-id "new-group-2" ^
     --engine "REDIS" ^
     --user-ids "new-default-user"
   ```

3. Tukarkan pengguna `default` yang baru dengan pengguna `default` yang asli.

   Untuk Linux, macOS, atau Unix:

   ```
   aws elasticache modify-user-group \
       --user-group-id test-group \
       --user-ids-to-add "new-default-user" \
   ```

```
    --user-ids-to-remove "default"
```

Untuk Windows:

```
aws elasticache modify-user-group ^
    --user-group-id test-group ^
    --user-ids-to-add "new-default-user" ^
    --user-ids-to-remove "default"
```

Saat operasi perubahan ini dipanggil, koneksi apa pun yang ada ke cache yang dimiliki oleh pengguna default asli akan dihentikan.

Saat membuat pengguna, Anda dapat menetapkan hingga dua kata sandi. Saat Anda mengubah kata sandi, koneksi apa pun yang ada ke cache akan dipertahankan.

Secara khusus, perhatikan batasan kata sandi pengguna ini saat menggunakan RBAC untuk ElastiCache for Redis:

- Kata sandi harus berupa 16–128 karakter yang dapat dicetak.
- Karakter nonalfanumerik berikut tidak diperbolehkan: `, "" / @`.

Mengelola Pengguna dengan Konsol dan CLI

Gunakan prosedur berikut untuk mengelola pengguna di konsol.

Mengelola pengguna di konsol

1. Masuk ke AWS Management Console dan buka konsol Amazon ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Di dasbor Amazon ElastiCache, pilih Manajemen pengguna. Pilihan berikut tersedia:

   - Buat pengguna – Saat membuat pengguna, Anda memasukkan ID pengguna, nama pengguna, mode autentikasi, dan string akses. String akses menetapkan tingkat izin untuk kunci dan perintah apa saja yang diperbolehkan kepada pengguna.

     Saat membuat pengguna, Anda dapat menetapkan hingga dua kata sandi. Saat Anda mengubah kata sandi, koneksi apa pun yang ada ke cache akan dipertahankan.

- Ubah pengguna – Memungkinkan Anda memperbarui pengaturan autentikasi pengguna atau mengubah string aksesnya.
- Hapus pengguna – Akun akan dihapus dari Grup Pengguna yang memiliki akun tersebut.

Gunakan prosedur berikut untuk mengelola pengguna dengan AWS CLI.

Untuk mengubah pengguna menggunakan CLI

- Gunakan perintah `modify-user` untuk memperbarui satu atau beberapa kata sandi pengguna atau mengubah izin akses pengguna.

  Saat pengguna diubah, grup pengguna yang terkait dengan pengguna akan diperbarui, beserta cache apa pun yang terkait dengan grup pengguna. Semua koneksi yang ada akan dipertahankan. Berikut ini adalah beberapa contohnya.

  Untuk Linux, macOS, atau Unix:

```
aws elasticache modify-user \
  --user-id user-id-1 \
  --access-string "~objects:* ~items:* ~public:*" \
  --no-password-required
```

  Untuk Windows:

```
aws elasticache modify-user ^
  --user-id user-id-1 ^
  --access-string "~objects:* ~items:* ~public:*" ^
  --no-password-required
```

> **Note**
>
> Sebaiknya jangan menggunakan opsi `nopass`. Jika Anda menggunakannya, sebaiknya tetapkan izin pengguna menjadi hanya-baca dengan akses ke kumpulan kunci yang terbatas.

Untuk menghapus pengguna dengan menggunakan CLI

- Gunakan perintah `delete-user` untuk menghapus pengguna. Akun dihapus dan dihapus dari grup pengguna yang memiliki akun tersebut. Berikut adalah contohnya.

  Untuk Linux, macOS, atau Unix:

  ```
  aws elasticache delete-user \
    --user-id user-id-2
  ```

  Untuk Windows:

  ```
  aws elasticache delete-user ^
    --user-id user-id-2
  ```

Untuk melihat daftar pengguna, panggil operasi describe-users.

```
aws elasticache describe-users
```

Mengelola Grup Pengguna dengan Konsol dan CLI

Anda dapat membuat grup pengguna untuk mengatur dan mengontrol akses pengguna ke satu atau beberapa cache, seperti yang ditunjukkan berikut.

Gunakan prosedur berikut untuk mengelola grup pengguna menggunakan konsol.

Mengelola grup pengguna menggunakan konsol

1. Masuk ke AWS Management Console dan buka konsol Amazon ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Di dasbor Amazon ElastiCache, pilih Manajemen grup pengguna.

   Operasi berikut tersedia untuk membuat grup pengguna baru:

   - Buat – Saat Anda membuat grup pengguna, Anda menambahkan pengguna, lalu menetapkan grup pengguna ke cache. Misalnya, Anda dapat membuat grup `Admin` pengguna untuk pengguna yang memiliki peran administratif pada cache.

> **Important**
>
> Saat Anda membuat grup pengguna, Anda diminta untuk memasukkan pengguna default.

- Tambahkan Pengguna – Menambahkan pengguna ke grup pengguna.
- Hapus Pengguna – Menghapus pengguna dari grup pengguna. Saat pengguna dihapus dari grup pengguna, koneksi apa pun yang dimiliki grup tersebut ke sebuah cache akan dihentikan.
- Hapus – Gunakan ini untuk menghapus grup pengguna. Perhatikan bahwa grup pengguna itu sendiri, bukan pengguna yang terdapat di dalam grup, yang akan dihapus.

Untuk grup pengguna yang ada, Anda dapat melakukan hal berikut:

- Tambahkan Pengguna – Menambahkan pengguna yang ada ke grup pengguna.
- Hapus Pengguna – Menghapus pengguna yang ada dari grup pengguna.

> **Note**
>
> Pengguna dihapus dari grup pengguna, tetapi tidak dihapus dari sistem.

Gunakan prosedur berikut untuk mengelola grup pengguna menggunakan CLI.

Untuk membuat grup pengguna baru dan menambahkan pengguna dengan menggunakan CLI

- Gunakan perintah `create-user-group` seperti berikut ini.

  Untuk Linux, macOS, atau Unix:

```
aws elasticache create-user-group \
  --user-group-id "new-group-1" \
  --engine "REDIS" \
  --user-ids user-id-1, user-id-2
```

  Untuk Windows:

```
aws elasticache create-user-group ^
  --user-group-id "new-group-1" ^
```

```
  --engine "REDIS" ^
  --user-ids user-id-1, user-id-2
```

Untuk mengubah grup pengguna dengan menambahkan pengguna baru atau menghapus anggota saat ini dengan menggunakan CLI

- Gunakan perintah `modify-user-group` seperti berikut ini.

  Untuk Linux, macOS, atau Unix:

```
aws elasticache modify-user-group --user-group-id new-group-1 \
--user-ids-to-add user-id-3 \
--user-ids-to-remove user-id-2
```

  Untuk Windows:

```
aws elasticache modify-user-group --user-group-id new-group-1 ^
--user-ids-to-add userid-3 ^
--user-ids-to-removere user-id-2
```

> **Note**
>
> Koneksi terbuka milik pengguna yang dihapus dari sebuah grup pengguna akan diakhiri dengan perintah ini.

Untuk menghapus grup pengguna dengan menggunakan CLI

- Gunakan perintah `delete-user-group` seperti berikut ini. Grup pengguna itu sendiri, bukan pengguna yang terdapat di dalam grup, yang akan dihapus.

  Untuk Linux, macOS, atau Unix:

```
aws elasticache delete-user-group /
   --user-group-id
```

  Untuk Windows:

```
aws elasticache delete-user-group ^
   --user-group-id
```

Untuk melihat daftar grup pengguna, Anda dapat memanggil operasi describe-user-groups.

```
aws elasticache describe-user-groups \
  --user-group-id test-group
```

Menetapkan Grup Pengguna ke Cache Nirserver

Setelah Anda membuat grup pengguna dan menambahkan pengguna, langkah terakhir dalam menerapkan RBAC adalah menempatkan grup pengguna ke cache nirserver.

Menetapkan Grup Pengguna ke Cache Nirserver

Untuk menambahkan grup pengguna ke cache nirserver menggunakan file AWS Management Console, lakukan hal berikut:

- Untuk yang menonaktifkan mode klaster, lihat Membuat klaster Redis (Mode Klaster Dinonaktifkan) (Konsol)
- Untuk yang mengaktifkan mode klaster, lihat Membuat klaster Redis (mode klaster diaktifkan) (Konsol)

Menerapkan Grup Pengguna ke Cache Nirserver Menggunakan AWS CLI

Operasi AWS CLI berikut membuat cache nirserver menggunakan parameter user-group-id dengan nilai *my-user-group-id*. Ganti grup subnet `sng-test` dengan grup subnet yang ada.

Parameter Kunci

- **`--engine`** – Harus berupa `redis`.
- **`--user-group-id`** – Nilai ini menyediakan ID grup pengguna, yang terdiri dari pengguna dengan izin akses yang ditentukan untuk cache.

Untuk Linux, macOS, atau Unix:

```
aws elasticache create-serverless-cache \
```

```
    --serverless-cache-name "new-serverless-cache" \
    --description "new-serverless-cache" \
    --engine "redis" \
    --user-group-id "new-group-1"
```

Untuk Windows:

```
aws elasticache create-serverless-cache ^
    --serverless-cache-name "new-serverless-cache" ^
    --description "new-serverless-cache" ^
    --engine "redis" ^
    --user-group-id "new-group-1"
```

Operasi AWS CLI berikut mengubah cache nirserver menggunakan parameter user-group-id dengan nilai *my-user-group-id*.

Untuk Linux, macOS, atau Unix:

```
aws elasticache modify-serverless-cache \
    --serverless-cache-name serverless-cache-1 \
    --user-group-id "new-group-2"
```

Untuk Windows:

```
aws elasticache modify-serverless-cache ^
    --serverless-cache-name serverless-cache-1 ^
    --user-group-id "new-group-2"
```

Perhatikan bahwa perubahan apa pun yang dibuat untuk cache akan diperbarui secara asinkron. Anda dapat memantau progres ini dengan melihat peristiwa. Untuk informasi selengkapnya, lihat Melihat peristiwa ElastiCache.

Menempatkan Grup Pengguna ke Grup Replikasi

Setelah Anda membuat grup pengguna dan menambahkan pengguna, langkah terakhir dalam menerapkan RBAC adalah menetapkan grup pengguna ke grup replikasi.

Menetapkan Grup Pengguna ke Grup Replikasi Menggunakan Konsol

Untuk menambahkan grup pengguna ke sebuah replikasi menggunakan AWS Management Console, lakukan hal berikut:

- Untuk yang menonaktifkan mode klaster, lihat Membuat klaster Redis (Mode Klaster Dinonaktifkan) (Konsol)
- Untuk yang mengaktifkan mode klaster, lihat Membuat klaster Redis (mode klaster diaktifkan) (Konsol)

Menetapkan Grup Pengguna ke Grup Replikasi Menggunakan AWS CLI

Operasi AWS CLI berikut membuat grup replikasi dengan pengaktifan enkripsi dalam transit (TLS) dan parameter user-group-ids dengan nilai *my-user-group-id*. Ganti grup subnet `sng-test` dengan grup subnet yang ada.

Parameter Kunci

- **`--engine`** – Harus berupa `redis`.
- **`--engine-version`** – Harus 6.0 atau yang lebih baru.
- **`--transit-encryption-enabled`** – Diperlukan untuk autentikasi dan untuk menghubungkan grup pengguna.
- **`--user-group-ids`** – Nilai ini menyediakan ID grup pengguna, yang terdiri dari pengguna dengan izin akses yang ditentukan untuk cache.
- **`--cache-subnet-group`** – Diperlukan untuk menghubungkan dengan grup pengguna.

Untuk Linux, macOS, atau Unix:

```
aws elasticache create-replication-group \
    --replication-group-id "new-replication-group" \
    --replication-group-description "new-replication-group" \
    --engine "redis" \
    --cache-node-type cache.m5.large \
    --transit-encryption-enabled \
    --user-group-ids "new-group-1" \
    --cache-subnet-group "cache-subnet-group"
```

Untuk Windows:

```
aws elasticache create-replication-group ^
    --replication-group-id "new-replication-group" ^
    --replication-group-description "new-replication-group" ^
```

```
    --engine "redis" ^
    --cache-node-type cache.m5.large ^
    --transit-encryption-enabled ^
    --user-group-ids "new-group-1" ^
    --cache-subnet-group "cache-subnet-group"
```

Operasi AWS CLI berikut mengubah grup replikasi dengan pengaktifan enkripsi dalam transit (TLS) dan parameter user-group-ids dengan nilai *my-user-group-id*.

Untuk Linux, macOS, atau Unix:

```
aws elasticache modify-replication-group \
    --replication-group-id replication-group-1 \
    --user-group-ids-to-remove "new-group-1" \
    --user-group-ids-to-add "new-group-2"
```

Untuk Windows:

```
aws elasticache modify-replication-group ^
    --replication-group-id replication-group-1 ^
    --user-group-ids-to-remove "new-group-1" ^
    --user-group-ids-to-add "new-group-2"
```

Perhatikan `PendingChanges` dalam respons. Perubahan apa pun yang dibuat untuk cache akan diperbarui secara asinkron. Anda dapat memantau progres ini dengan melihat peristiwa. Untuk informasi selengkapnya, lihat Melihat peristiwa ElastiCache.

Bermigrasi dari AUTH Redis ke RBAC

Jika Anda menggunakan AUTH Redis seperti yang dijelaskan dalam Autentikasi dengan perintah AUTH Redis dan ingin bermigrasi menggunakan RBAC, gunakan prosedur berikut.

Gunakan prosedur berikut untuk bermigrasi dari AUTH Redis ke RBAC menggunakan konsol.

Untuk bermigrasi dari AUTH Redis ke RBAC menggunakan konsol

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Dari daftar di sudut kanan atas, pilih Wilayah AWS tempat cache yang ingin diubah berada.
3. Di panel navigasi, pilih mesin yang berjalan pada cache yang ingin diubah.

Daftar cache mesin yang dipilih akan muncul.

4. Di daftar cache, pilih nama cache yang ingin diubah.
5. Untuk Tindakan, pilih Ubah.

   Jendela Ubah muncul.

6. Untuk Kontrol akses, pilih Daftar kontrol akses grup pengguna.
7. Untuk Daftar Kontrol Akses Grup Pengguna, pilih grup pengguna.
8. Pilih Pratinjau perubahan, lalu di layar berikutnya, pilih Ubah.

Gunakan prosedur berikut untuk bermigrasi dari AUTH Redis ke RBAC menggunakan CLI.

Untuk bermigrasi dari AUTH Redis ke RBAC menggunakan CLI

- Gunakan perintah `modify-replication-group` seperti berikut ini.

  Untuk Linux, macOS, atau Unix:

```
aws elasticache modify-replication-group --replication-group-id test \
    --auth-token-update-strategy DELETE \
    --user-group-ids-to-add user-group-1
```

  Untuk Windows:

```
aws elasticache modify-replication-group --replication-group-id test ^
    --auth-token-update-strategy DELETE ^
    --user-group-ids-to-add user-group-1
```

Bermigrasi dari RBAC ke AUTH Redis

Jika Anda menggunakan RBAC dan ingin bermigrasi ke AUTH Redis, lihat Bermigrasi dari RBAC ke AUTH Redis.

> **Note**
>
> Jika Anda perlu menonaktifkan kontrol akses di cache ElastiCache, Anda harus melakukannya melalui AWS CLI. Untuk informasi selengkapnya, lihat the section called "Menonaktifkan kontrol akses pada cache Redis ElastiCache".

Memutar kata sandi untuk pengguna secara otomatis

Dengan AWS Secrets Manager, Anda dapat mengganti kredensial hard-code dalam kode Anda (termasuk kata sandi) dengan panggilan API ke Secrets Manager untuk mengambil rahasia secara terprogram. Ini membantu memastikan bahwa rahasia tidak dapat dikompromikan oleh seseorang yang memeriksa kode Anda, karena rahasianya tidak ada di sana. Selain itu, Anda dapat mengonfigurasi Secrets Manager untuk memutar rahasia untuk Anda secara otomatis sesuai dengan jadwal yang Anda tentukan. Ini memungkinkan Anda untuk mengganti rahasia jangka panjang dengan rahasia jangka pendek, yang membantu mengurangi risiko kompromi secara signifikan.

Dengan menggunakan Secrets Manager, Anda dapat memutar kata sandi ElastiCache for Redis (yaitu, rahasia) secara otomatis menggunakan fungsi AWS Lambda yang disediakan oleh Secrets Manager.

Untuk informasi selengkapnya tentang AWS Secrets Manager, silakan lihat Apa itu AWS Secrets Manager?

Cara ElastiCache menggunakan rahasia

Dengan Redis 6, ElastiCache for Redis Kontrol Akses Berbasis Peran (RBAC) diperkenalkan untuk mengamankan klaster Redis. Fitur ini memungkinkan koneksi tertentu dibatasi dalam hal perintah yang dapat dijalankan dan kunci yang dapat diakses. Dengan RBAC, saat pelanggan membuat pengguna dengan kata sandi, nilai kata sandi harus dimasukkan secara manual dalam teks biasa dan terlihat oleh operator.

Dengan Secrets Manager, aplikasi mengambil kata sandi dari Secrets Manager bukannya memasukkannya secara manual dan menyimpannya dalam konfigurasi aplikasi. Untuk informasi tentang cara melakukannya, silakan lihat Bagaimana pengguna ElastiCache dikaitkan dengan rahasia.

Ada biaya yang dikeluarkan untuk menggunakan rahasia. Untuk informasi harga, lihat AWSHarga Secrets Manager.

Bagaimana pengguna ElastiCache dikaitkan dengan rahasia

Secrets Manager akan menyimpan referensi untuk pengguna terkait di bidang `SecretString` rahasia. Tidak akan ada referensi ke rahasia dari sisi ElastiCache.

```
{
    "password": "strongpassword",
    "username": "user1",
    "user_arn": "arn:aws:elasticache:us-east-1:xxxxxxxxxx918:user:user1" //this is the
 bond between the secret and the user
}
```

Fungsi rotasi Lambda

Untuk mengaktifkan rotasi kata sandi otomatis Secrets Manager, Anda akan membuat fungsi Lambda yang akan berinteraksi dengan API modify-user untuk memperbarui kata sandi pengguna.

Untuk informasi tentang cara kerjanya, lihat Cara kerja rotasi.

> **Note**
>
> Untuk beberapa layanan AWS, untuk menghindari skenario wakil yang membingungkan, AWS merekomendasikan agar Anda menggunakan baik kunci kondisi global `aws:SourceArn` dan `aws:SourceAccount`. Namun, jika Anda menyertakan kondisi `aws:SourceArn` dalam kebijakan fungsi rotasi Anda, fungsi rotasi hanya dapat digunakan untuk memutar rahasia yang ditentukan oleh ARN tersebut. Kami menyarankan Anda hanya menyertakan kunci konteks `aws:SourceAccount` sehingga Anda dapat menggunakan fungsi rotasi untuk beberapa rahasia.

Untuk masalah apa pun yang mungkin Anda temui, lihat Memecahkan masalah rotasi Secrets Manager AWS.

Cara membuat pengguna ElastiCache dan mengaitkannya dengan Secrets Manager

Langkah-langkah berikut menggambarkan cara membuat pengguna dan mengaitkannya dengan Secrets Manager:

1. Buat pengguna yang tidak aktif

   Untuk Linux, macOS, atau Unix:

```
aws elasticache create-user \
 --user-id user1 \
 --user-name user1 \
 --engine "REDIS" \
 --no-password \ // no authentication is required
 --access-string "*off* +get ~keys*" // this disables the user
```

Untuk Windows:

```
aws elasticache create-user ^
 --user-id user1 ^
 --user-name user1 ^
 --engine "REDIS" ^
 --no-password ^ // no authentication is required
 --access-string "*off* +get ~keys*" // this disables the user
```

Anda akan melihat respons yang mirip dengan berikut ini:

```
{
    "UserId": "user1",
    "UserName": "user1",
    "Status": "active",
    "Engine": "redis",
    "AccessString": "off ~keys* -@all +get",
    "UserGroupIds": [],
    "Authentication": {
        "Type": "no_password"
    },
    "ARN": "arn:aws:elasticache:us-east-1:xxxxxxxxxx918:user:user1"
}
```

2. Buat Rahasia

Untuk Linux, macOS, atau Unix:

```
aws secretsmanager create-secret \
--name production/ec/user1 \
--secret-string \
'{
   "user_arn": "arn:aws:elasticache:us-east-1:123456xxxx:user:user1",
    "username":"user1"
```

```
 }'
```

Untuk Windows:

```
aws secretsmanager create-secret ^
--name production/ec/user1 ^
--secret-string ^
'{
   "user_arn": "arn:aws:elasticache:us-east-1:123456xxxx:user:user1",
    "username":"user1"
 }'
```

Anda akan melihat respons yang mirip dengan berikut ini:

```
{
 "ARN": "arn:aws:secretsmanager:us-east-1:123456xxxx:secret:production/ec/user1-
eaFois",
 "Name": "production/ec/user1",
 "VersionId": "aae5b963-1e6b-4250-91c6-ebd6c47d0d95"
}
```

3. Konfigurasikan fungsi Lambda untuk memutar kata sandi Anda

   a. Masuk ke AWS Management Console dan buka konsol Lambda di https://console.aws.amazon.com/lambda/

   b. Pilih Fungsi pada panel navigasi lalu pilih fungsi yang Anda buat. Pilih nama fungsi, bukan kotak centang di sebelah kirinya.

   c. Pilih tab Konfigurasi.

   d. Dalam konfigurasi Umum, pilih Edit lalu atur Waktu habis setidaknya 12 menit.

   e. Pilih Simpan.

   f. Pilih Variabel lingkungan lalu atur yang berikut:

      i. SECRETS_MANAGER_ENDPOINT - https://secretsmanager.**REGION**.amazonaws.com

      ii. SECRET_ARN — Amazon Resource Name (ARN) dari rahasia yang Anda buat pada Langkah 2.

      iii. USER_NAME - Nama pengguna dari pengguna ElastiCache,

      iv. Pilih Simpan.

   g. Pilih Izin

h. Di bawah Peran eksekusi, pilih nama peran fungsi Lambda untuk dilihat di konsol IAM.

i. Fungsi Lambda akan memerlukan izin berikut untuk memodifikasi pengguna dan mengatur kata sandi:

ElastiCache

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
        "Effect": "Allow",
        "Action": [
            "elasticache:DescribeUsers",
            "elasticache:ModifyUser"
        ],
        "Resource": "arn:aws:elasticache:us-east-1:xxxxxxxxxx918:user:user1"
        }
    ]
}
```

Secrets Manager

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret",
                "secretsmanager:PutSecretValue",
                "secretsmanager:UpdateSecretVersionStage"
            ],
            "Resource": "arn:aws:secretsmanager:us-
east-1:xxxxxxxxxxx:secret:XXXX"
        },
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetRandomPassword",
            "Resource": "*"
        }
    ]
```

```
}
```

4. Mengatur rotasi rahasia Secrets Manager

   a. Dengan menggunakan AWS Management Console, lihat Mengatur rotasi otomatis untuk rahasia Secrets Manager AWS menggunakan konsol

      Untuk informasi selengkapnya tentang mengatur jadwal rotasi, lihat Menjadwalkan ekspresi di rotasi Secrets Manager.

   b. Dengan menggunakan AWS CLI, lihat Mengatur rotasi otomatis untuk AWS Secrets Manager menggunakan AWS Command Line Interface

Autentikasi dengan IAM

Topik

- Gambaran Umum
- Batasan
- Pengaturan
- Terhubung

Gambaran Umum

Dengan Autentikasi IAM, Anda dapat mengautentikasi koneksi ke ElastiCache for Redis menggunakan identitas IAM AWS, ketika cache Anda dikonfigurasi untuk menggunakan Redis versi 7 atau yang lebih baru. Hal ini memungkinkan Anda memperkuat model keamanan Anda dan menyederhanakan banyak tugas keamanan administratif. Anda juga dapat menggunakan Autentikasi IAM untuk mengonfigurasi kontrol akses ketat untuk setiap cache ElastiCache dan pengguna ElastiCache, dengan mengikuti prinsip izin hak akses paling rendah. Autentikasi IAM untuk ElastiCache for Redis bekerja dengan menyediakan token autentikasi IAM berumur pendek alih-alih kata sandi pengguna ElastiCache berumur panjang di `AUTH` Redis atau perintah `HELLO`. Untuk informasi selengkapnya tentang token autentikasi IAM, lihat proses penandatanganan Signature Version 4 di Panduan Referensi AWS Umum dan contoh kode di bawah ini.

Anda dapat menggunakan identitas IAM dan kebijakan terkait untuk lebih membatasi akses Redis. Anda juga dapat memberikan akses ke pengguna dari penyedia Identitas federasi mereka langsung ke cache Redis.

Untuk menggunakan AWS IAM dengan ElastiCache for Redis, Anda harus terlebih dahulu membuat pengguna ElastiCache dengan mode autentikasi diatur ke IAM, kemudian Anda dapat membuat atau menggunakan kembali identitas IAM. Identitas IAM memerlukan kebijakan terkait untuk memberikan tindakan `elasticache:Connect` ke cache ElastiCache dan pengguna ElastiCache. Setelah dikonfigurasi, Anda dapat membuat token autentiksai IAM menggunakan kredensial AWS peran atau pengguna IAM. Terakhir, Anda perlu memberikan token otentikasi IAM berumur pendek sebagai kata sandi di Klien Redis Anda saat menghubungkan ke cache Redis Anda. Klien Redis dengan dukungan untuk penyedia kredensi dapat secara otomatis menghasilkan kredensi sementara secara otomatis untuk setiap koneksi baru. ElastiCache for Redis akan melakukan autentikasi IAM untuk permintaan koneksi pengguna ElastiCache yang mendukung IAM dan akan memvalidasi permintaan koneksi dengan IAM.

Batasan

Saat menggunakan autentikasi IAM, batasan berikut berlaku:

- Autentikasi IAM tersedia saat menggunakan ElastiCache for Redis versi 7.0 atau yang lebih baru.
- Untuk pengguna ElastiCache yang mendukung IAM, properti nama pengguna dan id pengguna harus identik.
- Token autentikasi IAM berlaku selama 15 menit. Untuk koneksi yang berumur panjang, sebaiknya gunakan klien Redis yang mendukung antarmuka penyedia kredensial.
- Koneksi yang diautentikasi IAM ke ElastiCache for Redis akan secara otomatis terputus setelah 12 jam. Koneksi dapat diperpanjang selama 12 jam dengan mengirim perintah `AUTH` atau `HELLO` dengan token autentikasi IAM baru.
- Autentikasi IAM tidak didukung dalam perintah `MULTI EXEC`.
- Saat ini, autentikasi IAM mendukung kunci konteks kondisi global berikut ini:
  - Saat menggunakan autentikasi IAM dengan cache nirserver, `aws:VpcSourceIp`, `aws:SourceVpc`, `aws:SourceVpce`, `aws:CurrentTime`, `aws:EpochTime`, dan `aws:ResourceTag/%s` (dari cache dan pengguna tanpa server terkait) didukung.
  - Saat menggunakan autentikasi IAM dengan grup replikasi, `aws:SourceIp` dan `aws:ResourceTag/%s` (dari grup replikasi terkait dan pengguna) didukung.

  Untuk informasi lebih lanjut tentang kunci konteks kondisi global, lihat kunci konteks kondisi global AWS dalam Panduan Pengguna IAM.

Pengaturan

Untuk mengatur autentikasi IAM:

1. Buat cache

```
aws elasticache create-serverless-cache \
  --serverless-cache-name cache-01  \
  --description "ElastiCache IAM auth application" \
  --engine redis
```

2. Buat dokumen kebijakan kepercayaan IAM, seperti yang ditunjukkan di bawah ini, untuk peran Anda yang memungkinkan akun Anda mengambil peran baru. Simpan kebijakan ini ke file bernama trust-policy.json.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Principal": { "AWS": "arn:aws:iam::123456789012:root" },
        "Action": "sts:AssumeRole"
    }
}
```

3. Buat dokumen kebijakan IAM, seperti yang ditunjukkan di bawah ini. Simpan kebijakan ke file bernama policy.json.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect" : "Allow",
      "Action" : [
        "elasticache:Connect"
      ],
      "Resource" : [
        "arn:aws:elasticache:us-east-1:123456789012:serverlesscache:cache-01",
        "arn:aws:elasticache:us-east-1:123456789012:user:iam-user-01"
      ]
    }
  ]
}
```

4. Buat peran IAM.

```
aws iam create-role \
--role-name "elasticache-iam-auth-app" \
--assume-role-policy-document file://trust-policy.json
```

5. Buat kebijakan IAM.

```
aws iam create-policy \
  --policy-name "elasticache-allow-all" \
  --policy-document file://policy.json
```

6. Lampirkan kebijakan IAM di peran tersebut.

```
aws iam attach-role-policy \
 --role-name "elasticache-iam-auth-app" \
 --policy-arn "arn:aws:iam::123456789012:policy/elasticache-allow-all"
```

7. Buat pengguna baru yang mendukung IAM.

```
aws elasticache create-user \
  --user-name iam-user-01 \
  --user-id iam-user-01 \
  --authentication-mode Type=iam \
  --engine redis \
  --access-string "on ~* +@all"
```

8. Buat grup pengguna dan lampirkan pengguna.

```
aws elasticache create-user-group \
  --user-group-id iam-user-group-01 \
  --engine redis \
  --user-ids default iam-user-01

aws elasticache modify-serverless-cache \
  --serverless-cache-name cache-01  \
  --user-group-id iam-user-group-01
```

Terhubung

Terhubung dengan token sebagai kata sandi

Pertama-tama, Anda harus membuat token autentikasi IAM berumur pendek menggunakan permintaan AWS SigV4 yang telah ditandatangani sebelumnya. Setelah itu, Anda memberikan token autentikasi IAM sebagai kata sandi saat menghubungkan ke cache Redis, seperti yang ditunjukkan pada contoh di bawah ini.

```
String userId = "insert user id";
String cacheName = "insert cache name";
boolean isServerless = true;
String region = "insert region";

// Create a default AWS Credentials provider.
// This will look for AWS credentials defined in environment variables or system
 properties.
AWSCredentialsProvider awsCredentialsProvider = new
 DefaultAWSCredentialsProviderChain();

// Create an IAM authentication token request and signed it using the AWS credentials.
// The pre-signed request URL is used as an IAM authentication token for ElastiCache
 Redis.
IAMAuthTokenRequest iamAuthTokenRequest = new IAMAuthTokenRequest(userId, cacheName,
 region, isServerless);
String iamAuthToken =
 iamAuthTokenRequest.toSignedRequestUri(awsCredentialsProvider.getCredentials());

// Construct Redis URL with IAM Auth credentials provider
RedisURI redisURI = RedisURI.builder()
    .withHost(host)
    .withPort(port)
    .withSsl(ssl)
    .withAuthentication(userId, iamAuthToken)
    .build();

// Create a new Lettuce Redis client
RedisClient client = RedisClient.create(redisURI);
client.connect();
```

Di bawah ini adalah definisi untuk `IAMAuthTokenRequest`.

```
public class IAMAuthTokenRequest {
    private static final HttpMethodName REQUEST_METHOD = HttpMethodName.GET;
    private static final String REQUEST_PROTOCOL = "http://";
    private static final String PARAM_ACTION = "Action";
```

```
    private static final String PARAM_USER = "User";
    private static final String PARAM_RESOURCE_TYPE = "ResourceType";
    private static final String RESOURCE_TYPE_SERVERLESS_CACHE = "ServerlessCache";
    private static final String ACTION_NAME = "connect";
    private static final String SERVICE_NAME = "elasticache";
    private static final long TOKEN_EXPIRY_SECONDS = 900;

    private final String userId;
    private final String cacheName;
    private final String region;
    private final boolean isServerless;

    public IAMAuthTokenRequest(String userId, String cacheName, String region, boolean
isServerless) {
        this.userId = userId;
        this.cacheName = cacheName;
        this.region = region;
        this.isServerless = isServerless;
    }

    public String toSignedRequestUri(AWSCredentials credentials) throws
URISyntaxException {
        Request<Void> request = getSignableRequest();
        sign(request, credentials);
        return new URIBuilder(request.getEndpoint())
            .addParameters(toNamedValuePair(request.getParameters()))
            .build()
            .toString()
            .replace(REQUEST_PROTOCOL, "");
    }

    private <T> Request<T> getSignableRequest() {
        Request<T> request  = new DefaultRequest<>(SERVICE_NAME);
        request.setHttpMethod(REQUEST_METHOD);
        request.setEndpoint(getRequestUri());
        request.addParameters(PARAM_ACTION, Collections.singletonList(ACTION_NAME));
        request.addParameters(PARAM_USER, Collections.singletonList(userId));
        if (isServerless) {
            request.addParameters(PARAM_RESOURCE_TYPE,
Collections.singletonList(RESOURCE_TYPE_SERVERLESS_CACHE));
        }
        return request;
    }
```

```
    private URI getRequestUri() {
        return URI.create(String.format("%s%s/", REQUEST_PROTOCOL, cacheName));
    }

    private <T> void sign(SignableRequest<T> request, AWSCredentials credentials) {
        AWS4Signer signer = new AWS4Signer();
        signer.setRegionName(region);
        signer.setServiceName(SERVICE_NAME);

        DateTime dateTime = DateTime.now();
        dateTime = dateTime.plus(Duration.standardSeconds(TOKEN_EXPIRY_SECONDS));

        signer.presignRequest(request, credentials, dateTime.toDate());
    }

    private static List<NameValuePair> toNamedValuePair(Map<String, List<String>> in) {
        return in.entrySet().stream()
            .map(e -> new BasicNameValuePair(e.getKey(), e.getValue().get(0)))
            .collect(Collectors.toList());
    }
}
```

Terhubung dengan penyedia kredensial

Kode di bawah ini menunjukkan cara mengautentikasi dengan ElastiCache for Redis menggunakan penyedia kredensi autentikasi IAM.

```
String userId = "insert user id";
String cacheName = "insert cache name";
boolean isServerless = true;
String region = "insert region";

// Create a default AWS Credentials provider.
// This will look for AWS credentials defined in environment variables or system
 properties.
AWSCredentialsProvider awsCredentialsProvider = new
 DefaultAWSCredentialsProviderChain();

// Create an IAM authentication token request. Once this request is signed it can be
 used as an
// IAM authentication token for ElastiCache Redis.
IAMAuthTokenRequest iamAuthTokenRequest = new IAMAuthTokenRequest(userId, cacheName,
 region, isServerless);
```

```
// Create a Redis credentials provider using IAM credentials.
RedisCredentialsProvider redisCredentialsProvider = new
 RedisIAMAuthCredentialsProvider(
    userId, iamAuthTokenRequest, awsCredentialsProvider);

// Construct Redis URL with IAM Auth credentials provider
RedisURI redisURI = RedisURI.builder()
    .withHost(host)
    .withPort(port)
    .withSsl(ssl)
    .withAuthentication(redisCredentialsProvider)
    .build();

// Create a new Lettuce Redis client
RedisClient client = RedisClient.create(redisURI);
client.connect();
```

Di bawah ini adalah contoh klien Lettuce Redis yang membungkus IAMAuthTokenRequest dalam penyedia kredensial untuk menghasilkan kredensial sementara secara otomatis jika diperlukan.

```
public class RedisIAMAuthCredentialsProvider implements RedisCredentialsProvider {
    private static final long TOKEN_EXPIRY_SECONDS = 900;

    private final AWSCredentialsProvider awsCredentialsProvider;
    private final String userId;
    private final IAMAuthTokenRequest iamAuthTokenRequest;
    private final Supplier<String> iamAuthTokenSupplier;

    public RedisIAMAuthCredentialsProvider(String userId,
        IAMAuthTokenRequest iamAuthTokenRequest,
        AWSCredentialsProvider awsCredentialsProvider) {
        this.userName = userName;
        this.awsCredentialsProvider = awsCredentialsProvider;
        this.iamAuthTokenRequest = iamAuthTokenRequest;
        this.iamAuthTokenSupplier =
 Suppliers.memoizeWithExpiration(this::getIamAuthToken, TOKEN_EXPIRY_SECONDS,
 TimeUnit.SECONDS);
    }

    @Override
    public Mono<RedisCredentials> resolveCredentials() {
        return Mono.just(RedisCredentials.just(userId, iamAuthTokenSupplier.get()));
```

```
    }

    private String getIamAuthToken() {
        return
 iamAuthTokenRequest.toSignedRequestUri(awsCredentialsProvider.getCredentials());
    }
}
```

Autentikasi dengan perintah AUTH Redis

> **Note**
>
> Redis AUTH telah digantikan oleh the section called “Kontrol Akses Berbasis Peran (RBAC)”. Semua cache nirserver harus menggunakan RBAC untuk autentikasi.

Token, atau kata sandi, autentikasi Redis, memungkinkan Redis mewajibkan kata sandi sebelum mengizinkan klien menjalankan perintah, sehingga meningkatkan keamanan data. Redis AUTH hanya tersedia untuk klaster yang dirancang sendiri.

Topik

- Gambaran Umum AUTH di ElastiCache for Redis
- Menerapkan autentikasi ke klaster ElastiCache for Redis
- Memodifikasi token AUTH pada klaster ElastiCache for Redis yang ada
- Bermigrasi dari RBAC ke AUTH Redis

### Gambaran Umum AUTH di ElastiCache for Redis

Jika Anda menggunakan Redis AUTH dengan klaster ElastiCache for Redis, ada beberapa perbaikan.

Secara khusus perhatikan pembatasan token atau kata sandi AUTH ini saat menggunakan AUTH dengan ElastiCache for Redis:

- Token, atau kata sandi, harus 16–128 karakter yang dapat dicetak.
- Karakter nonalfanumerik yang diperbolehkan adalah (!, &, #, $, ^, <, >, -).
- AUTH hanya dapat diaktifkan untuk klaster ElastiCache for Redis yang mengaktifkan enkripsi in-transit.

Untuk menyiapkan token yang kuat, sebaiknya ikuti kebijakan kata sandi yang ketat, seperti yang mewajibkan hal berikut:

- Token, atau kata sandi, harus menyertakan setidaknya tiga dari beberapa jenis karakter berikut:
  - Karakter huruf besar
  - Karakter huruf kecil
  - Angka
  - Karakter nonalfanumerik (`!`, &, #, $, ^, <, >, `-`)
- Token, atau kata sandi, tidak boleh mengandung kata dari kamus atau kata dari kamus yang dimodifikasi sedikit.
- Token, atau kata sandi, tidak boleh sama dengan atau mirip dengan token yang baru saja digunakan.

Menerapkan autentikasi ke klaster ElastiCache for Redis

Anda dapat mewajibkan pengguna untuk memasukkan token (kata sandi) pada server Redis yang dilindungi token. Untuk melakukannya, sertakan parameter `--auth-token` (API: `AuthToken`) dengan token yang tepat ketika Anda membuat grup replikasi atau klaster Anda. Sertakan juga ke dalamnya semua perintah berikutnya untuk grup replikasi atau klaster.

Operasi AWS CLI berikut membuat grup replikasi dengan pengaktifan enkripsi dalam transit (TLS) dan token AUTH *`This-is-a-sample-token`*. Ganti grup subnet `sng-test` dengan grup subnet yang ada.

Parameter Kunci

- **`--engine`** – Harus berupa `redis`.
- **`--engine-version`** – Harus 3.2.6, 4.0.10, atau yang lebih baru.
- **`--transit-encryption-enabled`** – Diperlukan untuk autentikasi dan kelayakan HIPAA.
- **`--auth-token`** – Diperlukan untuk kelayakan HIPAA. Nilai ini harus berupa token yang benar untuk server Redis yang dilindungi token ini.
- **`--cache-subnet-group`** – Diperlukan untuk kelayakan HIPAA.

Untuk Linux, macOS, atau Unix:

```
aws elasticache create-replication-group \
```

```
    --replication-group-id authtestgroup \
    --replication-group-description authtest \
    --engine redis \
    --cache-node-type cache.m4.large \
    --num-node-groups 1 \
    --replicas-per-node-group 2 \
    --transit-encryption-enabled \
    --auth-token This-is-a-sample-token \
    --cache-subnet-group sng-test
```

Untuk Windows:

```
aws elasticache create-replication-group ^
    --replication-group-id authtestgroup ^
    --replication-group-description authtest ^
    --engine redis ^
    --cache-node-type cache.m4.large ^
    --num-node-groups 1 ^
    --replicas-per-node-group 2 ^
    --transit-encryption-enabled ^
    --auth-token This-is-a-sample-token ^
    --cache-subnet-group sng-test
```

Memodifikasi token AUTH pada klaster ElastiCache for Redis yang ada

Untuk mempermudah memperbarui autentikasi Anda, Anda dapat mengubah token AUTH yang digunakan pada klaster ElastiCache for Redis. Anda dapat melakukan modifikasi ini jika versi mesin 5.0.6 atau lebih tinggi dan jika ElastiCache for Redis mengaktifkan enkripsi in transit.

Memodifikasi token auth mendukung dua strategi: ROTATE dan SET. Strategi ROTATE menambahkan token AUTH tambahan pada server sambil tetap menggunakan token sebelumnya. Strategi SET memperbarui server untuk mendukung hanya satu token AUTH. Buat panggilan modifikasi ini dengan parameter `--apply-immediately` untuk menerapkan perubahan dengan segera.

Merotasi token AUTH

Untuk memperbarui server Redis dengan token AUTH yang baru, panggil API `ModifyReplicationGroup` dengan parameter `--auth-token` sebagai token auth yang baru dan `--auth-token-update-strategy` dengan nilai ROTATE. Setelah modifikasi selesai, klaster akan mendukung token AUTH sebelumnya selain token yang ditentukan dalam parameter `auth-token`.

> **Note**
>
> Jika Anda tidak membuat konfigurasi token AUTH sebelumnya, setelah modifikasi selesai, klaster akan mendukung token no AUTH selain token yang ditentukan dalam parameter auth-token.

Jika modifikasi ini dilakukan pada server yang sudah mendukung dua token AUTH, token AUTH yang paling lama juga akan dihapus selama operasi ini, yang memungkinkan server mendukung hingga dua token AUTH terbaru pada waktu yang ditentukan.

Pada titik ini, Anda dapat melanjutkan pembaruan klien untuk menggunakan token AUTH terbaru. Setelah klien diperbarui, Anda dapat menggunakan strategi SET untuk rotasi token AUTH (dijelaskan pada bagian berikut) untuk secara eksklusif mulai menggunakan token baru.

Operasi AWS CLI berikut memodifikasi grup replikasi untuk merotasi token AUTH *`This-is-the-rotated-token`*.

Untuk Linux, macOS, atau Unix:

```
aws elasticache modify-replication-group \
--replication-group-id authtestgroup \
--auth-token This-is-the-rotated-token \
--auth-token-update-strategy ROTATE \
--apply-immediately
```

Untuk Windows:

```
aws elasticache modify-replication-group ^
--replication-group-id authtestgroup ^
--auth-token This-is-the-rotated-token ^
--auth-token-update-strategy ROTATE ^
--apply-immediately
```

Mengatur token AUTH

Untuk memperbarui server Redis dengan dua token AUTH untuk mendukung satu token AUTH, panggil operasi API `ModifyReplicationGroup`. Panggil `ModifyReplicationGroup` dengan parameter `--auth-token` sebagai token AUTH baru dan parameter `--auth-token-update-`

`strategy` dengan nilai SET. Parameter `auth-token` harus mempunyai nilai yang sama dengan token AUTH yang terakhir dirotasi. Setelah modifikasi selesai, server Redis hanya mendukung token AUTH yang ditentukan dalam parameter `auth-token`.

Operasi AWS CLI berikut memodifikasi grup replikasi untuk menetapkan token AUTH ke *`This-is-the-set-token`*.

Untuk Linux, macOS, atau Unix:

```
aws elasticache modify-replication-group \
--replication-group-id authtestgroup \
--auth-token This-is-the-set-token \
--auth-token-update-strategy SET \
--apply-immediately
```

Untuk Windows:

```
aws elasticache modify-replication-group ^
--replication-group-id authtestgroup ^
--auth-token This-is-the-set-token ^
--auth-token-update-strategy SET ^
--apply-immediately
```

Mengaktifkan autentikasi pada klaster ElastiCache for Redis yang ada

Untuk mengaktifkan autentikasi pada server Redis yang ada, panggil operasi API `ModifyReplicationGroup`. Panggil `ModifyReplicationGroup` dengan parameter `--auth-token` sebagai token baru dan `--auth-token-update-strategy` dengan nilai ROTATE.

Setelah modifikasi selesai, klaster mendukung token AUTH yang ditentukan pada parameter `auth-token` selain mendukung koneksi tanpa autentikasi. Mengaktifkan autentikasi hanya didukung di server Redis dengan pengaktifan enkripsi dalam transit (TLS).

Bermigrasi dari RBAC ke AUTH Redis

Jika Anda melakukan autentikasi pengguna dengan Kontrol Akses Berbasis Peran (RBAC) Redis seperti yang dijelaskan pada Autentikasi pengguna dengan kontrol akses berbasis peran (RBAC) dan ingin bermigrasi ke AUTH Redis, gunakan prosedur berikut. Anda dapat bermigrasi menggunakan konsol atau CLI.

Untuk bermigrasi dari RBAC ke AUTH Redis menggunakan konsol

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Dari daftar di sudut kanan atas, pilih Wilayah AWS sesuai lokasi klaster yang akan diubah.
3. Di panel navigasi, pilih mesin yang berjalan pada klaster yang ingin diubah.

   Daftar klaster mesin yang dipilih akan muncul.
4. Dalam daftar klaster tersebut, pilih nama klaster yang ingin Anda ubah.
5. Untuk Tindakan, pilih Ubah.

   Jendela Ubah muncul.
6. Untuk kontrol Akses, pilih akses pengguna default Redis AUTH.
7. Di bawah token Redis AUTH, tetapkan token baru.
8. Pilih Pratinjau perubahan, lalu di layar berikutnya, pilih Ubah.

Untuk bermigrasi dari RBAC ke AUTH Redis menggunakan AWS CLI

- Gunakan salah satu perintah berikut ini.

  Untuk Linux, macOS, atau Unix:

```
aws elasticache modify-replication-group \
    --replication-group-id test \
    --remove-user-groups \
    --auth-token password \
    --auth-token-update-strategy SET \
    --apply-immediately
```

  Untuk Windows:

```
aws elasticache modify-replication-group ^
    --replication-group-id test ^
    --remove-user-groups ^
    --auth-token password ^
    --auth-token-update-strategy SET ^
    --apply-immediately
```

Untuk informasi selengkapnya tentang bekerja dengan AUTH, lihat Token AUTH di situs web redis.io.

> **Note**
>
> Jika Anda perlu menonaktifkan kontrol akses di Klaster ElastiCache Cluster, lihat the section called “Menonaktifkan kontrol akses pada cache Redis ElastiCache”

Menonaktifkan kontrol akses pada cache Redis ElastiCache

Ikuti petunjuk di bawah ini untuk menonaktifkan kontrol akses pada cache Redis dengan TLS aktif. Cache Redis Anda akan memiliki satu dari dua jenis konfigurasi yang berbeda: akses pengguna default AUTH Redis atau Daftar kontrol akses grup pengguna (RBAC). Jika cache Anda dibuat dengan konfigurasi AUTH, Anda harus mengubahnya ke konfigurasi RBAC sebelum Anda dapat menonaktifkan cache dengan menghapus grup pengguna. Jika cache Anda dibuat dengan konfigurasi RBAC, Anda dapat langsung menonaktifkannya.

Untuk menonaktifkan cache nirserver Redis yang dikonfigurasi dengan RBAC

1. Hapus grup pengguna untuk menonaktifkan kontrol akses.

```
aws elasticache modify-serverless-cache --serverless-cache-name <serverless-cache>
 --remove-user-group
```

2. (Opsional) Pastikan bahwa tidak ada grup pengguna yang terkait dengan cache nirserver.

```
aws elasticache describe-serverless-caches --serverless-cache-name <serverless-
cache>
{
    "..."
    "UserGroupId": ""
    "..."
}
```

Untuk menonaktifkan cache Redis yang dikonfigurasi dengan token AUTH

1. Ubah token AUTH ke RBAC dan tentukan grup pengguna yang akan ditambahkan.

```
aws elasticache modify-replication-group --replication-group-id <replication-group-
id-value> --auth-token-update-strategy DELETE --user-group-ids-to-add <user-group-
value>
```

2. Verifikasi bahwa token AUTH dinonaktifkan dan grup pengguna telah ditambahkan.

```
aws elasticache describe-replication-groups --replication-group-id <replication-
group-id-value>
{
    "..."
    "AuthTokenEnabled": false,
    "UserGroupIds": [
        "<user-group-value>"
    ]
    "..."
}
```

3. Hapus grup pengguna untuk menonaktifkan kontrol akses.

```
aws elasticache modify-replication-group --replication-group-id <replication-group-
value> --user-group-ids-to-remove <user-group-value>
{
    "..."
    "PendingModifiedValues": {
    "UserGroups": {
      "UserGroupIdsToAdd": [],
      "UserGroupIdsToRemove": [
        "<user-group-value>"
      ]
    }
    "..."
}
```

4. (Opsional) Pastikan bahwa tidak ada grup pengguna yang terkait dengan klaster. Bidang `AuthTokenEnabled` juga harus menampilkan salah (false).

```
aws elasticache describe-replication-groups --replication-group-id <replication-
group-value>
"AuthTokenEnabled": false
```

Untuk menonaktifkan klaster Redis yang dikonfigurasi dengan RBAC

1. Hapus grup pengguna untuk menonaktifkan kontrol akses.

```
aws elasticache modify-replication-group --replication-group-id <replication-group-
value> --user-group-ids-to-remove <user-group-value>
{
    "..."
    "PendingModifiedValues": {
    "UserGroups": {
      "UserGroupIdsToAdd": [],
      "UserGroupIdsToRemove": [
        "<user-group-value>"
      ]
    }
    "..."
}
```

2. (Opsional) Pastikan bahwa tidak ada grup pengguna yang terkait dengan klaster. Bidang `AuthTokenEnabled` juga harus menampilkan salah (false).

```
aws elasticache describe-replication-groups --replication-group-id <replication-
group-value>
"AuthTokenEnabled": false
```

# Privasi lalu lintas kerja internet

Amazon ElastiCache menggunakan teknik berikut untuk mengamankan data cache Anda dan melindunginya dari akses yang tidak sah:

- Amazon VPC dan keamanan ElastiCache menjelaskan jenis grup keamanan yang Anda butuhkan untuk instalasi Anda.
- Manajemen Identitas dan Akses untuk Amazon ElastiCache untuk memberikan dan membatasi tindakan pengguna, grup, dan peran.

## Amazon VPC dan keamanan ElastiCache

Karena keamanan data itu penting, ElastiCache menyediakan sarana bagi Anda untuk mengontrol siapa yang memiliki akses ke data Anda. Cara Anda mengontrol akses ke data Anda bergantung

pada iya atau tidaknya Anda meluncurkan klaster di Amazon Virtual Private Cloud (Amazon VPC) atau Amazon EC2-Classic.

> **Important**
>
> Kami tidak lagi menggunakan Amazon EC2-Classic untuk meluncurkan klaster ElastiCache. Semua simpul generasi terkini diluncurkan di Amazon Virtual Private Cloud saja.

Layanan Amazon Virtual Private Cloud (Amazon VPC) mendefinisikan jaringan virtual yang mirip dengan pusat data tradisional. Saat Anda mengonfigurasi Amazon VPC, Anda dapat memilih rentang alamat IP, membuat subnet, serta mengonfigurasi tabel rute, gateway jaringan, dan pengaturan keamanan. Anda juga dapat menambahkan klaster cache ke jaringan virtual, dan mengontrol akses ke klaster cache dengan menggunakan grup keamanan Amazon VPC.

Bagian ini menjelaskan cara mengonfigurasi klaster ElastiCache secara manual di Amazon VPC. Informasi ini ditujukan bagi pengguna yang menginginkan pemahaman yang lebih mendalam tentang cara kerja ElastiCache dan Amazon VPC.

Topik

## Memahami ElastiCache dan Amazon VPC

ElastiCache terintegrasi penuh dengan Amazon Virtual Private Cloud (Amazon VPC). Untuk pengguna ElastiCache, hal ini berarti sebagai berikut:

- Jika akun AWS Anda hanya mendukung platform EC2-VPC, ElastiCache selalu meluncurkan klaster Anda di Amazon VPC.
- Jika Anda baru menggunakan AWS, klaster Anda akan di-deploy ke Amazon VPC. VPC default akan dibuat untuk Anda secara otomatis.
- Jika Anda memiliki VPC default dan tidak menentukan subnet saat meluncurkan sebuah klaster, klaster tersebut akan diluncurkan ke Amazon VPC default Anda.

Untuk informasi selengkapnya, lihat Mendeteksi Platform Anda yang Didukung dan Apakah Anda Memiliki VPC Default.

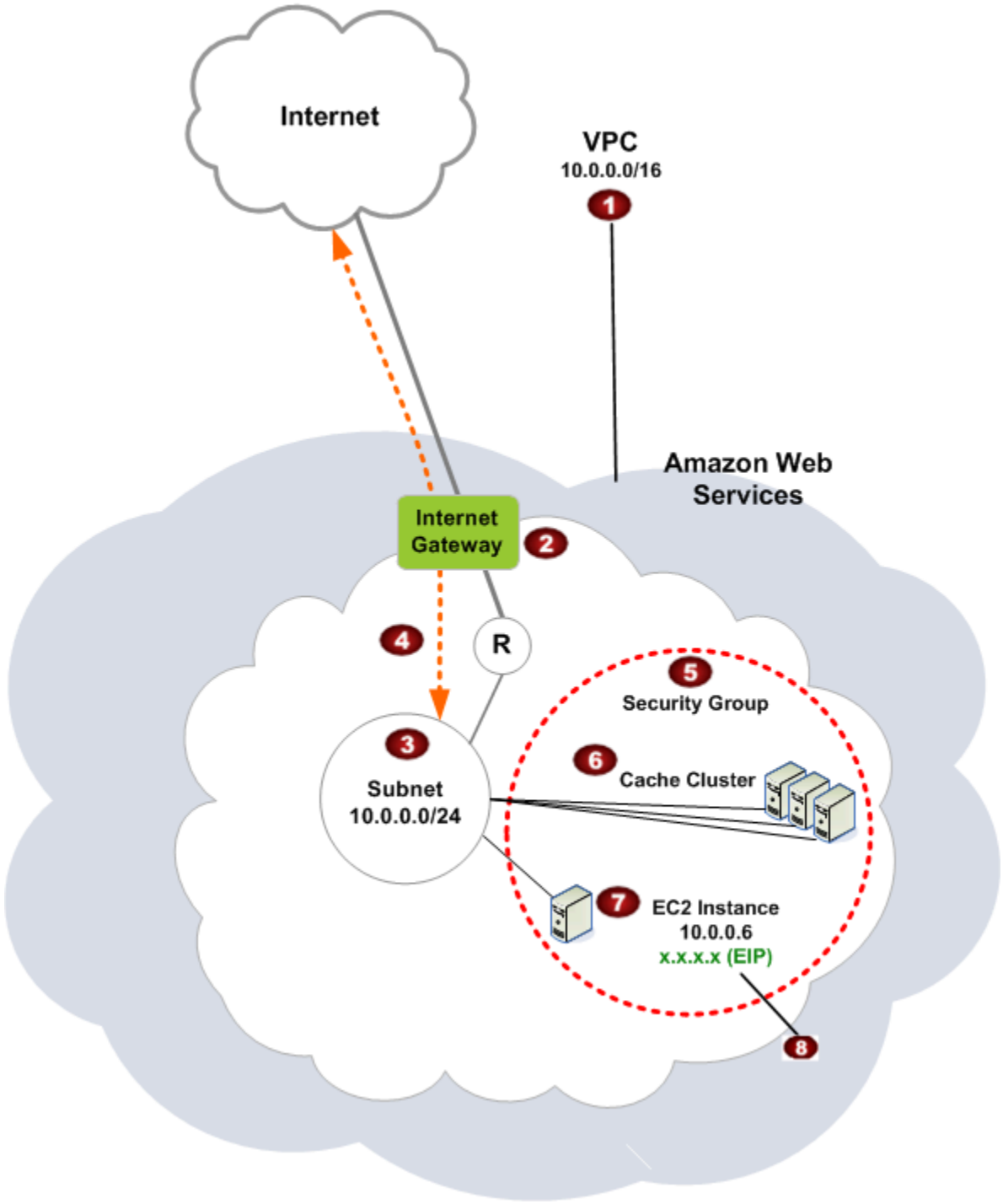
Dengan Amazon Virtual Private Cloud, Anda dapat membuat jaringan virtual di cloud AWS yang sangat menyerupai pusat data tradisional. Anda dapat mengonfigurasi Amazon VPC Anda, termasuk memilih rentang alamat IP, membuat subnet, dan mengonfigurasi tabel rute, gateway jaringan, dan pengaturan keamanan.

Fungsionalitas dasar ElastiCache sama di cloud privat virtual; ElastiCache mengelola peningkatan perangkat lunak, patching, deteksi kegagalan, dan pemulihan baik klaster Anda di-deploy di dalam atau di luar Amazon VPC.

Simpul cache ElastiCache yang di-deploy di luar Amazon VPC ditugaskan alamat IP yang diselesaikan oleh nama titik akhir/DNS. Ini menyediakan konektivitas dari instans Amazon Elastic Compute Cloud (Amazon EC2). Saat Anda meluncurkan klaster ElastiCache ke dalam subnet privat Amazon VPC, setiap simpul cache ditugaskan alamat IP privat dalam subnet tersebut.

Gambaran umum ElastiCache di Amazon VPC

Diagram dan tabel berikut menjelaskan lingkungan Amazon VPC, beserta klaster ElastiCache dan instans Amazon EC2 yang diluncurkan di Amazon VPC.




| | |
|---|---|
| 1 | Amazon VPC adalah bagian terisolasi dari Cloud AWS yang ditugaskan blok alamat IP sendiri. |
| 2 | |

| | |
|---|---|
| | Gateway Internet menghubungkan Amazon VPC Anda secara langsung ke Internet dan menyediakan akses ke sumber daya AWS lainnya seperti Amazon Simple Storage Service (Amazon S3) yang berjalan di luar Amazon VPC Anda. |
| 3 | Subnet Amazon VPC adalah segmen rentang alamat IP dari Amazon VPC tempat Anda dapat mengisolasi sumber daya AWS sesuai dengan kebutuhan keamanan dan operasional Anda. |
| 4 | Tabel rute di Amazon VPC mengarahkan lalu lintas jaringan antara subnet dan Internet. Amazon VPC memiliki router tersirat, yang dilambangkan dalam diagram ini menggunakan lingkaran dengan R. |
| 5 | Grup keamanan Amazon VPC mengendalikan lalu lintas masuk dan keluar untuk klaster ElastiCache dan instans Amazon EC2. |
| 6 | Anda dapat meluncurkan klaster ElastiCache di subnet. Simpul cache memiliki alamat IP privat dari rentang alamat subnet. |
| 7 | Anda juga dapat meluncurkan instans Amazon EC2 dalam subnet. Setiap instans Amazon EC2 memiliki alamat IP privat dari rentang alamat subnet. Instans Amazon EC2 dapat tersambung ke simpul cache apa pun di dalam subnet yang sama. |
| 8 | Agar instans Amazon EC2 di Amazon VPC dapat dijangkau dari Internet, Anda perlu menugaskan alamat publik statis yang disebut alamat IP Elastic ke instans tersebut. |

### Prasyarat

Untuk membuat klaster ElastiCache dalam Amazon VPC, Amazon VPC Anda harus memenuhi persyaratan berikut:

- Amazon VPC harus mengizinkan instans Amazon EC2 tidak terdedikasi. Anda tidak dapat menggunakan ElastiCache di Amazon VPC yang dikonfigurasi untuk penghunian instans khusus.

- Grup subnet cache harus ditentukan untuk Amazon VPC Anda. ElastiCache menggunakan grup subnet cache tersebut untuk memilih subnet dan alamat IP dalam subnet tersebut untuk mengasosiasikannya dengan titik akhir VPC dan simpul cache Anda.
- Blok CIDR untuk setiap subnet harus cukup besar untuk menyediakan alamat IP cadangan untuk ElastiCache untuk digunakan selama aktivitas pemeliharaan.

### Perutean dan keamanan

Anda dapat mengonfigurasi perutean di Amazon VPC untuk mengendalikan tempat arus lalu lintas (misalnya, ke gateway Internet atau gateway privat virtual). Dengan gateway Internet, Amazon VPC memiliki akses langsung ke sumber daya AWS lainnya yang tidak berjalan di Amazon VPC Anda. Jika Anda memilih untuk hanya memiliki gateway privat virtual dengan koneksi ke jaringan lokal dari organisasi Anda, maka Anda dapat merutekan lalu lintas dari dan ke Internet tersebut melalui VPN serta menggunakan kebijakan keamanan lokal dan firewall untuk mengontrol lalu lintas keluar. Dalam kasus ini, Anda dikenakan biaya bandwidth tambahan jika Anda mengakses sumber daya AWS melalui Internet.

Anda dapat menggunakan grup keamanan Amazon VPC untuk membantu mengamankan klaster ElastiCache dan instans Amazon EC2 di Amazon VPC Anda. Grup keamanan bertindak seperti firewall di tingkat instans, bukan di tingkat subnet.

> **Note**
>
> Sangat dianjurkan untuk menggunakan nama DNS untuk terhubung ke simpul cache Anda, karena alamat IP yang mendasarinya dapat berubah.

### Dokumentasi Amazon VPC

Amazon VPC memiliki kumpulan dokumentasinya sendiri untuk menjelaskan cara membuat dan menggunakan Amazon VPC Anda. Tabel berikut memberikan tautan ke panduan Amazon VPC.

| Deskripsi | Dokumentasi |
| --- | --- |
| Cara mulai menggunakan Amazon VPC | Mulai menggunakan Amazon VPC |
| Cara menggunakan Amazon VPC melalui AWS Management Console | Panduan Pengguna Amazon VPC |

| Deskripsi | Dokumentasi |
| --- | --- |
| Deskripsi lengkap dari semua perintah Amazon VPC | Referensi Baris Perintah Amazon EC2 (perintah Amazon VPC ditemukan di dalam referensi Amazon EC2) |
| Deskripsi lengkap dari operasi API Amazon VPC, tipe data, dan kesalahan | Referensi API Amazon EC2 (operasi API Amazon VPC ditemukan dalam referensi Amazon EC2) |
| Informasi untuk administrator jaringan yang perlu membuat konfigurasi gateway di ujung koneksi VPN IPsec opsional Anda | Apa itu Site-to-Site VPN AWS? |

Untuk informasi lebih detail tentang Amazon Virtual Private Cloud, lihat Amazon Virtual Private Cloud.

# Pola Akses untuk Mengakses Cache ElastiCache di Amazon VPC

Amazon ElastiCache mendukung skenario berikut untuk mengakses cache di Amazon VPC:

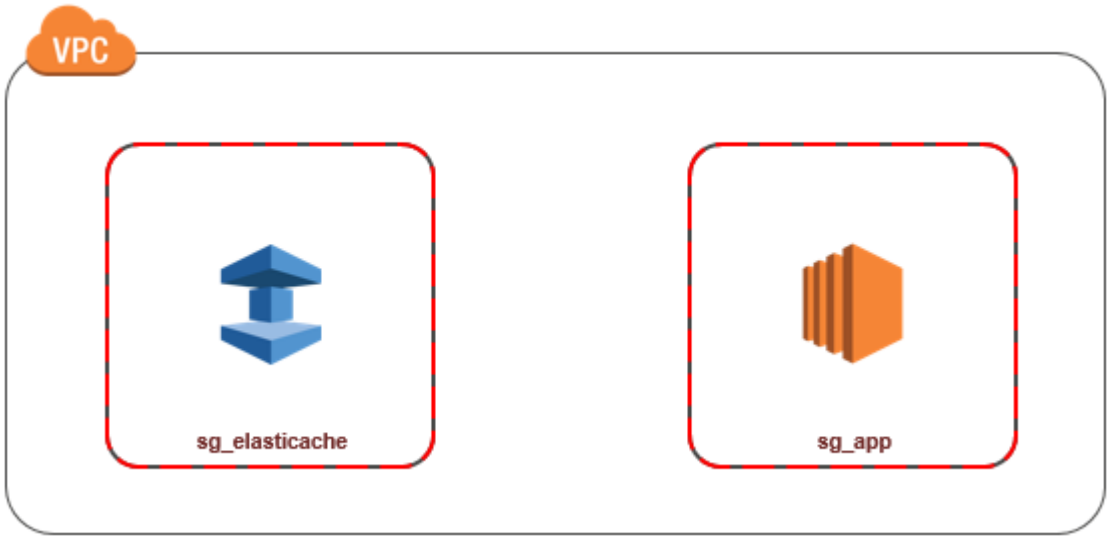
Daftar Isi



## Mengakses Cache ElastiCache saat berada di Amazon VPC yang sama dengan Instans Amazon EC2

Kasus penggunaan yang paling umum adalah saat aplikasi yang disebarkan pada instans EC2 perlu terhubung ke klaster di VPC yang sama.

Diagram berikut menggambarkan skenario ini




Cara paling sederhana untuk mengelola akses antara instans EC2 dan cache di VPC yang sama adalah dengan melakukan tindakan berikut:

1. Buat grup keamanan VPC untuk cache Anda. Grup keamanan ini dapat digunakan untuk membatasi akses ke cache. Contohnya, Anda dapat membuat aturan khusus untuk grup keamanan ini yang mengizinkan akses TCP menggunakan port yang Anda tetapkan untuk cache saat Anda membuatnya dan alamat IP yang Anda gunakan untuk mengakses cache tersebut.

   Port default untuk Redis adalah 6379.
2. Buat grup keamanan VPC untuk instans EC2 Anda (server web dan aplikasi). Jika diperlukan, grup keamanan ini dapat mengizinkan akses ke instans EC2 dari Internet dengan menggunakan tabel perutean VPC. Sebagai contoh, Anda dapat menetapkan aturan pada grup keamanan ini untuk mengizinkan akses TCP ke instans EC2 melalui port 22.
3. Buat aturan kustom di grup keamanan untuk cache Anda yang memungkinkan koneksi dari grup keamanan yang Anda buat untuk instans EC2 Anda. Hal ini akan mengizinkan semua anggota grup keamanan untuk mengakses cache.

> **Note**
>
> Jika Anda berencana untuk menggunakan Zona Lokal, pastikan Anda telah mengaktifkannya. Saat Anda membuat grup subnet di zona lokal, VPC Anda diperluas ke Zona Lokal tersebut dan VPC Anda akan memperlakukan subnet itu seperti subnet lain di Zona Ketersediaan lainnya. Semua gateway dan tabel rute yang berkaitan akan disesuaikan secara otomatis.

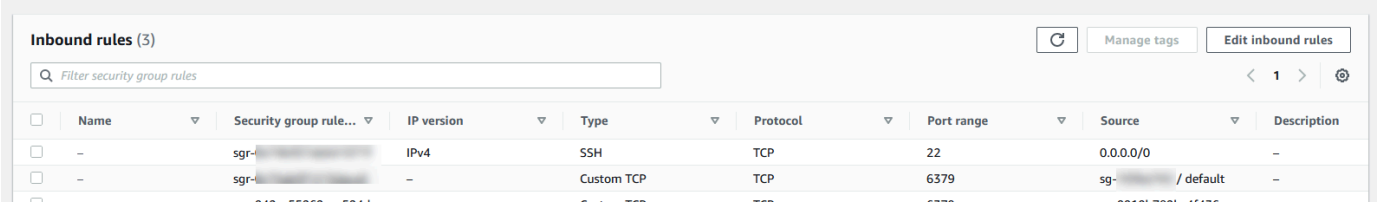
Untuk membuat aturan dalam grup keamanan VPC yang memungkinkan koneksi dari grup keamanan lain

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon VPC di https://console.aws.amazon.com/vpc.
2. Pada panel navigasi, pilih Grup Keamanan.
3. Pilih atau buat grup keamanan yang akan Anda gunakan untuk cache Anda. Di bawah Aturan Masuk, pilih Edit Aturan Masuk lalu pilih Tambahkan Aturan. Grup keamanan ini akan mengizinkan akses bagi anggota dari grup keamanan lain.
4. Dari Jenis, pilih Aturan TCP Khusus.

   a. Untuk Rentang Port, tentukan port yang Anda gunakan saat membuat cache.

      Port default untuk cache dan grup replikasi Redis adalah 6379.

   b. Pada kotak Sumber, masukkan ID dari grup keamanan. Dari daftar, pilih grup keamanan yang akan Anda gunakan untuk instans Amazon EC2 Anda.
5. Pilih Simpan jika selesai.




Mengakses Cache ElastiCache dan Instans Amazon EC2 yang berada di Amazon VPC yang Berbeda

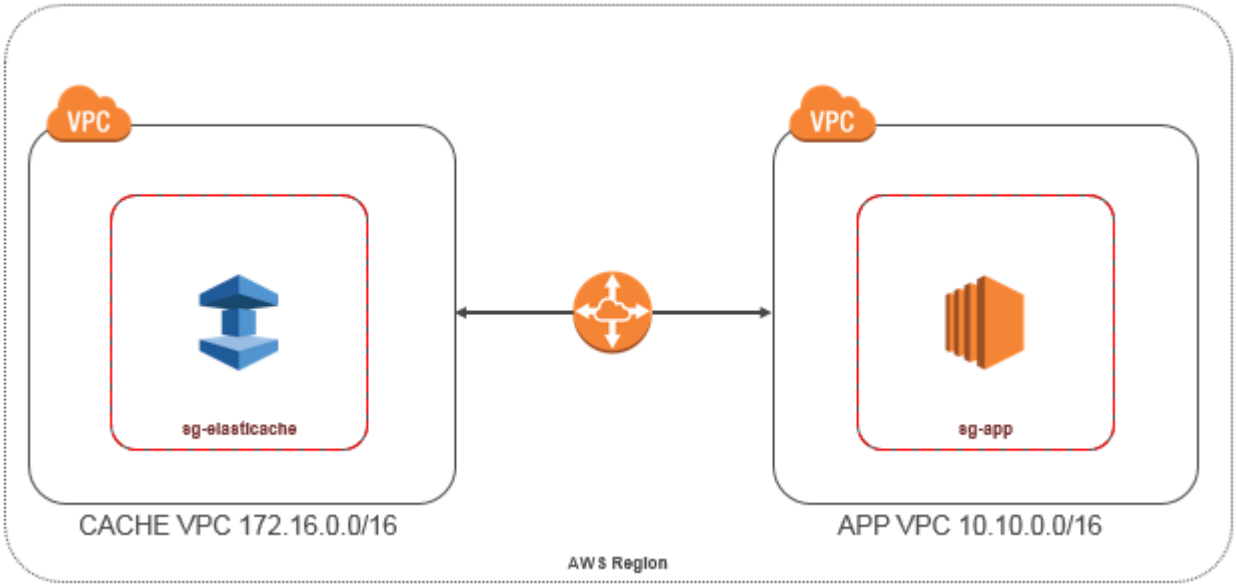
Jika cache Anda berada dalam VPC yang berbeda dengan instans EC2 yang Anda gunakan untuk mengaksesnya, ada beberapa cara untuk mengakses cache. Jika instans EC2 ada di VPC yang berbeda tetapi di wilayah yang sama, Anda dapat menggunakan peering VPC. Jika cache dan instans EC2 berada di wilayah yang berbeda, Anda dapat membuat konektivitas VPN di antara wilayah.

Topik

- Mengakses Cache ElastiCache dan Instans Amazon EC2 yang berada di Amazon VPC yang Berbeda di Wilayah yang Sama
- Mengakses Cache ElastiCache dan Instans Amazon EC2 yang berada di Amazon VPC yang Berbeda di Wilayah yang Berbeda

### Mengakses Cache ElastiCache dan Instans Amazon EC2 yang berada di Amazon VPC yang Berbeda di Wilayah yang Sama

Diagram berikut menunjukkan pengaksesan cache oleh instans Amazon EC2 di Amazon VPC yang berbeda di wilayah yang sama menggunakan koneksi peering VPC.




Cache yang diakses oleh instans Amazon EC2 di Amazon VPC yang berbeda di Wilayah yang sama - Koneksi Peering VPC

Koneksi peering VPC adalah koneksi jaringan antara dua VPC yang memungkinkan Anda merutekan lalu lintas di antara keduanya menggunakan alamat IP privat. Instans di kedua VPC tersebut dapat berkomunikasi satu sama lain seolah berada di jaringan yang sama. Anda dapat membuat koneksi peering VPC antara Amazon VPC Anda sendiri, atau dengan Amazon VPC di akun AWS lain dalam satu wilayah. Untuk mempelajari selengkapnya peering Amazon VPC, lihat Dokumentasi VPC.

> **Note**
>
> Resolusi nama DNS mungkin gagal untuk VPC yang telah di-peering, tergantung pada konfigurasi yang diterapkan ke VPC ElastiCache. Untuk mengatasi hal ini, kedua VPC harus mengaktifkan nama host DNS dan resolusi DNS. Untuk informasi selengkapnya, lihat Mengaktifkan resolusi DNS untuk koneksi peering VPC.

Untuk mengakses cache di Amazon VPC yang berbeda melalui peering

1. Pastikan bahwa kedua VPC tidak memiliki rentang IP yang tumpang tindih atau Anda tidak akan melakukan peering.
2. Buat koneksi peering di antara kedua VPC. Untuk informasi selengkapnya, lihat Membuat dan Menerima Koneksi Peering VPC Amazon.
3. Perbarui tabel rute Anda. Untuk informasi selengkapnya, lihat.Memperbarui Tabel Rute Anda untuk Koneksi Peering VPC

   Berikut adalah tampilan tabel rute untuk contoh pada diagram sebelumnya. Perhatikan bahwa pcx-a894f1c1 adalah koneksi peering.

| Destination | Target |
|---|---|
| 172.16.0.0/16 | local |
| 10.10.0.0/16 | pcx-a894f1c1 |

| Destination | Target |
|---|---|
| 10.10.0.0/16 | local |
| 0.0.0.0/0 | igw-bfdcccd8 |
| 172.16.0.0/16 | pcx-a894f1c1 |

   Tabel Perutean VPC
4. Ubah Grup Keamanan cache ElastiCache Anda agar mengizinkan koneksi masuk dari grup keamanan Aplikasi di dalam VPC yang tersambung peering. Untuk informasi selengkapnya, lihat Grup Keamanan VPC Peer Referensi.

Mengakses cache melalui koneksi peering akan dikenakan biaya transfer data tambahan.

Menggunakan Gateway Transit

Gateway transit memungkinkan Anda untuk menyematkan koneksi VPC dan VPN di Wilayah AWS yang sama dan mengarahkan rute lalu lintas di antara keduanya. Gateway transit bekerja di seluruh akun AWS, dan Anda dapat menggunakan AWS Resource Access Manager untuk berbagi transit gateway Anda dengan akun lain. Setelah Anda berbagi gateway transit dengan akun AWS lain, pemilik akun dapat menyematkan VPC mereka ke gateway transit Anda. Pengguna dari kedua akun dapat menghapus lampuran tersebut kapan saja.

Anda dapat mengaktifkan multicast pada gateway transit, lalu membuat domain multicast gateway transit yang mengizinkan lalu lintas multicast dikirim dari sumber multicast Anda untuk anggota grup multicast melalui lampiran VPC yang Anda kaitkan dengan domain.

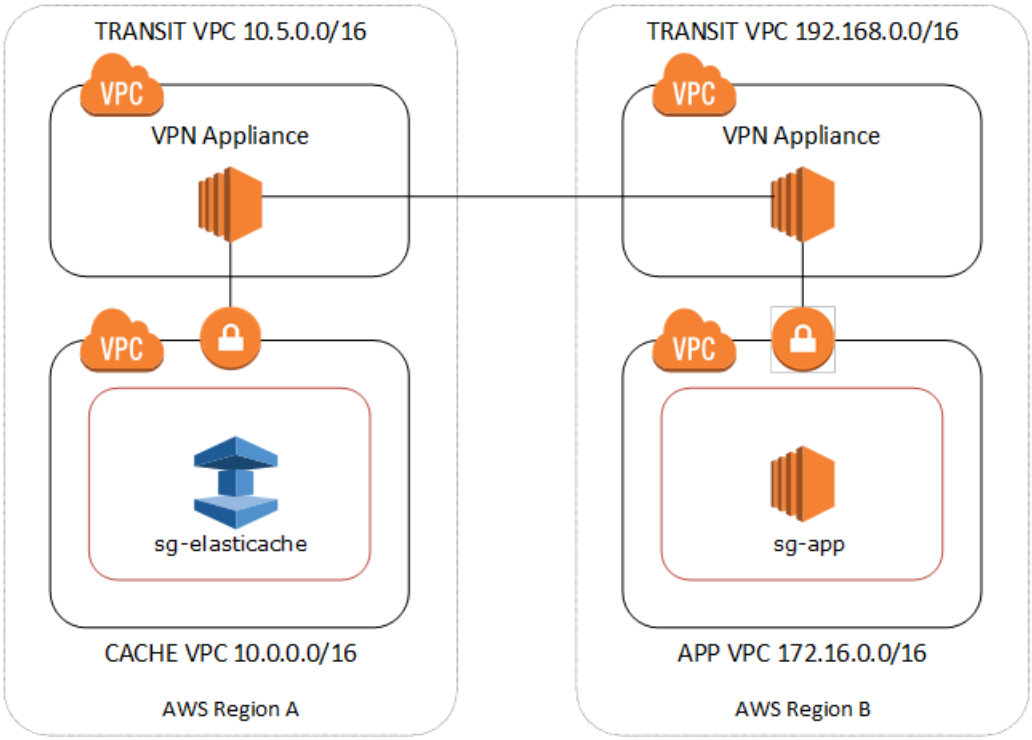
Anda juga dapat membuat lampiran koneksi peering antara gateway transi di Wilayah AWS yang berbeda. Hal ini memungkinkan Anda untuk mengatur rute lalu lintas antara beberapa lampiran gateway transit di Wilayah yang berbeda.

Untuk informasi selengkapnya, silakan lihat Gateway transit.

Mengakses Cache ElastiCache dan Instans Amazon EC2 yang berada di Amazon VPC yang Berbeda di Wilayah yang Berbeda

Menggunakan VPC Transit

Selain menggunakan peering VPC, strategi umum lain untuk menghubungkan beberapa VPC dan jaringan jauh yang berjauhan secara geografis adalah membuat VPC transit yang berfungsi sebagai pusat transit jaringan global. VPC transit menyederhanakan manajemen jaringan dan meminimalkan jumlah koneksi yang diperlukan untuk menghubungkan beberapa VPC dan jaringan jarak jauh. Rancangan ini dapat menghemat waktu dan tenaga serta mengurangi biaya, karena diimplementasikan secara virtual tanpa biaya tradisional untuk membangun kehadiran fisik di hub transit kolokasi atau menyebarkan peralatan jaringan fisik.




Menghubungkan seluruh VPC yang berbeda di wilayah yang berbeda

Setelah Amazon VPC Transit dibuat, aplikasi yang disebarkan di sebuah VPC “kisi” di satu wilayah dapat terhubung ke cache ElastiCache di VPC “kisi” di wilayah lain.

Mengakses cache di VPC yang berbeda di Wilayah AWS yang berbeda

1. Menyebarkan Solusi VPC Transit. Untuk informasi selengkapnya, lihat AWS Transit Gateway.
2. Perbarui tabel perutean VPC di VPC Cache dan Aplikasi untuk merutekan lalu lintas melalui VGW (Gateway Privat Virtual) dan Perangkat VPN. Dalam kasus Perutean Dinamis dengan Protokol Gateway Batas (BGP), rute Anda dapat disebarkan secara otomatis.
3. Ubah Grup Keamanan cache ElastiCache Anda agar mengizinkan koneksi masuk dari rentang IP instans Aplikasi. Perhatikan bahwa Anda tidak akan dapat mereferensikan Grup Keamanan server aplikasi dalam skenario ini.

Mengakses cache di seluruh wilayah akan menimbulkan latensi jaringan dan biaya transfer data lintas wilayah tambahan.

Mengakses Cache ElastiCache dari Aplikasi yang Berjalan di Pusat Data Pelanggan

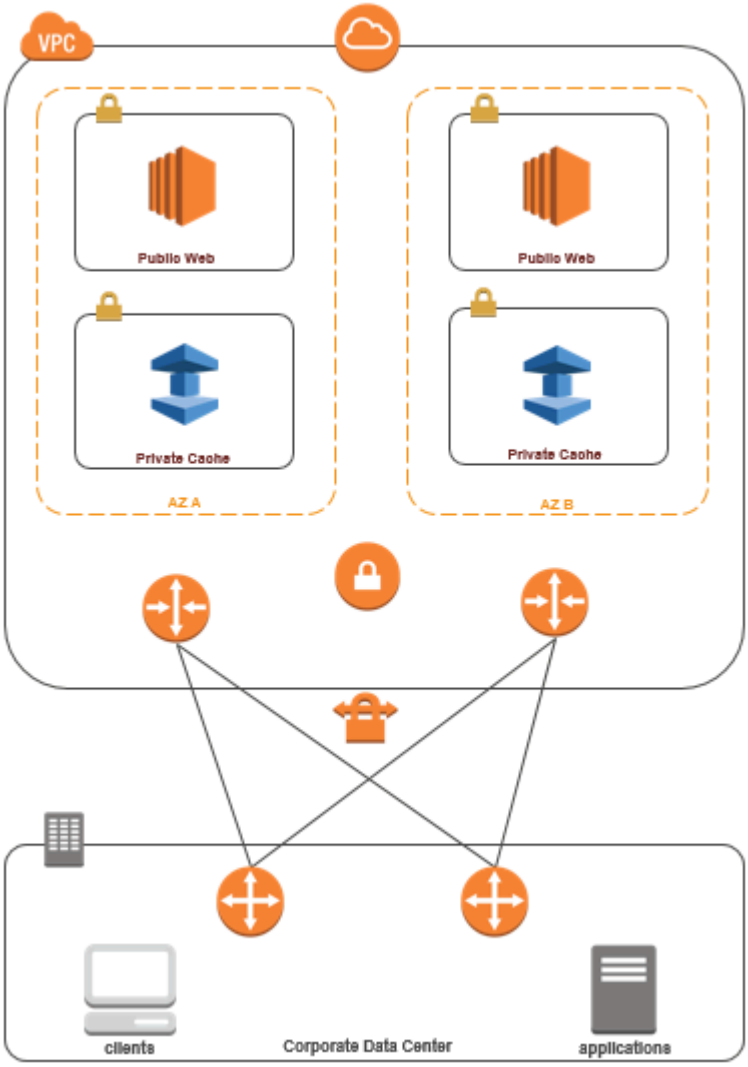
Skenario lain yang mungkin adalah arsitektur Hibrida di mana klien atau aplikasi di pusat data pelanggan mungkin perlu mengakses cache ElastiCache di VPC. Skenario ini juga didukung jika ada konektivitas antara VPC pelanggan dan pusat data baik melalui VPN atau Direct Connect.

Topik

- Mengakses Cache ElastiCache dari Aplikasi yang Berjalan di Pusat Data Pelanggan Menggunakan Konektivitas VPN
- Mengakses Cache ElastiCache dari Aplikasi yang Berjalan di Pusat Data Pelanggan Menggunakan Direct Connect

Mengakses Cache ElastiCache dari Aplikasi yang Berjalan di Pusat Data Pelanggan Menggunakan Konektivitas VPN

Diagram berikut menunjukkan pengaksesan cache ElastiCache dari aplikasi yang berjalan di jaringan perusahaan Anda menggunakan koneksi VPN.




Menyambungkan ke ElastiCache dari pusat data Anda melalui VPN

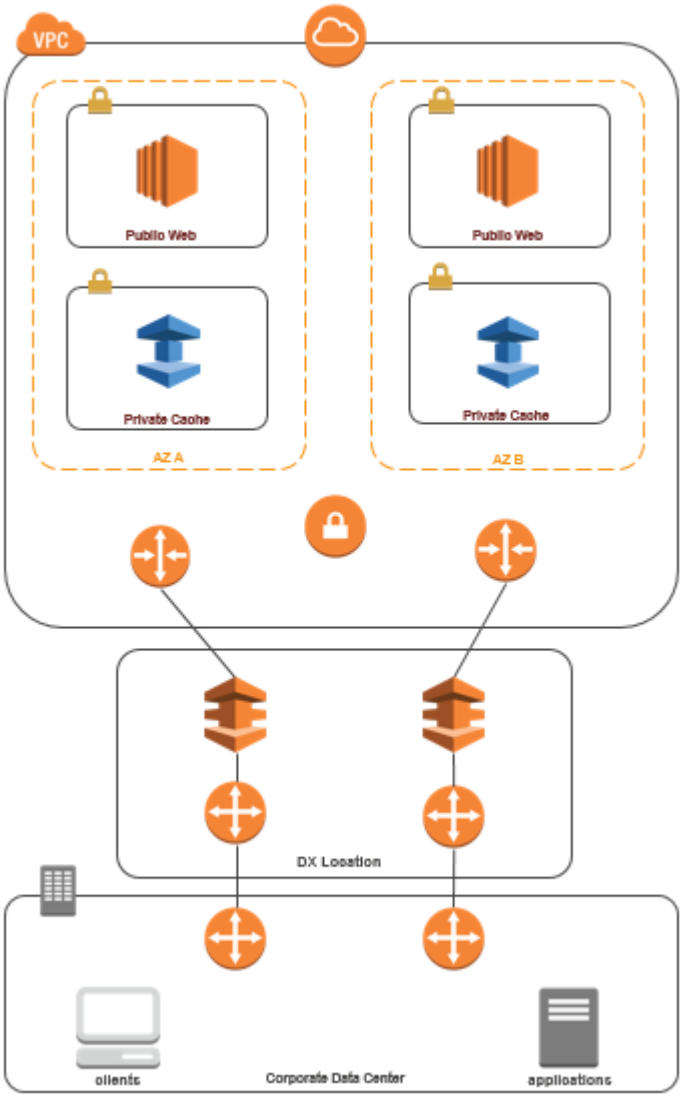
Untuk mengakses cache di VPC dari aplikasi on-premise melalui koneksi VPN

1. Buat Konektivitas VPN dengan menambahkan Gateway Privat Virtual perangkat keras ke VPC Anda. Untuk informasi selengkapnya, lihat Menambahkan Gateway Privat Virtual ke VPC Anda.
2. Perbarui tabel perutean VPC untuk subnet tempat cache ElastiCache Anda disebarkan guna memungkinkan lalu lintas dari server aplikasi on-premise Anda. Dalam kasus Perutean Dinamis dengan BGP, rute Anda dapat disebarkan secara otomatis.
3. Ubah Grup Keamanan cache ElastiCache Anda agar mengizinkan koneksi masuk dari server aplikasi on-premise.

Mengakses cache melalui koneksi VPN akan menimbulkan latensi jaringan dan biaya transfer data tambahan.

## Mengakses Cache ElastiCache dari Aplikasi yang Berjalan di Pusat Data Pelanggan Menggunakan Direct Connect

Diagram berikut menunjukkan pengaksesan cache ElastiCache dari aplikasi yang berjalan di jaringan perusahaan Anda menggunakan Direct Connect.




Menyambungkan ke ElastiCache dari pusat data Anda melalui Direct Connect

Mengakses cache ElastiCache dari aplikasi yang berjalan di jaringan Anda menggunakan Direct Connect

1. Buat konektivitas Direct Connect. Untuk informasi selengkapnya, lihat Memulai dengan AWS Direct Connect.
2. Ubah Grup Keamanan cache ElastiCache Anda agar mengizinkan koneksi masuk dari server aplikasi on-premise.

Mengakses cache melalui koneksi DX dapat menimbulkan latensi jaringan dan biaya transfer data tambahan.

## Membuat Cloud Privat Virtual (VPC)

Dalam contoh ini, Anda membuat Amazon VPC dengan subnet privat untuk setiap Zona Ketersediaan.

Membuat Amazon VPC (Konsol)

1. Login ke Konsol Manajemen AWS dan buka konsol Amazon VPC di https://console.aws.amazon.com/vpc/.
2. Pada dasbor VPC, pilih Buat VPC.
3. Di bagian Sumber daya yang akan dibuat, pilih VPC dan lainnya.
4. Di bagian Jumlah Zona Ketersediaan (AZ), pilih jumlah Zona Ketersediaan yang ingin Anda luncurkan subnet ke sana.
5. Di bagian Jumlah subnet publik, pilih jumlah subnet publik yang ingin Anda tambahkan ke VPC Anda.
6. Di bagian Jumlah subnet pribadi, pilih jumlah subnet pribadi yang ingin Anda tambahkan ke VPC Anda.

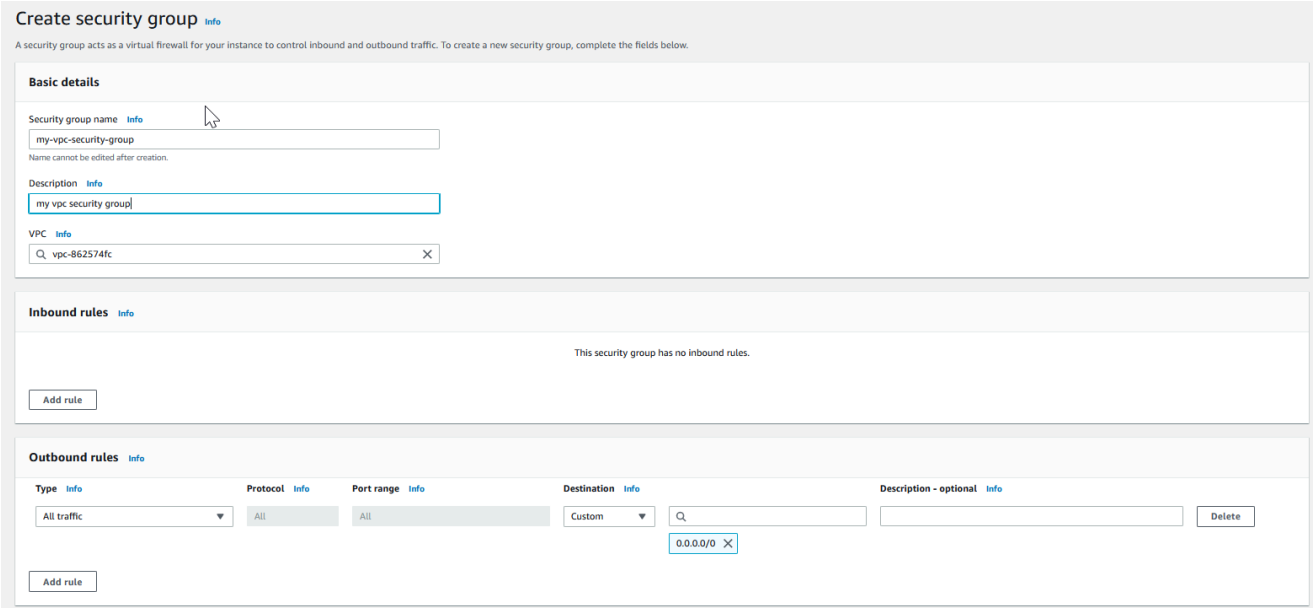
   > **Tip**
   >
   > Perhatikan pengidentifikasi subnet Anda, serta identifikasi mana yang publik dan privat. Anda akan membutuhkan informasi ini nanti saat meluncurkan klaster dan menambahkan instans Amazon EC2 ke Amazon VPC Anda.

7. Buat grup keamanan Amazon VPC. Anda akan menggunakan grup ini untuk klaster cache dan instans Amazon EC2 Anda.

   a. Pada panel navigasi Konsol Manajemen Amazon VPC, pilih Grup Keamanan.

   b. Pilih Buat Grup Keamanan.

   c. Ketikkan nama dan deskripsi untuk grup keamanan Anda di kotak yang sesuai. Di kotak VPC, pilih pengidentifikasi untuk Amazon VPC Anda.




d. Jika pengaturan sudah sesuai keinginan Anda, pilih Ya, Buat.

8. Tentukan aturan masuk jaringan untuk grup keamanan Anda. Aturan ini akan mengizinkan Anda terhubung ke instans Amazon EC2 dengan menggunakan Secure Shell (SSH).

   a. Di daftar navigasi, pilih Grup Keamanan.

   b. Temukan grup keamanan Anda di dalam daftar, lalu pilih grup tersebut.

   c. Di bagian Grup Keamanan, pilih tab Masuk. Di kotak Buat aturan baru, pilih SSH, lalu pilih Tambahkan Aturan.

   d. Tetapkan nilai berikut untuk aturan masuk baru yang akan mengizinkan akses HTTP:

      - Jenis: HTTP
      - Sumber: 0.0.0.0/0

      Pilih Terapkan Perubahan Aturan.

Sekarang Anda sudah siap untuk membuat grup subnet cache dan meluncurkan klaster cache di Amazon VPC Anda.

- Membuat grup subnet
- Membuat klaster Redis (Mode Klaster Dinonaktifkan) (Konsol).

## Menghubungkan ke cache yang berjalan di Amazon VPC

Contoh ini menunjukkan cara meluncurkan instans Amazon EC2 di Amazon VPC Anda. Anda kemudian dapat log masuk ke instans ini dan mengakses cache ElastiCache yang berjalan di Amazon VPC.

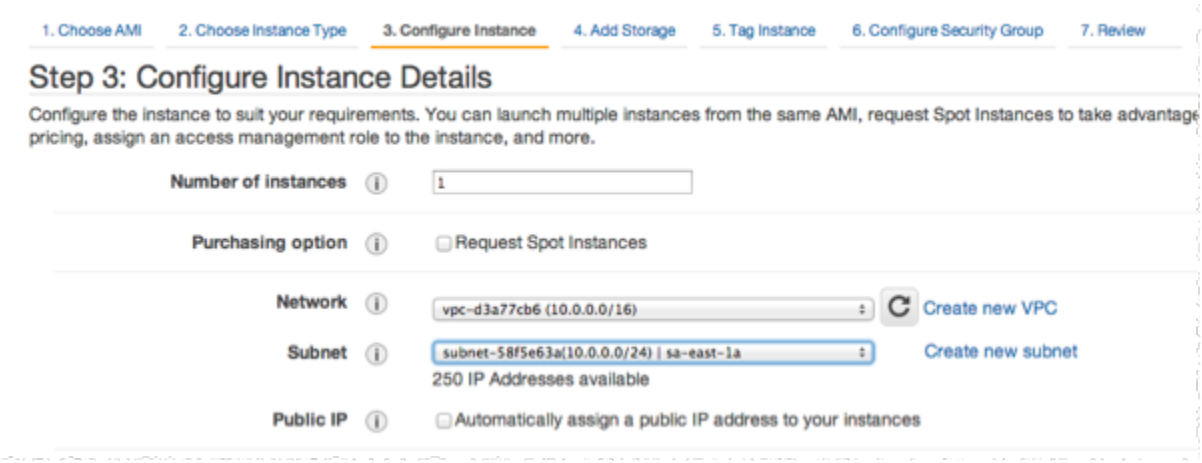
Menghubungkan ke cache yang berjalan di Amazon VPC (Konsol)

Dalam contoh ini, Anda membuat instans Amazon EC2 di Amazon VPC. Anda dapat menggunakan instans Amazon EC2 ini untuk terhubung ke simpul cache yang berjalan di Amazon VPC.

> **Note**
>
> Untuk informasi tentang menggunakan Amazon EC2, lihat Panduan Memulai Amazon EC2 di Dokumentasi Amazon EC2.

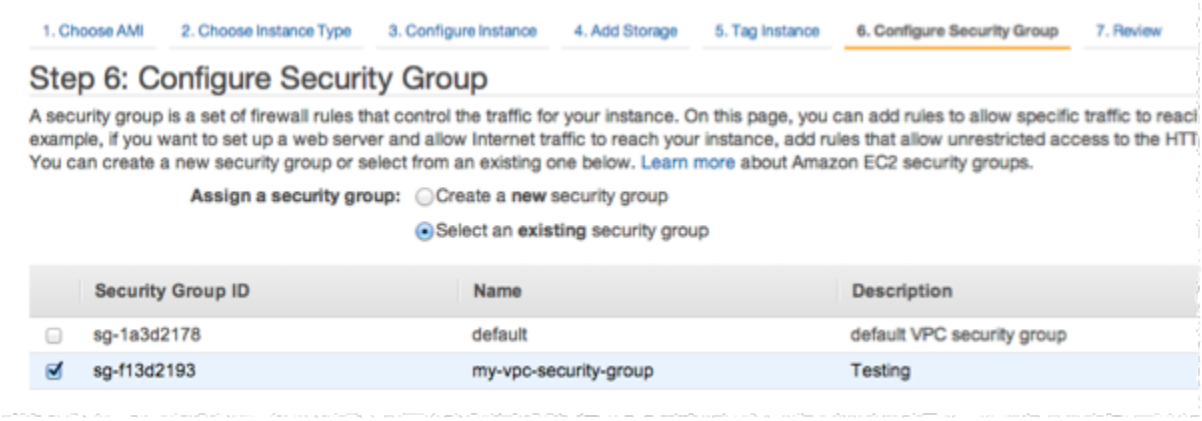
Untuk membuat instans Amazon EC2 di Amazon VPC dengan menggunakan konsol Amazon EC2

1. Masuk ke AWS Management Console lalu buka konsol Amazon EC2 di https://console.aws.amazon.com/ec2/.
2. Pada konsol, pilih Luncurkan Instans dan ikuti langkah-langkah ini:
3. Pada halaman Pilih Amazon Machine Image (AMI), pilih AMI Amazon Linux 64-bit, lalu pilihlah Pilih.
4. Pada halaman Pilih Jenis Instans, pilih 3. Konfigurasi Instans.
5. Pada halaman Konfigurasi Detail Instans, buat pilihan berikut:
   a. Di daftar Jaringan, pilih Amazon VPC Anda.
   b. Di daftar Subnet, pilih subnet publik Anda.




Jika pengaturan sudah sesuai keinginan Anda, pilih 4. Tambahkan Penyimpanan.

6. Pada halaman Tambahkan Penyimpanan, pilih 5. Tandai Instans.
7. Pada halaman Tandai Instans, ketikkan nama untuk instans Amazon EC2 Anda, dan kemudian pilih 6. Konfigurasi Grup Keamanan.
8. Pada halaman Konfigurasi Grup Keamanan, pilih Pilih grup keamanan yang sudah ada. Untuk informasi lebih lanjut tentang grup keamanan, silakan lihat Grup keamanan Amazon EC2 untuk instans Linux.




Pilih nama grup keamanan Amazon VPC Anda, dan kemudian pilih Tinjau dan Luncurkan.

9. Di halaman Tinjau Instans dan Luncurkan, pilih Luncurkan.
10. Di jendela Pilih pasangan kunci yang sudah ada atau buat pasangan kunci baru, tentukan pasangan kunci yang ingin Anda gunakan dengan instans ini.

> **Note**
>
> Untuk informasi tentang mengelola pasangan kunci, lihat Panduan Memulai Amazon EC2.

11. Ketika Anda siap untuk meluncurkan instans Amazon EC2 Anda, pilih Luncurkan instans.

Sekarang Anda dapat menetapkan alamat IP Elastis untuk instans Amazon EC2 yang baru saja Anda buat. Anda harus menggunakan alamat IP ini untuk terhubung ke instans Amazon EC2.

Untuk menetapkan alamat IP elastis (Konsol)

1. Buka konsol Amazon VPC di https://console.aws.amazon.com/vpc/.
2. Di daftar navigasi, pilih IP Elastis.
3. Pilih Alokasi alamat IP elastis.
4. Di kotak dialog Alokasi Alamat IP Elastis, terima Grup Perbatasan Jaringan default lalu pilih Alokasikan.
5. Pilih alamat IP Elastis yang baru saja Anda alokasikan dari daftar lalu pilih  Kaitkan Alamat.
6. Di kotak dialog Kaitkan Alamat, di kotak Instans, pilih ID instans Amazon EC2 yang Anda luncurkan.

   Di kotak Alamat IP privat, pilih kotak untuk mendapatkan alamat IP privat dan kemudian pilih Kaitkan.

   Sekarang Anda dapat menggunakan SSH untuk terhubung ke instans Amazon EC2 menggunakan alamat IP Elastis yang Anda buat.

Hubungkan ke instans Amazon EC2 Anda

- Buka jendela perintah. Pada prompt perintah, keluarkan perintah berikut, menggantikan mykeypair.pem dengan nama file pasangan kunci Anda dan 54.207.55.251 dengan alamat IP Elastis Anda.

```
ssh -i mykeypair.pem ec2-user@54.207.55.251
```

> **Important**
>
> Jangan dulu log keluar dari instans Amazon EC2 Anda.

Sekarang Anda siap untuk berinteraksi dengan klaster ElastiCache Anda. Sebelum Anda dapat melakukannya, Anda perlu menginstal utilitas telnet jika Anda belum melakukannya.

Untuk menginstal telnet dan berinteraksi dengan klaster cache Anda (AWS CLI)

1. Buka jendela perintah. Di prompt perintah, keluarkan perintah berikut. Pada prompt konfirmasi, ketik y.

```
sudo yum install telnet
Loaded plugins: priorities, security, update-motd, upgrade-helper
Setting up Install Process
Resolving Dependencies
--> Running transaction check

...(output omitted)...

Total download size: 63 k
Installed size: 109 k
Is this ok [y/N]: y
Downloading Packages:
telnet-0.17-47.7.amzn1.x86_64.rpm                          |  63 kB     00:00

...(output omitted)...

Complete!
```

2. Gunakan telnet untuk tersambung ke titik akhir simpul cache Anda melalui port 6379. Ganti nama host yang ditunjukkan di bawah ini dengan nama host dari simpul cache Anda.

```
telnet my-cache-cluster.7wufxa.0001.use1.cache.amazonaws.com 6379
```

   Anda sekarang terhubung ke mesin cache dan dapat mengeluarkan perintah. Dalam contoh ini, Anda menambahkan item data ke cache dan kemudian mendapatkannya segera sesudahnya. Akhirnya, Anda akan terputus sambungan dari simpul cache.

Untuk menyimpan kunci dan nilai, ketik dua baris berikut:

```
set mykey myvalue
```

Mesin cache merespons dengan berikut ini:

```
OK
```

Untuk mengambil nilai untuk `mykey`, ketik berikut ini:

```
get mykey
```

Untuk memutuskan sambungan dari mesin cache, ketik yang berikut ini:

```
quit
```

3. Buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/ dan dapatkan titik akhir untuk salah satu simpul dalam klaster cache Anda. Untuk informasi selengkapnya, silakan lihat Mendapatkan temuan titik akhir sambungan untuk Redis.
4. Gunakan telnet untuk tersambung ke titik akhir simpul cache Anda melalui port 6379. Ganti nama host yang ditunjukkan di bawah ini dengan nama host dari simpul cache Anda.

```
telnet my-cache-cluster.7wufxa.0001.use1.cache.amazonaws.com 6379
```

Anda sekarang terhubung ke mesin cache dan dapat mengeluarkan perintah. Dalam contoh ini, Anda menambahkan item data ke cache dan kemudian mendapatkannya segera sesudahnya. Akhirnya, Anda akan terputus sambungan dari simpul cache.

Untuk menyimpan kunci dan nilai, ketik yang berikut ini:

```
set mykey myvalue
```

Mesin cache merespons dengan berikut ini:

```
OK
```

Untuk mengambil nilai untuk `mykey`, ketik berikut ini:

```
get mykey
```

Mesin cache merespons dengan berikut ini:

```
get mykey
myvalue
```

Untuk memutuskan sambungan dari mesin cache, ketik yang berikut ini:

```
quit
```

> **Important**
>
> Agar tidak dikenakan biaya tambahan pada akun AWS Anda, pastikan untuk menghapus segala sumber daya AWS yang tidak lagi Anda inginkan setelah mencoba contoh-contoh ini.

## Titik akhir VPC dan API Amazon ElastiCache (AWS PrivateLink)

Anda dapat membuat koneksi privat antara titik akhir VPC dengan membuat titik akhir VPC dan API Amazon ElastiCache dengan membuat titik akhir VPC antarmuka. Titik akhir antarmuka didukung oleh AWS PrivateLink. AWS PrivateLink memungkinkan Anda mengakses operasi API Amazon ElastiCache tanpa Gateway internet, perangkat NAT, koneksi VPN, atau koneksi AWS Direct Connect.

Instans dalam VPC Anda tidak memerlukan alamat IP publik untuk berkomunikasi dengan titik akhir API Amazon ElastiCache. Instans Anda juga tidak memerlukan alamat IP publik untuk menggunakan operasi API ElastiCache yang tersedia. Lalu lintas antara VPC dan Amazon ElastiCache tidak keluar dari jaringan Amazon. Setiap titik akhir antarmuka diwakili oleh satu atau beberapa antarmuka jaringan elastis di subnet Anda. Untuk informasi selengkapnya tentang antarmuka jaringan elastis, lihat Antarmuka jaringan elastis dalam Panduan Pengguna Amazon EC2.

- Untuk informasi selengkapnya titik akhir VPC, lihat Titik Akhir VPC Antarmuka (AWS PrivateLink) dalam Panduan Pengguna Amazon VPC.
- Untuk informasi selengkapnya operasi API ElastiCache, lihat Operasi API ElastiCache.

Setelah membuat titik akhir VPC antarmuka, jika Anda mengaktifkan nama host DNS privat untuk titik akhir, maka titik akhir ElastiCache default (https://elasticache.*Region*.amazonaws.com) menyelesaikan titik akhir VPC Anda. Jika Anda tidak mengaktifkan nama host DNS privat, Amazon VPC menyediakan nama titik akhir DNS yang dapat Anda gunakan dalam format berikut:

```
VPC_Endpoint_ID.elasticache.Region.vpce.amazonaws.com
```

Untuk informasi selengkapnya, lihat Titik Akhir VPC Antarmuka (AWS PrivateLink) dalam Panduan Pengguna Amazon VPC. ElastiCache mendukung pembuatan panggilan ke semua Tindakan API dalam VPC Anda.

> **Note**
>
> Nama host DNS privat dapat diaktifkan hanya untuk satu titik akhir VPC di VPC. Jika Anda ingin membuat titik akhir VPC tambahan maka nama host DNS privat harus dinonaktifkan.

## Pertimbangan untuk titik akhir VPC

Sebelum Anda menyiapkan titik akhir VPC antarmuka untuk titik akhir API Amazon ElastiCache, pastikan bahwa Anda meninjau Properti dan batasan titik akhir antarmuka dalam Panduan Pengguna Amazon VPC. Semua operasi API ElastiCache yang relevan dengan pengelolaan sumber daya Amazon ElastiCache tersedia dari VPC Anda menggunakan AWS PrivateLink.

Kebijakan titik akhir VPC didukung untuk titik akhir API ElastiCache. Secara default, akses penuh ke operasi API ElastiCache diizinkan melalui titik akhir. Untuk informasi selengkapnya, lihat Mengontrol Akses ke Layanan dengan titik akhir VPC dalam Panduan Pengguna Amazon VPC.

## Membuat titik akhir VPC antarmuka untuk API ElastiCache

Anda dapat membuat titik akhir VPC untuk API Amazon ElastiCache menggunakan konsol Amazon VPC atau AWS CLI. Untuk informasi selengkapnya, lihat Membuat titik akhir antarmuka dalam Panduan Pengguna Amazon VPC.

Setelah membuat titik akhir VPC antarmuka, Anda dapat mengaktifkan nama host DNS privat untuk titik akhir. Ketika Anda melakukannya, titik akhir Amazon ElastiCache default (https://elasticache.*Region*.amazonaws.com) menyelesaikan titik akhir VPC Anda. Untuk Wilayah AWS Tiongkok (Beijing) dan Tiongkok (Ningxia), Anda dapat membuat permintaan API dengan titik akhir VPC menggunakan `elasticache.cn-north-1.amazonaws.com.cn` untuk Beijing

dan `elasticache.cn-northwest-1.amazonaws.com.cn` untuk Ningxia. Untuk informasi selengkapnya, lihat Mengakses layanan melalui titik akhir antarmuka dalam Panduan Pengguna Amazon VPC.

## Membuat kebijakan titik akhir VPC untuk API Amazon ElastiCache

Anda dapat melampirkan kebijakan titik akhir ke titik akhir VPC Anda yang mengontrol akses ke API ElastiCache. Kebijakan menentukan informasi berikut:

- Pengguna utama yang dapat melakukan tindakan.
- Tindakan-tindakan yang dapat dilakukan.
- Sumber daya yang menjadi objek tindakan.

Untuk informasi selengkapnya, lihat Mengontrol Akses ke Layanan dengan titik akhir VPC dalam Panduan Pengguna Amazon VPC.

Example Kebijakan titik akhir VPC untuk tindakan API ElastiCache

Berikut adalah contoh kebijakan titik akhir untuk API ElastiCache. Saat dilampirkan ke sebuah titik akhir, kebijakan ini memberikan akses ke tindakan yang terdaftar untuk semua pengguna utama pada semua sumber daya.

```
{
 "Statement": [{
  "Principal": "*",
  "Effect": "Allow",
  "Action": [
   "elasticache:CreateCacheCluster",
   "elasticache:ModifyCacheCluster",
   "elasticache:CreateSnapshot"
  ],
  "Resource": "*"
 }]
}
```

Example Kebijakan titik akhir VPC yang menolak semua akses dari akun AWS tertentu

Kebijakan titik akhir VPC berikut menolak akun AWS *123456789012* dari semua akses ke sumber daya yang menggunakan titik akhir tersebut. Kebijakan ini memungkinkan semua tindakan dari akun lain.

```
{
 "Statement": [{
   "Action": "*",
   "Effect": "Allow",
   "Resource": "*",
   "Principal": "*"
  },
  {
   "Action": "*",
   "Effect": "Deny",
   "Resource": "*",
   "Principal": {
    "AWS": [
     "123456789012"
    ]
   }
  }
 ]
}
```

## Subnet dan grup subnet

Grup subnet adalah sekumpulan subnet (biasanya privat) yang dapat Anda tetapkan untuk klaster Anda yang dirancang sendiri di lingkungan Amazon Virtual Private Cloud (Amazon VPC).

Jika Anda membuat klaster yang dirancang sendiri di Amazon VPC, Anda harus menggunakan grup subnet. ElastiCache menggunakan grup subnet tersebut untuk memilih subnet dan alamat IP dalam subnet tersebut untuk mengasosiasikannya dengan simpul Anda.

ElastiCache menyediakan grup subnet IPv4 default atau Anda dapat memilih untuk membuat yang baru. Untuk IPv6, Anda perlu membuat grup subnet dengan blok CIDR IPv6. Jika Anda memilih dual-stack, Anda harus memilih jenis IP Discovery, baik IPv6 atau IPv4.

ElastiCache Nirserver tidak menggunakan sumber daya grup subnet, dan sebagai gantinya mengambil daftar subnet secara langsung selama pembuatan.

Bagian ini membahas cara membuat dan memanfaatkan subnet dan grup subnet untuk mengelola akses ke sumber daya ElastiCache Anda.

Untuk informasi selengkapnya tentang penggunaan grup subnet di lingkungan Amazon VPC, lihat Mengakses klaster atau grup replikasi Anda.

Topik

- Membuat grup subnet
- Menetapkan grup subnet ke cache
- Mengubah grup subnet
- Menghapus grup subnet

## Membuat grup subnet

Grup subnet cache adalah kumpulan subnet yang dapat ditetapkan untuk cache Anda di dalam VPC. Saat meluncurkan cache di VPC, Anda harus memilih grup subnet cache. Kemudian ElastiCache menggunakan grup subnet cache tersebut untuk memberikan alamat IP di dalam subnet tersebut ke setiap simpul cache di dalam cache.

Saat Anda membuat grup subnet baru, perhatikan jumlah alamat IP yang tersedia. Jika subnet memiliki sangat sedikit alamat IP yang tersedia, Anda akan dibatasi dalam hal jumlah simpul yang dapat ditambahkan ke klaster. Untuk mengatasi masalah ini, Anda dapat menetapkan satu atau beberapa subnet ke grup subnet sehingga Anda memiliki jumlah alamat IP yang cukup di dalam Zona Ketersediaan dari klaster Anda. Setelah itu, Anda dapat menambahkan lebih banyak simpul ke klaster Anda.

Jika Anda memilih IPV4 sebagai jenis jaringan Anda, grup subnet default akan tersedia atau Anda dapat memilih untuk membuat yang baru. ElastiCache menggunakan grup subnet tersebut untuk memilih subnet dan alamat IP dalam subnet tersebut yang akan dikaitkan dengan simpul Anda. Jika Anda memilih dual-stack atau IPV6, Anda akan diarahkan untuk membuat subnet dual-stack atau IPV6. Untuk informasi selengkapnya tentang jenis jaringan, lihat Jenis jaringan. Untuk informasi selengkapnya, lihat Membuat subnet di VPC Anda.

Prosedur berikut menunjukkan cara membuat grup subnet yang disebut `mysubnetgroup` (konsol), AWS CLI, dan API ElastiCache.

### Membuat grup subnet (Konsol)

Prosedur berikut menunjukkan cara membuat grup subnet (konsol).

Untuk membuat grup subnet (Konsol)

1. Masuk ke Konsol Manajemen AWS dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Di daftar navigasi, pilih Grup subnet.
3. Pilih Buat grup subnet.
4. Pada wizard Buat Grup Subnet, lakukan hal berikut. Jika semua pengaturan sudah sesuai keinginan Anda, pilih Buat.
   a. Pada kotak Nama, ketik nama grup subnet Anda.
   b. Pada kotak Deskripsi, ketik deskripsi untuk grup subnet Anda.

c. Di kotak ID VPC, pilih Amazon VPC Anda.

d. Semua subnet dipilih secara default. Di panel Subnet terpilih, klik Kelola dan pilih Zona Ketersediaan atau Zona Lokal dan ID subnet pribadi Anda, lalu pilihlah Pilih.

5. Pada pesan konfirmasi yang muncul, pilih Tutup.

Grup subnet baru Anda muncul pada daftar Grup Subnet dari konsol ElastiCache. Di bagian bawah jendela Anda dapat memilih grup subnet untuk melihat detailnya, misalnya semua subnet yang terkait dengan grup ini.

Membuat grup subnet (AWS CLI)

Pada prompt perintah, gunakan perintah `create-cache-subnet-group` untuk membuat grup subnet.

Untuk Linux, macOS, atau Unix:

```
aws elasticache create-cache-subnet-group \
    --cache-subnet-group-name mysubnetgroup \
    --cache-subnet-group-description "Testing" \
    --subnet-ids subnet-53df9c3a
```

Untuk Windows:

```
aws elasticache create-cache-subnet-group ^
    --cache-subnet-group-name mysubnetgroup ^
    --cache-subnet-group-description "Testing" ^
    --subnet-ids subnet-53df9c3a
```

Perintah ini seharusnya menghasilkan keluaran yang serupa dengan yang berikut:

```
{
    "CacheSubnetGroup": {
        "VpcId": "vpc-37c3cd17",
        "CacheSubnetGroupDescription": "Testing",
        "Subnets": [
            {
                "SubnetIdentifier": "subnet-53df9c3a",
                "SubnetAvailabilityZone": {
                    "Name": "us-west-2a"
                }
```

```
            }
        ],
        "CacheSubnetGroupName": "mysubnetgroup"
    }
}
```

Untuk informasi selengkapnya, lihat AWS CLI topik create-cache-subnet-group.

## Menetapkan grup subnet ke cache

Setelah Anda membuat grup subnet, Anda dapat meluncurkan cache di Amazon VPC. Untuk informasi selengkapnya, lihat hal berikut.

- Klaster Redis mandiri – Untuk meluncurkan klaster Redis simpul tunggal, lihat Membuat klaster Redis (Mode Klaster Dinonaktifkan) (Konsol). Pada langkah 7.a (Pengaturan Redis Lanjutan), pilih grup subnet VPC.
- Grup replikasi Redis (mode klaster nonaktif) – Untuk meluncurkan grup replikasi Redis (mode klaster nonaktif) dalam VPC, lihat Membuat grup replikasi Redis (Mode Klaster Dinonaktifkan) dari awal. Pada langkah 7.b (Pengaturan Redis Lanjutan), pilih grup subnet VPC.
- Grup replikasi Redis (mode klaster aktif) – Membuat sebuah klaster Redis (Mode Klaster Diaktifkan) (Konsol). Pada langkah 6.i (Pengaturan Redis Lanjutan), pilih grup subnet VPC.

## Mengubah grup subnet

Anda dapat mengubah deskripsi grup subnet, atau mengubah daftar ID subnet yang terhubung dengan grup subnet. Anda tidak dapat menghapus ID subnet dari grup subnet jika cache saat ini menggunakan subnet tersebut.

Prosedur berikut menunjukkan cara mengubah grup subnet.

### Mengubah grup subnet (Konsol)

Untuk mengubah grup subnet

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Di panel navigasi, pilih Grup Subnet.
3. Pada daftar grup subnet, pilih tombol radio yang ingin Anda ubah dan pilih Ubah.
4. Di panel subnet yang dipilih, pilih Kelola.
5. Buat perubahan apa pun pada subnet yang dipilih dan klik Pilih.
6. Klik Simpan perubahan untuk menyimpan perubahan Anda.

### Mengubah grup subnet (AWS CLI)

Pada prompt perintah, gunakan perintah `modify-cache-subnet-group` untuk mengubah grup subnet.

Untuk Linux, macOS, atau Unix:

```
aws elasticache modify-cache-subnet-group \
    --cache-subnet-group-name mysubnetgroup \
    --cache-subnet-group-description "New description" \
    --subnet-ids "subnet-42df9c3a" "subnet-48fc21a9"
```

Untuk Windows:

```
aws elasticache modify-cache-subnet-group ^
    --cache-subnet-group-name mysubnetgroup ^
    --cache-subnet-group-description "New description" ^
    --subnet-ids "subnet-42df9c3a" "subnet-48fc21a9"
```

Perintah ini seharusnya menghasilkan keluaran yang serupa dengan yang berikut:

```
{
    "CacheSubnetGroup": {
        "VpcId": "vpc-73cd3c17",
        "CacheSubnetGroupDescription": "New description",
        "Subnets": [
            {
                "SubnetIdentifier": "subnet-42dcf93a",
                "SubnetAvailabilityZone": {
                    "Name": "us-west-2a"
                }
            },
            {
                "SubnetIdentifier": "subnet-48fc12a9",
                "SubnetAvailabilityZone": {
                    "Name": "us-west-2a"
                }
            }
        ],
        "CacheSubnetGroupName": "mysubnetgroup"
    }
}
```

Untuk informasi selengkapnya, lihat AWS CLI topik modify-cache-subnet-group.

## Menghapus grup subnet

Jika Anda memutuskan bahwa Anda tidak lagi memerlukan grup subnet, Anda dapat menghapusnya. Anda tidak dapat menghapus grup subnet jika saat ini digunakan oleh cache.

Prosedur berikut menunjukkan cara menghapus grup subnet.

### Menghapus grup subnet (Konsol)

Untuk menghapus grup subnet

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Di panel navigasi, pilih Grup Subnet.
3. Pada daftar grup subnet, pilih grup subnet yang ingin dihapus, dan kemudian pilih Hapus.
4. Saat Anda diminta untuk mengonfirmasi operasi ini, ketik nama grup subnet di kolom input teks dan pilih Hapus.

### Menghapus grup subnet (AWS CLI)

Menggunakan AWS CLI, memanggil perintah delete-cache-subnet-group dengan parameter berikut:

- `--cache-subnet-group-name`*`mysubnetgroup`*

Untuk Linux, macOS, atau Unix:

```
aws elasticache delete-cache-subnet-group \
    --cache-subnet-group-name mysubnetgroup
```

Untuk Windows:

```
aws elasticache delete-cache-subnet-group ^
    --cache-subnet-group-name mysubnetgroup
```

Perintah ini tidak menghasilkan output.

Untuk informasi selengkapnya, lihat AWS CLI topik delete-cache-subnet-group.

# Manajemen Identitas dan Akses untuk Amazon ElastiCache

AWS Identity and Access Management (IAM) adalah Layanan AWS yang membantu administrator mengendalikan akses ke sumber daya AWS secara aman. Administrator IAM mengontrol siapa yang dapat diautentikasi (masuk) dan diotoriasasi (memiliki izin) untuk menggunakan sumber daya ElastiCache. IAM adalah Layanan AWS yang dapat Anda gunakan tanpa dikenakan biaya tambahan.

Topik



## Audiens

Cara Anda menggunakan (IAM) AWS Identity and Access Management berbeda, tergantung pada pekerjaan yang Anda lakukan di ElastiCache.

Pengguna layanan – Jika Anda menggunakan layanan ElastiCache untuk melakukan tugas Anda, administrator Anda akan memberikan kredensial dan izin yang Anda butuhkan. Saat Anda menggunakan lebih banyak fitur ElastiCache untuk melakukan pekerjaan, Anda mungkin memerlukan izin tambahan. Memahami cara akses dikelola dapat membantu Anda meminta izin yang tepat dari administrator Anda. Jika Anda tidak dapat mengakses suatu fitur di ElastiCache, silakan lihat Pemecahan masalah identitas dan akses Amazon ElastiCache.

Administrator layanan – Jika Anda bertanggung jawab atas sumber daya ElastiCache di perusahaan Anda, Anda mungkin memiliki akses penuh ke ElastiCache. Merupakan tugas Anda untuk menentukan fitur dan sumber daya ElastiCache mana yang dapat diakses oleh pengguna layanan Anda. Kemudian, Anda harus mengirimkan permintaan kepada administrator IAM untuk mengubah

izin pengguna layanan Anda. Tinjau informasi di halaman ini untuk memahami konsep dasar IAM. Untuk mempelajari selengkapnya cara perusahaan Anda dapat menggunakan IAM dengan ElastiCache, silakan lihat Cara Amazon ElastiCache bekerja dengan IAM.

Administrator IAM – Jika Anda seorang administrator IAM, Anda mungkin ingin belajar dengan lebih detail cara Anda dapat menulis kebijakan untuk mengelola akses ke ElastiCache. Untuk melihat contoh kebijakan berbasis identitas ElastiCache yang dapat Anda gunakan di IAM, silakan lihat Contoh kebijakan berbasis identitas untuk Amazon ElastiCache.

## Mengautentikasi dengan identitas

Autentikasi merupakan cara Anda untuk masuk ke AWS menggunakan kredensial identitas Anda. Anda harus terautentikasi (masuk ke AWS) sebagai Pengguna root akun AWS, sebagai pengguna IAM, atau dengan mengambil peran IAM.

Anda dapat masuk ke AWS sebagai identitas terfederasi dengan menggunakan kredensial yang disediakan melalui sumber identitas. Pengguna AWS IAM Identity Center (Pusat Identitas IAM), autentikasi masuk tunggal perusahaan Anda, dan kredensial Google atau Facebook Anda merupakan contoh identitas terfederasi. Saat Anda masuk sebagai identitas terfederasi, administrator Anda sebelumnya menyiapkan federasi identitas dengan menggunakan peran IAM. Ketika Anda mengakses AWS dengan menggunakan federasi, Anda secara tidak langsung mengambil suatu peran.

Tergantung pada jenis pengguna Anda, Anda dapat masuk ke AWS Management Console atau portal akses AWS. Untuk informasi selengkapnya tentang masuk ke AWS, lihat Cara masuk ke Akun AWS Anda dalam Panduan Pengguna AWS Sign-In.

Jika Anda mengakses AWS secara programatis, AWS menyediakan kit pengembangan perangkat lunak (SDK) dan antarmuka baris perintah (CLI) untuk menandatangani permintaan Anda secara kriptografis dengan menggunakan kredensial Anda. Jika Anda tidak menggunakan alat AWS, maka Anda harus menandatangani sendiri permintaan tersebut. Untuk informasi selengkapnya tentang penggunaan metode yang disarankan untuk menandatangani permintaan sendiri, lihat Menandatangani permintaan API AWS dalam Panduan Pengguna IAM.

Terlepas dari metode autentikasi yang Anda gunakan, Anda mungkin juga diminta untuk menyediakan informasi keamanan tambahan. Misalnya, AWS menyarankan agar Anda menggunakan autentikasi multi-faktor (MFA) untuk meningkatkan keamanan akun Anda. Untuk mempelajari selengkapnya, lihat Autentikasi multi-faktor dalam Panduan Pengguna AWS IAM Identity Center dan Menggunakan autentikasi multi-faktor (MFA) dalam AWS dalam Panduan Pengguna IAM.

## Pengguna root Akun AWS

Ketika Anda membuat Akun AWS, Anda memulai dengan satu identitas masuk yang memiliki akses ke semua Layanan AWS dan sumber daya di akun tersebut. Identitas ini disebut pengguna root Akun AWS dan diakses dengan cara masuk dengan alamat email dan kata sandi yang Anda gunakan untuk membuat akun. Kami sangat menyarankan Anda tidak menggunakan pengguna root untuk tugas sehari-hari. Lindungi kredensial pengguna root Anda dan gunakan kredensial tersebut untuk melakukan tugas yang hanya dapat dilakukan pengguna root. Untuk daftar lengkap tugas yang mengharuskan Anda masuk sebagai pengguna root, lihat Tugas yang memerlukan kredensial pengguna root dalam Panduan Pengguna IAM.

## Identitas terfederasi

Sebagai praktik terbaik, wajibkan pengguna manusia, termasuk pengguna yang memerlukan akses administrator, untuk menggunakan federasi dengan penyedia identitas untuk mengakses Layanan AWS dengan kredensial sementara.

Identitas terfederasi adalah pengguna dari direktori pengguna korporasi Anda, penyedia identitas web, AWS Directory Service, direktori Pusat Identitas, atau pengguna mana pun yang mengakses Layanan AWS dengan menggunakan kredensial yang disediakan melalui sumber identitas. Ketika identitas terfederasi mengakses Akun AWS, identitas tersebut mengambil peran, dan peran ini memberikan kredensial sementara.

Untuk pengelolaan akses terpusat, kami sarankan Anda menggunakan AWS IAM Identity Center. Anda dapat membuat pengguna dan grup di Pusat Identitas IAM, atau Anda dapat menghubungkan dan menyinkronkan ke sekumpulan pengguna dan grup di sumber identitas Anda sendiri untuk digunakan di semua Akun AWS Anda dan aplikasi Anda. Untuk informasi tentang Pusat Identitas IAM, lihat Apakah itu Pusat Identitas IAM? dalam Panduan Pengguna AWS IAM Identity Center.

## Pengguna dan Grup IAM

Pengguna IAM adalah identitas dalam Akun AWS Anda yang memiliki izin khusus untuk satu orang atau aplikasi. Jika memungkinkan, kami merekomendasikan untuk mengandalkan kredensial sementara alih-alih membuat pengguna IAM yang memiliki kredensial jangka panjang seperti kata sandi dan kunci akses. Namun, jika Anda memiliki kasus penggunaan tertentu yang memerlukan kredensial jangka panjang dengan pengguna IAM, kami merekomendasikan Anda merotasi kunci akses. Untuk informasi selengkapnya, lihat Merotasi kunci akses secara teratur untuk kasus penggunaan yang memerlukan kredensial jangka panjang dalam Panduan Pengguna IAM.

Grup IAM adalah identitas yang menentukan sekumpulan pengguna IAM. Anda tidak dapat masuk sebagai grup. Anda dapat menggunakan grup untuk menentukan izin bagi beberapa pengguna sekaligus. Grup mempermudah pengelolaan izin untuk sejumlah besar pengguna sekaligus. Misalnya, Anda dapat memiliki grup yang bernama IAMAdmins dan memberikan izin ke grup tersebut untuk mengelola sumber daya IAM.

Pengguna berbeda dari peran. Pengguna secara unik terkait dengan satu orang atau aplikasi, tetapi peran dimaksudkan untuk dapat digunakan oleh siapa pun yang membutuhkannya. Pengguna memiliki kredensial jangka panjang permanen, tetapi peran memberikan kredensial sementara. Untuk mempelajari selengkapnya, silakan lihat Kapan harus membuat pengguna IAM (bukan peran) dalam Panduan Pengguna IAM.

## Peran IAM

Peran IAM merupakan identitas dalam Akun AWS Anda yang memiliki izin khusus. Peran ini mirip dengan pengguna IAM, tetapi tidak terkait dengan orang tertentu. Anda dapat mengambil peran IAM untuk sementara di AWS Management Console dengan beralih peran. Anda dapat mengambil peran dengan cara memanggil operasi API AWS CLI atau AWS atau menggunakan URL kustom. Untuk informasi selengkapnya tentang cara menggunakan peran, lihat Menggunakan peran IAM dalam Panduan Pengguna IAM.

Peran IAM dengan kredensial sementara berguna dalam situasi berikut:

- Akses pengguna terfederasi – Untuk menetapkan izin ke identitas terfederasi, Anda membuat peran dan menentukan izin untuk peran tersebut. Ketika identitas terfederasi mengautentikasi, identitas tersebut terhubung dengan peran dan diberi izin yang ditentukan oleh peran. Untuk informasi tentang peran-peran untuk federasi, lihat Membuat peran untuk Penyedia Identitas pihak ketiga dalam Panduan Pengguna IAM. Jika menggunakan Pusat Identitas IAM, Anda harus mengonfigurasi set izin. Untuk mengontrol apa yang dapat diakses identitas Anda setelah identitas tersebut diautentikasi, Pusat Identitas IAM akan mengorelasikan set izin ke peran dalam IAM. Untuk informasi tentang set izin, lihat Set izin dalam Panduan Pengguna AWS IAM Identity Center.
- Izin pengguna IAM sementara – Pengguna atau peran IAM dapat mengambil peran IAM guna mendapatkan berbagai izin secara sementara untuk tugas tertentu.
- Akses lintas akun – Anda dapat menggunakan peran IAM untuk mengizinkan seseorang (prinsipal tepercaya) di akun lain untuk mengakses sumber daya di akun Anda. Peran adalah cara utama untuk memberikan akses lintas akun. Namun, pada beberapa Layanan AWS, Anda dapat melampirkan kebijakan secara langsung ke sumber daya (alih-alih menggunakan suatu peran sebagai perantara). Untuk mempelajari perbedaan antara kebijakan berbasis peran dan berbasis

sumber daya untuk akses lintas akun, lihat Bagaimana peran IAM berbeda dari kebijakan berbasis sumber daya dalam Panduan Pengguna IAM.

- Akses lintas layanan – Sebagian Layanan AWS menggunakan fitur di Layanan AWS lainnya. Sebagai contoh, ketika Anda memanggil suatu layanan, biasanya layanan tersebut menjalankan aplikasi di Amazon EC2 atau menyimpan objek di Amazon S3. Sebuah layanan mungkin melakukannya menggunakan izin prinsipal yang memanggil, menggunakan peran layanan, atau peran terkait layanan.
  - Sesi akses maju (FAS) – Ketika Anda menggunakan pengguna IAM atau peran IAM untuk melakukan tindakan-tindakan di AWS, Anda akan dianggap sebagai prinsipal. Ketika Anda menggunakan beberapa layanan, Anda mungkin melakukan sebuah tindakan yang kemudian menginisiasi tindakan lain di layanan yang berbeda. FAS menggunakan izin dari prinsipal yang memanggil Layanan AWS, yang dikombinasikan dengan Layanan AWS, untuk mengajukan permintaan ke layanan hilir. Permintaan FAS hanya diajukan ketika sebuah layanan menerima permintaan yang memerlukan interaksi dengan Layanan AWS lain atau sumber daya lain untuk diselesaikan. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan ketika mengajukan permintaan FAS, lihat Sesi akses maju.
  - Peran layanan – Sebuah peran layanan adalah sebuah peran IAM yang dijalankan oleh suatu layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, memodifikasi, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat Membuat sebuah peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan pengguna IAM.
  - Peran terkait layanan – Peran terkait layanan adalah tipe peran layanan yang tertaut dengan Layanan AWS. Layanan tersebut dapat menjalankan peran untuk melakukan sebuah tindakan atas nama Anda. Peran terkait layanan akan muncul di Akun AWS Anda dan dimiliki oleh layanan tersebut. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.
- Aplikasi yang berjalan di Amazon EC2 – Anda dapat menggunakan peran IAM untuk mengelola kredensial sementara untuk aplikasi yang berjalan di instans EC2 dan mengajukan permintaan API AWS CLI atau AWS. Cara ini lebih baik daripada menyimpan kunci akses dalam instans EC2. Untuk memberikan peran AWS ke instans EC2 dan menyediakannya di semua aplikasinya, Anda dapat membuat profil instans yang dilampirkan ke instans tersebut. Profil instans berisi peran dan memungkinkan program yang berjalan di instans EC2 mendapatkan kredensial sementara. Untuk informasi selengkapnya, lihat Menggunakan peran IAM untuk memberikan izin ke aplikasi yang berjalan dalam instans Amazon EC2 dalam Panduan Pengguna IAM.

Untuk mempelajari apakah kita harus menggunakan peran IAM atau pengguna IAM, lihat Kapan harus membuat peran IAM (bukan pengguna) dalam Panduan Pengguna IAM.

## Mengelola kebijakan menggunakan akses

Anda mengendalikan akses di AWS dengan membuat kebijakan dan melampirkannya ke identitas atau sumber daya AWS. Kebijakan adalah objek di AWS yang, ketika terkait dengan identitas atau sumber daya, akan menentukan izinnya. AWS mengevaluasi kebijakan-kebijakan tersebut ketika prinsipal (pengguna, pengguna root, atau sesi peran) mengajukan permintaan. Izin dalam kebijakan menentukan apakah permintaan diizinkan atau ditolak. Sebagian besar kebijakan disimpan di AWS sebagai dokumen JSON. Untuk informasi selengkapnya tentang struktur dan isi dokumen kebijakan JSON, lihat Gambaran umum kebijakan JSON dalam Panduan Pengguna IAM.

Administrator dapat menggunakan kebijakan JSON AWS untuk menentukan secara spesifik siapa yang memiliki akses pada apa. Artinya, prinsipal manakah yang dapat melakukan tindakan pada sumber daya apa, dan dengan kondisi apa.

Secara default, pengguna dan peran tidak memiliki izin. Untuk mengabulkan izin kepada pengguna untuk melakukan tindakan di sumber daya yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM. Administrator kemudian dapat menambahkan kebijakan IAM ke peran, dan pengguna dapat menjalankan peran.

Kebijakan IAM mendefinisikan izin untuk suatu tindakan terlepas dari metode yang Anda gunakan untuk melakukan operasinya. Misalnya, anggaplah Anda memiliki kebijakan yang mengizinkan tindakan `iam:GetRole`. Pengguna dengan kebijakan tersebut dapat memperoleh informasi peran dari AWS Management Console, AWS CLI, atau API AWS.

### Kebijakan berbasis identitas

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke sebuah identitas, misalnya pengguna IAM, grup pengguna IAM, atau peran IAM. Kebijakan ini mengontrol tindakan apa yang dapat dilakukan oleh pengguna dan peran, pada sumber daya mana, dan dalam kondisi apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Membuat kebijakan IAM dalam Panduan Pengguna IAM.

Kebijakan berbasis identitas dapat dikategorikan lebih lanjut sebagai kebijakan inline atau kebijakan terkelola. Kebijakan inline disematkan langsung ke satu pengguna, grup, atau peran. Kebijakan terkelola adalah kebijakan mandiri yang dapat Anda lampirkan ke beberapa pengguna, grup, dan peran di Akun AWS Anda. Kebijakan terkelola meliputi kebijakan yang dikelola AWS dan kebijakan

yang dikelola pelanggan. Untuk mempelajari cara memilih antara kebijakan terkelola atau kebijakan inline, lihat Memilih antara kebijakan terkelola dan kebijakan inline dalam Panduan Pengguna IAM.

### Kebijakan berbasis sumber daya

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contoh kebijakan-kebijakan berbasis sumber daya adalah kebijakan tepercaya peran IAM dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Untuk sumber daya tempat kebijakan dilampirkan, kebijakan menentukan tindakan apa yang dapat dilakukan oleh prinsipal tertentu pada sumber daya tersebut dan dalam kondisi apa. Anda harus menentukan principal dalam kebijakan berbasis sumber daya. Prinsipal dapat mencakup akun, pengguna, peran, pengguna terfederasi, atau Layanan AWS.

Kebijakan berbasis sumber daya merupakan kebijakan inline yang terletak di layanan tersebut. Anda tidak dapat menggunakan kebijakan yang dikelola AWS dari IAM dalam kebijakan berbasis sumber daya.

### Daftar kontrol akses (ACL)

Daftar kontrol akses (ACL) mengendalikan prinsipal mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACL serupa dengan kebijakan berbasis sumber daya, meskipun kebijakan tersebut tidak menggunakan format dokumen kebijakan JSON.

Amazon S3, AWS WAF, dan Amazon VPC adalah contoh-contoh layanan yang mendukung ACL. Untuk mempelajari ACL selengkapnya, lihat Gambaran umum daftar kontrol akses (ACL) dalam Panduan Developer Amazon Simple Storage Service.

### Tipe-tipe kebijakan lain

AWS mendukung tipe kebijakan tambahan, yang kurang umum. Tipe-tipe kebijakan ini dapat mengatur izin maksimum yang diberikan kepada Anda oleh tipe kebijakan yang lebih umum.

- Batasan izin – Batasan izin adalah fitur lanjutan tempat Anda mengatur izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas ke entitas IAM (pengguna IAM atau peran IAM). Anda dapat menetapkan batasan izin untuk suatu entitas. Izin yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas milik entitas dan batas izinnya. Kebijakan berbasis sumber daya yang menentukan pengguna atau peran dalam bidang `Principal` tidak dibatasi oleh batasan izin. Penolakan eksplisit dalam salah satu kebijakan ini akan menggantikan pemberian izin. Untuk

informasi selengkapnya tentang batasan izin, lihat Batasan izin untuk entitas IAM dalam Panduan Pengguna IAM.

- Kebijakan kontrol layanan (SCP) – SCP adalah kebijakan JSON yang menentukan izin maksimum untuk sebuah organisasi atau unit organisasi (OU) di AWS Organizations. AWS Organizations adalah layanan untuk mengelompokkan dan secara terpusat mengelola beberapa Akun AWS yang dimiliki bisnis Anda. Jika Anda mengaktifkan semua fitur di suatu organisasi, maka Anda dapat menerapkan kebijakan kontrol layanan (SCP) ke salah satu atau ke semua akun Anda. SCP membatasi izin untuk entitas dalam akun anggota, termasuk setiap Pengguna root akun AWS. Untuk informasi selengkapnya tentang Organisasi dan SCP, lihat Cara kerja SCP dalam Panduan Pengguna AWS Organizations.
- Kebijakan sesi – Kebijakan sesi adalah kebijakan lanjutan yang Anda berikan sebagai parameter ketika Anda membuat sesi sementara secara programatis untuk peran atau pengguna terfederasi. Izin sesi yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas pengguna atau peran dan kebijakan sesi. Izin juga dapat berasal dari kebijakan berbasis sumber daya. Penolakan eksplisit dalam salah satu kebijakan ini menindihi izin. Untuk informasi selengkapnya, lihat Kebijakan sesi dalam Panduan Pengguna IAM.

### Berbagai tipe kebijakan

Ketika beberapa tipe kebijakan berlaku pada suatu permintaan, izin yang dihasilkan lebih rumit untuk dipahami. Untuk mempelajari cara AWS menentukan apakah akan mengizinkan permintaan ketika beberapa tipe kebijakan digunakan, lihat Logika evaluasi kebijakan dalam Panduan Pengguna IAM.

## Cara Amazon ElastiCache bekerja dengan IAM

Sebelum Anda menggunakan IAM untuk mengelola akses ke ElastiCache, pelajari fitur IAM apakah yang tersedia untuk digunakan pada ElastiCache.

Fitur-fitur IAM yang dapat Anda gunakan dengan Amazon ElastiCache

| Fitur IAM | Dukungan ElastiCache |
|---|---|
| Kebijakan berbasis identitas | Ya |
| Kebijakan berbasis sumber daya | Tidak |
| Tindakan kebijakan | Ya |

| Fitur IAM | Dukungan ElastiCache |
| --- | --- |
| Sumber daya kebijakan | Ya |
| Kunci-kunci persyaratan kebijakan | Ya |
| ACL | Ya |
| ABAC (tag dalam kebijakan) | Ya |
| Kredensial sementara | Ya |
| Izin pengguna utama | Ya |
| Peran layanan | Ya |
| Peran tertaut layanan | Ya |

Untuk mendapatkan tampilan tingkat tinggi tentang cara ElastiCache dan layanan AWS lain bekerja dengan sebagian besar fitur IAM, silakan lihat Layanan AWS yang bekerja dengan IAM di Panduan Pengguna IAM.

## Kebijakan berbasis identitas untuk ElastiCache

| | |
| --- | --- |
| Mendukung kebijakan berbasis identitas | Ya |

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke sebuah identitas, misalnya pengguna IAM, grup pengguna IAM, atau peran IAM. Kebijakan ini mengontrol tindakan apa yang dapat dilakukan oleh pengguna dan peran, pada sumber daya mana, dan dalam kondisi apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Membuat kebijakan IAM dalam Panduan Pengguna IAM.

Dengan kebijakan berbasis identitas IAM, Anda dapat menentukan secara spesifik apakah tindakan dan sumber daya diizinkan atau ditolak, serta persyaratan yang menjadi dasar dikabulkan atau ditolaknya tindakan tersebut. Anda tidak dapat menentukan secara spesifik pengguna utama dalam sebuah kebijakan berbasis identitas karena pengguna utama berlaku bagi pengguna atau peran yang melekat kepadanya. Untuk mempelajari semua elemen yang dapat Anda gunakan dalam kebijakan JSON, lihat Referensi elemen kebijakan JSON IAM dalam Panduan Pengguna IAM.

Contoh kebijakan berbasis identitas untuk ElastiCache

Untuk melihat contoh kebijakan-kebijakan berbasis identitas ElastiCache, silakan lihat Contoh kebijakan berbasis identitas untuk Amazon ElastiCache.

## Kebijakan berbasis sumber daya dalam ElastiCache

| | |
|---|---|
| Mendukung kebijakan berbasis sumber daya | Tidak |

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contoh kebijakan-kebijakan berbasis sumber daya adalah kebijakan tepercaya peran IAM dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Untuk sumber daya tempat kebijakan dilampirkan, kebijakan menentukan tindakan apa yang dapat dilakukan oleh prinsipal tertentu pada sumber daya tersebut dan dalam kondisi apa. Anda harus menentukan principal dalam kebijakan berbasis sumber daya. Pengguna utama dapat mencakup akun, pengguna, peran, pengguna gabungan, atau Layanan AWS.

Untuk mengaktifkan akses lintas akun, Anda dapat menentukan secara spesifik seluruh akun atau entitas IAM di akun lain sebagai pengguna utama dalam kebijakan berbasis sumber daya. Menambahkan pengguna utama akun silang ke kebijakan berbasis sumber daya hanya setengah dari membangun hubungan kepercayaan. Ketika pengguna utama dan sumber daya berada dalam Akun AWS yang berbeda, Administrator IAM di akun tepercaya juga harus memberikan izin kepada entitas pengguna utama (pengguna atau peran) untuk mengakses sumber daya. Mereka memberikan izin melampirkan kebijakan berbasis identitas kepada entitas. Namun, jika kebijakan berbasis sumber daya memberikan akses kepada pengguna utama dalam akun yang sama, tidak diperlukan kebijakan berbasis identitas tambahan. Untuk informasi selengkapnya, lihat Perbedaan antara peran IAM dan kebijakan berbasis sumber daya di Panduan Pengguna IAM.

## Tindakan kebijakan untuk ElastiCache

| | |
|---|---|
| Mendukung tindakan kebijakan | Ya |

Administrator dapat menggunakan kebijakan JSON AWS untuk menentukan secara spesifik siapa yang memiliki akses pada apa. Yaitu, pengguna utama manakah yang dapat melakukan tindakan pada sumber daya apa, dan dengan syarat apa.

Elemen `Action` dari kebijakan JSON menjelaskan tindakan-tindakan yang dapat Anda gunakan untuk mengizinkan atau menolak akses dalam sebuah kebijakan. Tindakan kebijakan biasanya memiliki nama yang sama sebagaimana operasi API AWS yang dikaitkan padanya. Ada beberapa pengecualian, misalnya tindakan hanya izin yang tidak memiliki operasi API yang cocok. Ada juga beberapa operasi yang memerlukan beberapa tindakan dalam sebuah kebijakan. Tindakan-tindakan tambahan ini disebut tindakan dependen.

Menyertakan tindakan dalam suatu kebijakan untuk memberikan izin guna melakukan operasi yang terkait.

Untuk melihat daftar tindakan ElastiCache, silakan lihat Tindakan yang ditentukan oleh Amazon ElastiCache di Rujukan Otorisasi Layanan.

Tindakan kebijakan di ElastiCache menggunakan awalan berikut sebelum tindakan:

```
elasticache
```

Untuk menetapkan secara spesifik beberapa tindakan dalam satu pernyataan, pisahkan tindakan-tindakan tersebut dengan koma.

```
"Action": [
      "elasticache:action1",
      "elasticache:action2"
         ]
```

Anda juga dapat menentukan beberapa tindakan menggunakan wildcard (*). Sebagai contoh, untuk menentukan semua tindakan yang dimulai dengan kata `Describe`, sertakan tindakan berikut:

```
"Action": "elasticache:Describe*"
```

Untuk melihat contoh kebijakan-kebijakan berbasis identitas ElastiCache, silakan lihat Contoh kebijakan berbasis identitas untuk Amazon ElastiCache.

## Sumber daya kebijakan untuk ElastiCache

| | |
|---|---|
| Mendukung sumber daya kebijakan | Ya |

Administrator dapat menggunakan kebijakan JSON AWS untuk menentukan secara spesifik siapa yang memiliki akses pada apa. Yaitu, pengguna utama manakah yang dapat melakukan tindakan pada sumber daya apa, dan dengan syarat apa.

Elemen kebijakan JSON `Resource` menentukan objek atau objek-objek yang menjadi target penerapan tindakan. Pernyataan harus menyertakan elemen `Resource` atau `NotResource`. Praktik terbaiknya, tentukan sumber daya menggunakan Amazon Resource Name (ARN). Anda dapat melakukan ini untuk tindakan-tindakan yang mendukung jenis sumber daya tertentu, yang dikenal sebagai izin tingkat sumber daya.

Untuk tindakan yang tidak mendukung izin di tingkat sumber daya, misalnya operasi pencantuman, gunakan wildcard (*) untuk mengindikasikan bahwa pernyataan tersebut berlaku bagi semua sumber daya.

```
"Resource": "*"
```

Untuk melihat daftar jenis sumber daya ElastiCache dan ARN-nya, lihat Sumber Daya yang Ditentukan oleh Amazon ElastiCache di Referensi Otorisasi Layanan. Untuk mempelajari jenis tindakan yang dapat Anda tentukan dengan ARN di tiap sumber daya, lihat Tindakan yang Ditentukan oleh Amazon ElastiCache.

Untuk melihat contoh kebijakan-kebijakan berbasis identitas ElastiCache, silakan lihat Contoh kebijakan berbasis identitas untuk Amazon ElastiCache.

## Kunci syarat kebijakan untuk ElastiCache

| | |
|---|---|
| Mendukung kunci kondisi kebijakan khusus layanan | Ya |

Administrator dapat menggunakan kebijakan JSON AWS untuk menentukan secara spesifik siapa yang memiliki akses pada apa. Yaitu, pengguna utama manakah yang dapat melakukan tindakan pada sumber daya apa, dan dengan syarat apa.

Elemen `Condition` (atau blok `Condition`) akan memungkinkan Anda menentukan syarat yang menjadi dasar suatu pernyataan berlaku. Elemen `Condition` bersifat opsional. Anda dapat membuat ekspresi bersyarat yang menggunakan operator syarat, misalnya sama dengan atau kurang dari, untuk mencocokkan syarat dalam kebijakan dengan nilai-nilai yang diminta.

Jika Anda menentukan beberapa elemen `Condition` dalam sebuah pernyataan, atau beberapa kunci dalam elemen `Condition` tunggal, maka AWS akan mengevaluasinya dengan menggunakan operasi `AND` logis. Jika Anda menentukan beberapa nilai untuk satu kunci kondisi, AWS akan mengevaluasi syarat tersebut menggunakan operasi `OR` logis. Semua persyaratan harus dipenuhi sebelum izin pernyataan diberikan.

Anda juga dapat menggunakan variabel placeholder saat menentukan syarat. Sebagai contoh, Anda dapat memberikan izin kepada pengguna IAM untuk mengakses sumber daya hanya jika izin tersebut mempunyai tag yang sesuai dengan nama pengguna IAM terkait. Untuk informasi selengkapnya, lihat Elemen kebijakan IAM: variabel dan tag di Panduan Pengguna IAM.

AWS mendukung kunci kondisi global dan kunci kondisi spesifik layanan. Untuk melihat semua kunci kondisi global AWS, lihat kunci konteks syarat global AWS di Panduan Pengguna IAM.

Untuk melihat daftar kunci-kunci persyaratan ElastiCache, silakan lihat kunci persyaratan untuk Amazon ElastiCache di Rujukan Otorisasi Layanan. Untuk mempelajari tindakan dan sumber daya yang dapat Anda gunakan dengan kunci syarat, lihat Tindakan yang Ditentukan oleh Amazon ElastiCache.

Untuk melihat contoh kebijakan-kebijakan berbasis identitas ElastiCache, silakan lihat Contoh kebijakan berbasis identitas untuk Amazon ElastiCache.

## Daftar kontrol akses (ACL) di ElastiCache

| | |
|---|---|
| Mendukung ACL | Ya |

Daftar kontrol akses (ACL) mengendalikan prinsipal mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACL serupa dengan kebijakan berbasis sumber daya, meskipun kebijakan tersebut tidak menggunakan format dokumen kebijakan JSON.

## Kontrol akses berbasis atribut (ABAC) dengan ElastiCache

| | |
|---|---|
| Mendukung ABAC (tag dalam kebijakan) | Ya |

Kontrol akses berbasis atribut (ABAC) adalah strategi otorisasi yang menentukan izin berdasarkan atribut. Di AWS, atribut-atribut ini disebut tag. Anda dapat melampirkan tag ke entitas IAM (pengguna atau peran) dan ke banyak sumber daya AWS. Pemberian tag ke entitas dan sumber daya adalah

langkah pertama dari ABAC. Kemudian rancanglah kebijakan ABAC untuk mengizinkan operasi ketika tag milik pengguna utama cocok dengan tag yang ada di sumber daya yang ingin diakses.

ABAC sangat berguna di lingkungan yang berkembang dengan cepat dan berguna di situasi dimana pengelolaan kebijakan menjadi rumit.

Untuk mengendalikan akses berdasarkan tag, berikan informasi tentang tag di elemen syarat dari kebijakan dengan menggunakan kunci kondisi `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, atau `aws:TagKeys`.

Jika sebuah layanan mendukung ketiga kunci kondisi untuk setiap jenis sumber daya, nilainya adalah Ya untuk layanan tersebut. Jika suatu layanan mendukung ketiga kunci kondisi untuk hanya beberapa jenis sumber daya, nilainya adalah Parsial.

Untuk informasi selengkapnya tentang ABAC, lihat Apa itu ABAC? di Panduan Pengguna IAM. Untuk melihat tutorial yang menguraikan langkah-langkah pengaturan ABAC, lihat Menggunakan kontrol akses berbasis atribut (ABAC) di Panduan Pengguna IAM.

## Menggunakan kredensial Temporer dengan ElastiCache

| | |
|---|---|
| Mendukung penggunaan kredensial sementara | Ya |

Beberapa Layanan AWS tidak berfungsi saat Anda masuk menggunakan kredensial sementara. Sebagai informasi tambahan, termasuk tentang Layanan AWS mana saja yang berfungsi dengan kredensial sementara, lihat Layanan AWS yang berfungsi dengan IAM di Panduan Pengguna IAM.

Anda menggunakan kredensial sementara jika Anda masuk ke AWS Management Console menggunakan metode apa pun kecuali nama pengguna dan kata sandi. Sebagai contoh, ketika Anda mengakses AWS dengan menggunakan tautan masuk tunggal (SSO) milik perusahaan Anda, proses itu secara otomatis akan membuat kredensial sementara. Anda juga akan secara otomatis membuat kredensial sementara ketika Anda masuk ke konsol sebagai seorang pengguna dan kemudian beralih peran. Untuk informasi selengkapnya tentang peralihan peran, lihat Peralihan peran (konsol) di Panduan Pengguna IAM.

Anda dapat membuat kredensial sementara secara manual menggunakan API AWS CLI atau AWS. Anda kemudian dapat menggunakan kredensial sementara tersebut untuk mengakses AWS. AWS menyarankan agar Anda membuat kredensial sementara secara dinamis alih-alih menggunakan kunci akses jangka panjang. Untuk informasi selengkapnya, silakan lihat Kredensial keamanan sementara di IAM.

## Izin pengguna utama lintas layanan untuk ElastiCache

| | |
|---|---|
| Mendukung sesi akses maju (FAS) | Ya |

Saat Anda menggunakan pengguna IAM atau peran IAM untuk melakukan tindakan di AWS, Anda akan dianggap sebagai pengguna utama. Ketika Anda menggunakan beberapa layanan, Anda mungkin melakukan sebuah tindakan yang kemudian dilanjutkan oleh tindakan lain pada layanan yang berbeda. FAS menggunakan izin dari prinsipal yang memanggil Layanan AWS, yang dikombinasikan dengan Layanan AWS, untuk mengajukan permintaan ke layanan hilir. Permintaan FAS hanya diajukan ketika sebuah layanan menerima permintaan yang memerlukan interaksi dengan Layanan AWS lain atau sumber daya lain untuk diselesaikan. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan ketika mengajukan permintaan FAS, lihat Meneruskan sesi akses.

## Peran layanan untuk ElastiCache

| | |
|---|---|
| Mendukung peran layanan | Ya |

Peran layanan adalah sebuah peran IAM yang diambil oleh sebuah layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, memodifikasi, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat Membuat sebuah peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan pengguna IAM.

> **Warning**
>
> Mengubah izin untuk sebuah peran layanan dapat merusak fungsionalitas ElastiCache. Melakukan edit peran layanan hanya jika ElastiCache memberikan panduan untuk melakukannya.

## Peran tertaut layanan untuk ElastiCache

| | |
|---|---|
| Mendukung peran yang tertaut layanan | Ya |

Peran yang tertaut layanan adalah jenis peran layanan yang tertaut dengan Layanan AWS. Layanan tersebut dapat menjalankan peran untuk melakukan sebuah tindakan atas nama Anda. Peran terkait layanan akan muncul di Akun AWS Anda dan dimiliki oleh layanan tersebut. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.

Untuk detail tentang pembuatan atau pengelolaan peran tertaut layanan, silakan lihat Layanan AWS yang berfungsi dengan IAM. Cari layanan dalam tabel yang memiliki `Yes` di kolom Peran tertaut layanan. Pilih tautan Ya untuk melihat dokumentasi peran tertaut layanan untuk layanan tersebut.

## Contoh kebijakan berbasis identitas untuk Amazon ElastiCache

Secara bawaan, pengguna dan peran tidak memiliki izin untuk membuat atau melakukan modifikasi atas sumber daya ElastiCache. Pengguna dan peran tersebut juga tidak dapat melakukan tugas dengan menggunakan API AWS Management Console, AWS Command Line Interface (AWS CLI), atau AWS. Untuk memberikan izin kepada pengguna untuk melakukan tindakan di sumber daya yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM. Administrator kemudian dapat menambahkan kebijakan IAM ke peran, dan pengguna dapat mengambil peran.

Untuk mempelajari cara membuat kebijakan berbasis identitas IAM menggunakan contoh dokumen kebijakan JSON ini, lihat Membuat kebijakan IAM di Panduan Pengguna IAM.

Untuk mengetahui hal detail tentang tindakan dan jenis sumber daya yang ditentukan oleh ElastiCache, termasuk format ARN untuk setiap jenis sumber daya, silakan lihat Tindakan, Sumber Daya, dan kunci kondisi untuk Amazon ElastiCache di Rujukan Otorisasi Layanan.

Topik

- Praktik terbaik kebijakan
- Menggunakan Konsol ElastiCache
- Izinkan pengguna melihat izin mereka sendiri

### Praktik terbaik kebijakan

Kebijakan-kebijakan berbasis identitas menentukan apakah seseorang dapat membuat, mengakses, atau menghapus sumber daya ElastiCache yang ada di akun Anda. Tindakan ini mengenakan biaya kepada Akun AWS Anda. Ketika Anda membuat atau mengedit kebijakan berbasis identitas, ikuti panduan dan rekomendasi ini:

- Memulai kebijakan terkelola AWS dan beralih ke izin dengan hak akses paling rendah – Untuk mulai memberikan izin kepada pengguna dan beban kerja Anda, gunakan kebijakan terkelola AWS yang memberikan izin untuk banyak kasus penggunaan umum. Kebijakan tersedia di Akun AWS Anda. Sebaiknya kurangi izin lebih lanjut dengan menentukan kebijakan yang dikelola pelanggan AWS yang bersifat khusus untuk kasus penggunaan Anda. Untuk informasi selengkapnya, lihat Kebijakan terkelola AWS atau Kebijakan terkelola AWS untuk fungsi tugas di Panduan Pengguna IAM.
- Menerapkan izin dengan hak akses paling rendah – Ketika Anda menetapkan izin dengan kebijakan IAM, hanya berikan izin yang diperlukan untuk melakukan tugas. Anda melakukannya dengan mendefinisikan tindakan yang dapat diambil pada sumber daya tertentu dalam kondisi tertentu, yang juga dikenal sebagai izin dengan hak akses paling rendah. Untuk informasi selengkapnya tentang cara menggunakan IAM untuk mengajukan izin, lihat Kebijakan dan izin di IAM di Panduan Pengguna IAM.
- Gunakan syarat dalam kebijakan IAM untuk membatasi akses lebih lanjut – Anda dapat menambahkan suatu syarat ke kebijakan Anda untuk membatasi akses ke tindakan dan sumber daya. Sebagai contoh, Anda dapat menulis syarat kebijakan untuk menentukan bahwa semua permintaan harus dikirim menggunakan SSL. Anda juga dapat menggunakan syarat untuk memberi akses ke tindakan layanan jika digunakan melalui Layanan AWS tertentu, seperti AWS CloudFormation. Untuk informasi selengkapnya, lihat Elemen kebijakan JSON IAM: Syarat di Panduan Pengguna IAM.
- Gunakan IAM Access Analyzer untuk memvalidasi kebijakan IAM Anda untuk memastikan izin yang aman dan fungsional – IAM Access Analyzer memvalidasi kebijakan baru dan yang sudah ada sehingga kebijakan tersebut mematuhi bahasa kebijakan IAM (JSON) dan praktik terbaik IAM. IAM Access Analyzer menyediakan lebih dari 100 pemeriksaan kebijakan dan rekomendasi yang dapat ditindaklanjuti untuk membantu Anda membuat kebijakan yang aman dan fungsional. Untuk informasi selengkapnya, lihat Validasi kebijakan IAM Access Analyzer di Panduan Pengguna IAM.
- Memerlukan autentikasi multi-faktor (MFA) – Jika Anda memiliki skenario yang mengharuskan pengguna IAM atau pengguna root di Akun AWS Anda, aktifkan MFA untuk keamanan tambahan. Untuk meminta MFA ketika operasi API dipanggil, tambahkan syarat MFA pada kebijakan Anda. Untuk informasi selengkapnya, silakan lihat Mengonfigurasi akses API yang dilindungi MFA di Panduan Pengguna IAM.

Untuk informasi selengkapnya tentang praktik terbaik di IAM, lihat Praktik terbaik keamanan di IAM di Panduan Pengguna IAM.

## Menggunakan Konsol ElastiCache

Untuk mengakses konsol Amazon ElastiCache, Anda harus memiliki serangkaian izin minimum. Izin tersebut harus memperbolehkan Anda untuk membuat daftar dan melihat detail tentang sumber daya ElastiCache di akun Akun AWS Anda. Jika Anda membuat kebijakan berbasis identitas yang lebih ketat daripada izin minimum yang diperlukan, konsol tidak akan berfungsi sebagaimana mestinya untuk entitas (pengguna atau peran) dengan kebijakan tersebut.

Anda tidak perlu meloloskan izin konsol minimum bagi pengguna yang hanya melakukan panggilan ke API AWS CLI atau AWS. Sebagai gantinya, izinkan akses hanya ke tindakan yang sesuai dengan operasi API yang coba mereka lakukan.

Untuk memastikan bahwa pengguna dan peran masih dapat menggunakan konsol ElastiCache, lampirkan juga ElastiCache `ConsoleAccess` atau kebijakan terkelola `ReadOnly` AWS ke entitas. Untuk informasi selengkapnya, lihat Menambah izin untuk pengguna di Panduan Pengguna IAM.

### Izinkan pengguna melihat izin mereka sendiri

Contoh ini menunjukkan cara membuat kebijakan yang mengizinkan para pengguna IAM melihat kebijakan inline dan terkelola yang dilampirkan ke identitas pengguna mereka. Kebijakan ini mencakup izin untuk menyelesaikan tindakan pada konsol atau menggunakan API AWS CLI atau AWS secara terprogram.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
```

```
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

## Pemecahan masalah identitas dan akses Amazon ElastiCache

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah umum yang mungkin Anda hadapi ketika bekerja dengan ElastiCache dan IAM.

Topik

- Saya tidak berwenang untuk melakukan tindakan di ElastiCache
- Saya tidak memiliki otorisasi untuk melakukan iam:PeranMasuk
- Saya ingin mengizinkan orang di luar akun AWS saya untuk mengakses sumber daya ElastiCache

### Saya tidak berwenang untuk melakukan tindakan di ElastiCache

Jika AWS Management Console memberi tahu bahwa Anda tidak diberi otorisasi untuk melakukan tindakan, Anda harus menghubungi administrator untuk mendapatkan bantuan. Administrator Anda adalah orang yang memberikan nama pengguna dan kata sandi Anda.

Contoh kesalahan berikut terjadi ketika pengguna IAM `mateojackson` mencoba menggunakan konsol untuk melihat detail tentang suatu sumber daya *`my-example-widget`* rekaan, tetapi tidak memiliki izin `elasticache:`*`GetWidget`* rekaan.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 elasticache:GetWidget on resource: my-example-widget
```

Dalam hal ini, Mateo meminta administratornya untuk memperbarui kebijakannya untuk mengizinkan dia mengakses sumber daya *my-example-widget* menggunakan tindakan elasticache:*GetWidget*.

## Saya tidak memiliki otorisasi untuk melakukan iam:PeranMasuk

Jika Anda menerima pesan kesalahan bahwa Anda tidak memiliki otorisasi untuk melakukan tindakan iam:PassRole, kebijakan Anda harus diperbarui agar Anda memiliki izin untuk meneruskan peran ke ElastiCache.

Sebagian Layanan AWS mengizinkan Anda untuk memberikan peran yang sudah ada ke layanan tersebut alih-alih membuat peran layanan baru atau peran tertaut-layanan. Untuk melakukan tindakan tersebut, Anda harus memiliki izin untuk memberikan peran pada layanan tersebut.

Contoh kesalahan berikut terjadi ketika pengguna IAM bernama marymajor mencoba menggunakan konsol tersebut untuk melakukan tindakan di ElastiCache. Namun, tindakan tersebut memerlukan layanan untuk mendapatkan izin yang diberikan oleh peran layanan. Mary tidak memiliki izin untuk meneruskan peran tersebut pada layanan.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Dalam kasus ini, kebijakan Mary harus diperbarui agar dia mendapatkan izin untuk melakukan tindakan iam:PassRole tersebut.

Jika Anda membutuhkan bantuan, hubungi administrator AWS Anda. Administrator Anda adalah orang yang memberikan kredensial masuk Anda.

## Saya ingin mengizinkan orang di luar akun AWS saya untuk mengakses sumber daya ElastiCache

Anda dapat membuat peran yang dapat digunakan para pengguna di akun lain atau orang-orang di luar organisasi Anda untuk mengakses sumber daya Anda. Anda dapat menentukan siapa yang dipercaya untuk mengambil peran tersebut. Untuk layanan yang mendukung kebijakan berbasis sumber daya atau daftar kontrol akses (ACL), Anda dapat menggunakan kebijakan tersebut untuk memberi akses kepada orang ke sumber daya Anda.

Untuk mempelajari selengkapnya, periksa hal berikut:

- Untuk mempelajari apakah ElastiCache mendukung fitur-fitur ini, silakan lihat Cara Amazon ElastiCache bekerja dengan IAM.
- Untuk mempelajari cara memberikan akses ke sumber daya di seluruh Akun AWS yang Anda miliki, silakan lihat Menyediakan akses ke pengguna IAM di Akun AWS lainnya yang Anda miliki di Panduan Pengguna IAM.
- Untuk mempelajari cara memberikan akses ke sumber daya Anda ke pihak ketiga Akun AWS, silakan lihat Menyediakan akses ke akun Akun AWS yang dimiliki oleh pihak ketiga dalam Panduan Pengguna IAM.
- Untuk mempelajari cara memberikan akses melalui federasi identitas, silakan lihat Menyediakan akses ke pengguna terautentikasi eksternal (federasi identitas) di Panduan Pengguna IAM.
- Untuk mempelajari perbedaan antara penggunaan kebijakan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, silakan lihat Bagaimana peran IAM berbeda dari kebijakan berbasis sumber daya di Panduan Pengguna IAM.

## Kontrol akses

Anda dapat memiliki kredensial yang valid untuk mengautentikasi permintaan, tetapi kecuali jika Anda memiliki izin, Anda tidak dapat membuat atau mengakses sumber daya ElastiCache. Misalnya, Anda harus memiliki izin untuk membuat klaster ElastiCache.

Bagian berikut menjelaskan cara mengelola izin untuk ElastiCache. Anda disarankan untuk membaca gambaran umum terlebih dahulu.

- Gambaran umum pengelolaan izin akses untuk sumber daya ElastiCache Anda
- Menggunakan kebijakan berbasis identitas (kebijakan IAM) untuk Amazon ElastiCache

# Gambaran umum pengelolaan izin akses untuk sumber daya ElastiCache Anda

Setiap sumber daya AWS dimiliki oleh akun AWS, dan izin untuk membuat atau mengakses sumber daya diatur oleh kebijakan izin. Administrator akun dapat melampirkan kebijakan izin pada identitas IAM (yaitu pengguna, grup, dan peran). Selain itu, Amazon ElastiCache juga mendukung pelampiran kebijakan izin pada sumber daya.

> **Note**
>
> Administrator akun (atau pengguna administrator) adalah pengguna dengan hak akses administrator. Untuk informasi selengkapnya tentang administrator, lihat Praktik Terbaik IAM dalam Panduan Pengguna IAM.

Untuk memberikan akses, tambahkan izin ke pengguna, grup, atau peran Anda:

- Pengguna dan grup di AWS IAM Identity Center:

  Buat rangkaian izin. Ikuti petunjuk dalam Buat set izin dalam Panduan Pengguna AWS IAM Identity Center.
- Pengguna yang dikelola di IAM melalui penyedia identitas:

  Buat peran untuk federasi identitas. Ikuti petunjuk dalam Membuat peran untuk penyedia identitas pihak ketiga (federasi) dalam Panduan Pengguna IAM.
- Pengguna IAM:
  - Buat peran yang dapat diambil pengguna Anda. Ikuti petunjuk dalam Membuat peran untuk pengguna IAM dalam Panduan Pengguna IAM.
  - (Tidak disarankan) Pasang kebijakan langsung ke pengguna atau tambahkan pengguna ke grup pengguna. Ikuti petunjuk dalam Menambahkan izin ke pengguna (konsol) dalam Panduan Pengguna IAM.

Topik

- Kebijakan terkelola AWS untuk Amazon ElastiCache
- Menggunakan kebijakan berbasis identitas (kebijakan IAM) untuk Amazon ElastiCache
- Izin tingkat sumber daya
- Menggunakan kunci syarat
- Menggunakan Peran Tertaut Layanan untuk Amazon ElastiCache
- Izin API ElastiCache: Referensi tindakan, sumber daya, dan kondisi

## Sumber daya dan operasi Amazon ElastiCache

Untuk melihat daftar jenis sumber daya ElastiCache dan ARN-nya, lihat Sumber Daya yang Ditentukan oleh Amazon ElastiCache dalam Referensi Otorisasi Layanan. Untuk mempelajari jenis tindakan yang dapat Anda gunakan untuk menentukan ARN dari setiap sumber daya, lihat Tindakan yang Ditentukan oleh Amazon ElastiCache.

## Memahami kepemilikan sumber daya

Pemilik sumber daya adalah akun AWS yang membuat sumber daya. Artinya, pemilik sumber daya adalah akun AWS dari entitas prinsipal yang melakukan autentikasi permintaan yang membuat sumber daya. Entitas prinsipal dapat berupa akun root, pengguna IAM, atau peran IAM). Contoh berikut menggambarkan cara kerjanya:

- Misalkan Anda menggunakan kredensial akun root dari akun AWS Anda untuk membuat sebuah klaster cache. Dalam hal ini, akun AWS Anda adalah pemilik sumber daya. Di ElastiCache, sumber dayanya adalah klaster cache.
- Misalkan Anda membuat pengguna IAM dalam akun AWS Anda dan memberikan izin untuk membuat klaster cache kepada pengguna tersebut. Dalam hal ini, pengguna tersebut dapat membuat klaster cache. Namun, akun AWS Anda, yang berisi pengguna tersebut, adalah pemilik sumber daya klaster cache.
- Misalkan Anda membuat pengguna IAM dalam akun AWS Anda dengan izin untuk membuat klaster cache. Dalam hal ini, siapa pun yang dapat mengambil peran tersebut akan dapat membuat klaster cache. Akun AWS Anda, yang berisi peran tersebut, adalah pemilik sumber daya klaster cache.

## Mengelola akses ke sumber daya

Kebijakan izin menjelaskan siapa yang memiliki akses ke suatu objek. Bagian berikut menjelaskan opsi yang tersedia untuk membuat kebijakan izin.

> **Note**
>
> Bagian ini membahas penggunaan IAM dalam konteks Amazon ElastiCache. Bagian ini tidak memberikan informasi yang mendetail tentang layanan IAM. Untuk dokumentasi IAM lengkap, lihat Apa yang Dimaksud dengan IAM? dalam Panduan Pengguna IAM. Untuk informasi tentang sintaksis dan deskripsi kebijakan IAM, lihat Referensi Kebijakan IAM AWS dalam Panduan Pengguna IAM.

Kebijakan yang terlampir pada identitas IAM disebut sebagai kebijakan berbasis identitas (kebijakan IAM). Kebijakan yang dilampirkan pada sumber daya disebut sebagai kebijakan berbasis-sumber daya.

Topik

- Kebijakan berbasis identitas (kebijakan IAM)
- Menentukan elemen kebijakan: Tindakan, efek, sumber daya, dan prinsipal
- Menentukan kondisi dalam kebijakan

### Kebijakan berbasis identitas (kebijakan IAM)

Anda dapat melampirkan kebijakan ke identitas IAM Anda. Misalnya, Anda dapat melakukan hal berikut:

- Melampirkan kebijakan izin pada pengguna atau grup dalam akun Anda – Akun administrator dapat menggunakan kebijakan izin yang terkait dengan pengguna tertentu untuk memberikan izin. Dalam hal ini, izin diberikan agar pengguna tersebut dapat membuat sumber daya ElastiCache, seperti klaster cache, grup parameter, atau grup keamanan.
- Melampirkan kebijakan izin pada peran (memberikan izin lintas akun) – Anda dapat melampirkan kebijakan izin berbasis identitas ke peran IAM untuk memberikan izin lintas akun. Misalnya, administrator di Akun A dapat membuat peran untuk memberikan izin lintas akun ke akun AWS lain (misalnya, Akun B) atau layanan AWS sebagai berikut:

1. Administrator akun A membuat peran IAM dan melampirkan kebijakan izin ke peran ini yang memberikan izin pada sumber daya di akun A.
2. Administrator akun A melampirkan kebijakan kepercayaan ke peran yang mengidentifikasi Akun B sebagai prinsipal yang dapat mengambil peran tersebut.
3. Administrator Akun B kemudian dapat mendelegasikan izin untuk mengambil peran bagi semua pengguna di Akun B. Dengan demikian, pengguna di akun B dapat membuat atau mengakses sumber daya di akun A. Dalam beberapa kasus, Anda mungkin ingin memberi layanan AWS izin untuk mengambil peran tersebut. Untuk mendukung pendekatan ini, prinsipal dalam kebijakan kepercayaan juga dapat merupakan prinsipal layanan AWS.

Untuk informasi selengkapnya tentang penggunaan IAM untuk mendelegasikan izin, lihat Manajemen Akses dalam Panduan Pengguna IAM.

Berikut adalah contoh kebijakan yang mengizinkan pengguna melakukan tindakan `DescribeCacheClusters` untuk akun AWS Anda. ElastiCache juga mendukung identifikasi sumber daya tertentu menggunakan ARN sumber daya untuk tindakan API. (Pendekatan ini juga disebut sebagai izin tingkat sumber daya).

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Sid": "DescribeCacheClusters",
      "Effect": "Allow",
      "Action": [
         "elasticache:DescribeCacheClusters"],
      "Resource": resource-arn
      }
   ]
}
```

Untuk informasi selengkapnya tentang penggunaan kebijakan berbasis identitas dengan ElastiCache, lihat Menggunakan kebijakan berbasis identitas (kebijakan IAM) untuk Amazon ElastiCache. Untuk informasi selengkapnya tentang pengguna, grup, peran, dan izin, lihat Identitas (Pengguna, Grup, dan Peran) dalam Panduan Pengguna IAM.

Menentukan elemen kebijakan: Tindakan, efek, sumber daya, dan prinsipal

Untuk setiap sumber daya Amazon ElastiCache (lihat Sumber daya dan operasi Amazon ElastiCache), layanan menentukan kumpulan operasi API (lihat Tindakan). Untuk memberikan

izin bagi operasi API ini, ElastiCache menentukan kumpulan tindakan yang dapat Anda tetapkan dalam kebijakan. Misalnya, untuk sumber daya klaster ElastiCache, tindakan berikut ditentukan: `CreateCacheCluster`, `DeleteCacheCluster`, and `DescribeCacheCluster`. Operasi API dapat memerlukan izin untuk lebih dari satu tindakan.

Berikut adalah elemen-elemen kebijakan yang paling dasar:

- Sumber daya – Dalam kebijakan, Anda menggunakan Amazon Resource Name (ARN) untuk mengidentifikasi sumber daya yang diatur kebijakan. Untuk informasi selengkapnya, lihat Sumber daya dan operasi Amazon ElastiCache.
- Tindakan – Anda menggunakan kata kunci tindakan untuk mengidentifikasi operasi sumber daya yang ingin Anda izinkan atau tolak. Misalnya, tergantung pada `Effect` yang ditentukan, izin `elasticache:CreateCacheCluster` mengizinkan atau menolak pengguna untuk melakukan operasi `CreateCacheCluster` Amazon ElastiCache.
- Efek – Anda menentukan efek ketika pengguna meminta tindakan tertentu—baik mengizinkan maupun menolak. Jika Anda tidak secara eksplisit memberikan akses ke (mengizinkan) sumber daya, akses akan ditolak secara implisit. Anda juga dapat secara eksplisit menolak akses ke sumber daya. Misalnya, Anda mungkin melakukannya untuk memastikan agar pengguna tidak dapat mengakses sumber daya, meskipun jika ada kebijakan berbeda yang memberikan akses.
- Prinsipal – Dalam kebijakan berbasis identitas (Kebijakan IAM), pengguna yang dilampiri kebijakan adalah prinsipal secara implisit. Untuk kebijakan berbasis sumber daya, Anda menentukan pengguna, akun, layanan, atau entitas lain yang diinginkan untuk menerima izin (berlaku hanya untuk kebijakan berbasis sumber daya).

Untuk mempelajari selengkapnya tentang sintaksis dan deskripsi kebijakan IAM, lihat Referensi Kebijakan IAM AWS dalam Panduan Pengguna IAM.

Untuk tabel yang menunjukkan semua tindakan API Amazon ElastiCache, lihat Izin API ElastiCache: Referensi tindakan, sumber daya, dan kondisi.

Menentukan kondisi dalam kebijakan

Ketika Anda memberikan izin, Anda dapat menggunakan bahasa kebijakan IAM untuk menentukan kondisi ketika kebijakan harus berlaku. Misalnya, Anda mungkin ingin kebijakan diterapkan hanya setelah tanggal tertentu. Untuk informasi selengkapnya tentang menentukan kondisi dalam bahasa kebijakan, lihat Kondisi dalam Panduan Pengguna IAM.

Untuk menyatakan kondisi, Anda menggunakan kunci kondisi standar. Untuk menggunakan kunci kondisi khusus ElastiCache, lihat Menggunakan kunci syarat. Terdapat kunci kondisi yang berlaku di seluruh AWS yang dapat Anda gunakan sesuai kebutuhan. Untuk daftar lengkap kunci di seluruh AWS, lihat Kunci yang Tersedia untuk Syarat dalam Panduan Pengguna IAM.

# Kebijakan terkelola AWS untuk Amazon ElastiCache

Kebijakan terkelola AWS adalah kebijakan mandiri yang dibuat dan dikelola AWS. Kebijakan terkelola AWS dirancang untuk memberikan izin bagi banyak kasus penggunaan umum agar Anda dapat mulai menetapkan izin kepada pengguna, grup, dan peran.

Perlu diingat bahwa kebijakan yang dikelola AWS mungkin tidak memberikan izin hak akses paling rendah untuk kasus penggunaan spesifik Anda karena kebijakan ini tersedia untuk digunakan semua pelanggan AWS. Kami menyarankan Anda untuk mengurangi izin lebih lanjut dengan menentukan kebijakan yang dikelola pelanggan yang khusus untuk kasus penggunaan Anda.

Anda tidak dapat mengubah izin yang ditentukan dalam kebijakan yang dikelola AWS. Jika AWS memperbarui izin yang ditentukan dalam sebuah kebijakan yang dikelola AWS, maka pembaruan tersebut akan memengaruhi semua identitas prinsipal (pengguna, grup, dan peran) yang terkait dengan kebijakan tersebut. AWS kemungkinan besar akan memperbarui kebijakan yang dikelola AWS saat sebuah Layanan AWS baru diluncurkan atau operasi API baru tersedia untuk layanan yang sudah ada.

Untuk informasi selengkapnya, lihat Kebijakan yang dikelola AWS dalam Panduan Pengguna IAM.

## Kebijakan terkelola AWS: ElastiCacheServiceRolePolicy

Anda tidak dapat melampirkan ElastiCacheServiceRolePolicy ke entitas IAM Anda. Kebijakan ini dilampirkan ke peran tertaut layanan yang memungkinkan ElastiCache melakukan mewakili Anda.

Kebijakan ini memungkinkan ElastiCache mengelola sumber daya AWS mewakili Anda seperti yang diperlukan untuk mengelola cache Anda:

- `ec2` - Kelola sumber daya jaringan EC2 untuk dilampirkan ke simpul cache, termasuk titik akhir VPC (untuk cache nirserver), Antarmuka Jaringan Elastis (ENI) (untuk klaster yang dirancang sendiri), dan grup keamanan.
- `cloudwatch` - Memancarkan data metrik dari layanan ke CloudWatch.
- `outposts` - Mengizinkan pembuatan simpul cache pada AWS Outpost.

Anda dapat menemukan kebijakan ElastiCacheServiceRolePolicy pada konsol IAM dan ElastiCacheServiceRolePolicy pada Panduan Referensi Kebijakan Terkelola AWS.

Kebijakan terkelola AWS: AmazonElastiCacheFullAccess

Anda dapat melampirkan kebijakan `AmazonElastiCacheFullAccess` ke identitas IAM Anda.

Kebijakan ini memungkinkan pengguna utama memiliki akses penuh ke ElastiCache menggunakan Konsol Manajemen AWS:

- `elasticache` - Mengakses semua API.
- `iam` - Membuat peran terkait layanan yang diperlukan untuk operasi layanan.
- `ec2` - Menjelaskan sumber daya EC2 dependen yang diperlukan untuk pembuatan cache (VPC, subnet, grup keamanan) dan mengizinkan pembuatan titik akhir VPC (untuk cache nirserver).
- `kms` - Mengizinkan penggunaan CMK yang dikelola pelanggan untuk enkripsi data diam.
- `cloudwatch` - Mengizinkan akses ke metrik untuk menampilkan metrik ElastiCache di konsol.
- `application-autoscaling` - Mengizinkan akses untuk menjelaskan kebijakan penskalaan otomatis untuk cache.
- `logs` - Digunakan untuk mengisi log stream untuk fungsionalitas pengiriman log di konsol.
- `firehose` - Digunakan untuk mengisi pengiriman stream untuk fungsionalitas pengiriman log di konsol.
- `s3` - Digunakan untuk mengisi bucket S3 untuk fungsionalitas pemulihan snapshot di konsol.
- `outposts` - Digunakan untuk mengisi AWS Outposts untuk pembuatan cache di konsol.
- `sns` - Digunakan untuk mengisi topik SNS untuk fungsionalitas notifikasi di konsol.

Anda dapat menemukan kebijakan AmazonElastiCacheFullAccess pada konsol IAM dan AmazonElastiCacheFullAccess pada Panduan Referensi Kebijakan Terkelola AWS.

Kebijakan terkelola AWS: AmazonElastiCacheReadOnlyAccess

Anda dapat melampirkan kebijakan `AmazonElastiCacheReadOnlyAccess` ke identitas IAM Anda.

Kebijakan ini memungkinkan pengguna utama memiliki akses baca saja ke ElastiCache menggunakan Konsol Manajemen AWS:

- `elasticache` - Mengakses API `Describe` baca saja.

Anda dapat menemukan kebijakan AmazonElastiCacheReadOnlyAccess pada konsol IAM dan AmazonElastiCacheReadOnlyAccess pada Panduan Referensi Kebijakan Terkelola AWS.

ElastiCache memperbarui kebijakan terkelola AWS

Tampilkan detail tentang pembaruan untuk kebijakan terkelola AWS untuk ElastiCache sejak layanan ini mulai melacak perubahan-perubahan tersebut. Untuk peringatan otomatis tentang perubahan pada halaman ini, silakan berlangganan ke umpan RSS di halaman Riwayat dokumen ElastiCache.

| Perubahan | Deskripsi | Tanggal |
| --- | --- | --- |
| AmazonElastiCacheFullAccess — Pembaruan ke kebijakan yang sudah ada | ElastiCache menambahkan izin baru untuk memungkinkan pengelolaan cache nirserver , dan untuk mengaktifkan penggunaan semua fitur layanan melalui konsol. | 27 November 2023 |
| ElastiCacheServiceRolePolicy – Pembaruan ke kebijakan yang sudah ada | ElastiCache menambahkan izin baru untuk memungkinkan pengelolaan titik akhir VPC untuk sumber daya cache nirserver. | 27 November 2023 |
| ElastiCache mulai melacak perubahan | ElastiCache mulai melacak perubahan untuk kebijakan terkelola AWS. | 7 Februari 2020 |

## Menggunakan kebijakan berbasis identitas (kebijakan IAM) untuk Amazon ElastiCache

Topik ini memberikan contoh kebijakan berbasis identitas di mana administrator akun dapat melampirkan kebijakan izin ke identitas IAM (yaitu, pengguna, grup, dan peran).

> **Important**
>
> Sebaiknya Anda terlebih dahulu membaca topik yang menjelaskan konsep dasar dan opsi untuk mengelola akses ke sumber daya Amazon ElastiCache. Untuk informasi selengkapnya, lihat Gambaran umum pengelolaan izin akses untuk sumber daya ElastiCache Anda.

Bagian dalam topik ini mencakup hal berikut:

- Kebijakan terkelola AWS untuk Amazon ElastiCache
- Contoh kebijakan yang dikelola pelanggan

Berikut adalah contoh kebijakan izin.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowClusterPermissions",
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateServerlessCache",
                "elasticache:CreateCacheCluster",
                "elasticache:DescribeServerlessCaches",
                "elasticache:DescribeReplicationGroups",
                "elasticache:DescribeCacheClusters",
                "elasticache:ModifyServerlessCache",
                "elasticache:ModifyReplicationGroup",
                "elasticache:ModifyCacheCluster"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowUserToPassRole",
            "Effect": "Allow",
            "Action": [ "iam:PassRole" ],
            "Resource": "arn:aws:iam::123456789012:role/EC2-roles-for-cluster"
        }
    ]
}
```

Kebijakan tersebut memiliki dua pernyataan:

- Pernyataan pertama memberikan izin untuk tindakan Amazon ElastiCache (`elasticache:Create*`, `elasticache:Describe*`, `elasticache:Modify*`)
- Pernyataan kedua memberikan izin untuk tindakan IAM (`iam:PassRole`) pada nama peran IAM yang ditentukan pada akhir nilai `Resource`.

Kebijakan tidak menentukan elemen `Principal` karena dalam kebijakan berbasis identitas, Anda tidak menentukan pengguna utama yang mendapatkan izin. Saat Anda menyematkan kebijakan kepada pengguna, pengguna tersebut menjadi pengguna utama secara implisit. Saat Anda menyematkan kebijakan izin pada peran IAM, pengguna utama yang diidentifikasi dalam kebijakan kepercayaan peran tersebut akan mendapatkan izin.

Untuk tabel yang menampilkan semua tindakan API Amazon ElastiCache dan sumber daya yang diterapkan, lihat Izin API ElastiCache: Referensi tindakan, sumber daya, dan kondisi.

Contoh kebijakan yang dikelola pelanggan

Jika Anda tidak menggunakan kebijakan default dan memilih untuk menggunakan kebijakan yang dikelola khusus, pastikan salah satu dari dua hal berikut. Apakah Anda harus memiliki izin untuk memanggil `iam:createServiceLinkedRole` (untuk informasi selengkapnya, lihat Contoh 4: Mengizinkan pengguna untuk memanggil API CreateServiceLinkedRole IAM). Atau Anda perlu membuat peran tertaut layanan ElastiCache.

Saat digabungkan dengan izin minimum yang diperlukan untuk menggunakan konsol Amazon ElastiCache, contoh kebijakan di bagian ini memberikan izin tambahan. Contoh ini juga relevan untuk SDK AWS dan AWS CLI.

Untuk instruksi pengaturan pengguna dan grup IAM, lihat Membuat Pengguna dan Grup Administrator IAM Pertama Anda dalam Panduan Pengguna IAM.

> **Important**
>
> Selalu uji kebijakan IAM Anda secara menyeluruh sebelum menggunakannya dalam produksi. Beberapa tindakan ElastiCache yang tampak sederhana mungkin memerlukan dukungan tindakan lain saat Anda menggunakan konsol ElastiCache. Misalnya, `elasticache:CreateCacheCluster` memberikan izin untuk membuat klaster cache

ElastiCache. Namun, untuk melakukan operasi ini, konsol ElastiCache menggunakan sejumlah tindakan `Describe` dan `List` untuk mengisi daftar konsol.

Contoh

- Contoh 1: Mengizinkan akses baca-saja kepada pengguna ke sumber daya ElastiCache
- Contoh 2: Mengizinkan pengguna untuk melakukan tugas umum administrator sistem ElastiCache
- Contoh 3: Mengizinkan pengguna untuk mengakses semua tindakan API ElastiCache
- Contoh 4: Mengizinkan pengguna untuk memanggil API CreateServiceLinkedRole IAM
- Contoh 5: Mengizinkan pengguna untuk tersambung ke cache nirserver menggunakan autentikasi IAM

Contoh 1: Mengizinkan akses baca-saja kepada pengguna ke sumber daya ElastiCache

Kebijakan berikut memberikan izin tindakan ElastiCache yang mengizinkan pengguna untuk menampilkan daftar sumber daya. Biasanya, Anda menyematkan jenis kebijakan izin ini untuk grup manajer.

```
{
   "Version": "2012-10-17",
   "Statement":[{
      "Sid": "ECReadOnly",
      "Effect":"Allow",
      "Action": [
          "elasticache:Describe*",
          "elasticache:List*"],
      "Resource":"*"
      }
   ]
}
```

Contoh 2: Mengizinkan pengguna untuk melakukan tugas umum administrator sistem ElastiCache

Tugas umum administrator sistem termasuk mengubah sumber daya. Administrator sistem mungkin juga ingin mendapatkan informasi tentang peristiwa ElastiCache. Kebijakan berikut memberikan izin kepada pengguna untuk melakukan tindakan ElastiCache untuk tugas umum administrator sistem tersebut. Biasanya, Anda menyematkan jenis kebijakan izin ini untuk grup administrator sistem.

```
{
   "Version": "2012-10-17",
   "Statement":[{
      "Sid": "ECAllowMutations",
      "Effect":"Allow",
      "Action":[
          "elasticache:Modify*",
          "elasticache:Describe*",
          "elasticache:ResetCacheParameterGroup"
      ],
      "Resource":"*"
      }
   ]
}
```

Contoh 3: Mengizinkan pengguna untuk mengakses semua tindakan API ElastiCache

Kebijakan berikut mengizinkan pengguna mengakses semua tindakan ElastiCache. Sebaiknya Anda memberikan jenis kebijakan izin ini hanya untuk pengguna administrator.

```
{
   "Version": "2012-10-17",
   "Statement":[{
      "Sid": "ECAllowAll",
      "Effect":"Allow",
      "Action":[
          "elasticache:*"
      ],
      "Resource":"*"
      }
   ]
}
```

Contoh 4: Mengizinkan pengguna untuk memanggil API CreateServiceLinkedRole IAM

Kebijakan berikut mengizinkan pengguna untuk memanggil API `CreateServiceLinkedRole` IAM. Sebaiknya Anda memberikan jenis kebijakan izin ini kepada pengguna yang menginvokasi operasi ElastiCache mutatif.

```
{
  "Version":"2012-10-17",
  "Statement":[
```

```
    {
      "Sid":"CreateSLRAllows",
      "Effect":"Allow",
      "Action":[
        "iam:CreateServiceLinkedRole"
      ],
      "Resource":"*",
      "Condition":{
        "StringLike":{
          "iam:AWSServiceName":"elasticache.amazonaws.com"
        }
      }
    }
  ]
}
```

Contoh 5: Mengizinkan pengguna untuk tersambung ke cache nirserver menggunakan autentikasi IAM

Kebijakan berikut mengizinkan setiap pengguna untuk tersambung ke cache nirserver menggunakan autentikasi IAM antara 01-04-2023 hingga 30-06-2023.

```
{
  "Version" : "2012-10-17",
  "Statement" :
  [
    {
      "Effect" : "Allow",
      "Action" : ["elasticache:Connect"],
      "Resource" : [
        "arn:aws:elasticache:us-east-1:123456789012:serverlesscache:*"
      ],
      "Condition": {
        "DateGreaterThan": {"aws:CurrentTime": "2023-04-01T00:00:00Z"},
        "DateLessThan": {"aws:CurrentTime": "2023-06-30T23:59:59Z"}
      }
    },
    {
      "Effect" : "Allow",
      "Action" : ["elasticache:Connect"],
      "Resource" : [
        "arn:aws:elasticache:us-east-1:123456789012:user:*"
      ]
```

```
    }
  ]
}
```

## Izin tingkat sumber daya

Anda dapat membatasi cakupan izin dengan menentukan sumber daya dalam kebijakan IAM. Banyak tindakan API ElastiCache yang mendukung jenis sumber daya yang bervariasi tergantung pada perilaku tindakan itu. Setiap pernyataan kebijakan IAM memberikan izin untuk tindakan yang dilakukan pada sumber daya. Saat tindakan tersebut tidak dilakukan pada sumber daya yang disebutkan, atau saat Anda memberikan izin untuk melakukan tindakan pada semua sumber daya, maka nilai sumber daya dalam kebijakan tersebut adalah wildcard (*). Untuk banyak tindakan API, Anda dapat membatasi sumber daya yang dapat diubah oleh pengguna dengan menentukan Amazon Resource Name (ARN) sumber daya tersebut, atau pola ARN yang cocok dengan beberapa sumber daya. Untuk membatasi izin berdasarkan sumber daya, tentukan sumber daya berdasarkan ARN.

Untuk melihat daftar jenis sumber daya ElastiCache dan ARN-nya, lihat Sumber Daya yang Ditentukan oleh Amazon ElastiCache di Referensi Otorisasi Layanan. Untuk mempelajari jenis tindakan yang dapat Anda gunakan menentukan ARN dari setiap sumber daya, lihat Tindakan yang Ditentukan oleh Amazon ElastiCache.

Contoh

- Contoh 1: Memberikan akses penuh pada pengguna ke jenis sumber daya ElastiCache tertentu
- Contoh 2: Menolak akses pengguna ke cache nirserver.

Contoh 1: Memberikan akses penuh pada pengguna ke jenis sumber daya ElastiCache tertentu

Kebijakan berikut secara eksplisit mengizinkan semua sumber daya dari jenis cache nirserver.

```
{
        "Sid": "Example1",
        "Effect": "Allow",
        "Action": "elasticache:*",
        "Resource": [
             "arn:aws:elasticache:us-east-1:account-id:serverlesscache:*"
        ]
}
```

Contoh 2: Menolak akses pengguna ke cache nirserver.

Contoh berikut secara eksplisit menolak akses ke cache nirserver tertentu.

```
{
        "Sid": "Example2",
        "Effect": "Deny",
        "Action": "elasticache:*",
        "Resource": [
            "arn:aws:elasticache:us-east-1:account-id:serverlesscache:name"
        ]
}
```

## Menggunakan kunci syarat

Anda dapat menentukan kondisi yang menentukan penerapan kebijakan IAM. Pada ElastiCache, Anda dapat menggunakan elemen `Condition` kebijakan JSON untuk membandingkan kunci dalam konteks permintaan dengan nilai kunci yang Anda tentukan di dalam kebijakan Anda. Untuk informasi selengkapnya, lihat Elemen kebijakan JSON IAM: Syarat.

Untuk melihat daftar kunci-kunci persyaratan ElastiCache, lihat Kunci persyaratan untuk Amazon ElastiCache di Rujukan Otorisasi Layanan.

Untuk melihat daftar kunci syarat global, lihat Kunci konteks syarat global AWS.

Menentukan Syarat: Menggunakan Kunci Syarat

Untuk mengimplementasikan kontrol yang lebih spesifik, Anda menuliskan kebijakan izin IAM yang menentukan syarat untuk mengontrol set parameter individu pada permintaan tertentu. Kemudian Anda menerapkan kebijakan ke pengguna, grup, atau peran IAM yang Anda buat menggunakan konsol IAM.

Untuk menerapkan persyaratan tersebut, Anda menambahkan informasi persyaratan pada pernyataan kebijakan IAM. Pada contoh berikut, Anda menentukan syarat bahwa setiap klaster cache yang dirancang sendiri akan menjadi jenis simpul `cache.r5.large`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*",
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "StringEquals": {
                    "elasticache:CacheNodeType": [
                        "cache.r5.large"
                    ]
                }
            }
        }
    ]
}
```

Untuk informasi selengkapnya, lihat: Contoh kebijakan kontrol akses Berbasis Tag.

Untuk informasi selengkapnya penggunaan operator syarat kebijakan, lihat Izin API ElastiCache: Referensi tindakan, sumber daya, dan kondisi.

Kebijakan Contoh: Menggunakan Syarat untuk Kontrol Parameter Terperinci

Bagian ini menunjukkan contoh kebijakan untuk mengimplementasikan kontrol akses yang terperinci pada parameter ElastiCache yang dicantumkan sebelumnya.

1. elasticache:MaximumDataStorage:  Menentukan penyimpanan data maksimum cache nirserver. Dengan menggunakan syarat yang disediakan, pelanggan tidak dapat membuat cache yang dapat menyimpan lebih dari jumlah data ditentukan.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowDependentResources",
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateServerlessCache"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:serverlesscachesnapshot:*",
                "arn:aws:elasticache:*:*:snapshot:*",
                "arn:aws:elasticache:*:*:usergroup:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateServerlessCache"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:serverlesscache:*"
            ],
            "Condition": {
                "NumericLessThanEquals": {
                    "elasticache:MaximumDataStorage": "30"
                },
                "StringEquals": {
                    "elasticache:DataStorageUnit": "GB"
                }
            }
        }
    ]
}
```

2. elasticache:MaximumECPUPerSecond: Menentukan nilai ECPU maksimum per detik dari cache nirserver. Dengan menggunakan syarat yang disediakan, pelanggan tidak dapat membuat cache yang dapat menjalankan ECPU per detik lebih dari dari jumlah yang ditentukan.

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Sid": "AllowDependentResources",
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateServerlessCache"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:serverlesscachesnapshot:*",
                "arn:aws:elasticache:*:*:snapshot:*",
                "arn:aws:elasticache:*:*:usergroup:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateServerlessCache"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:serverlesscache:*"
            ],
            "Condition": {
                "NumericLessThanEquals": {
                    "elasticache:MaximumECPUPerSecond": "100000"
                }
            }
        }
    ]
}
```

3. elasticache:CacheNodeType: Menentukan NodeType yang dapat dibuat pengguna. Dengan syarat yang disediakan, pelanggan dapat menentukan nilai tunggal atau rentang untuk jenis simpul.

```
{
    "Version": "2012-10-17",
    "Statement": [
         {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
```

```
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*",
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "StringEquals": {
                    "elasticache:CacheNodeType": [
                        "cache.t2.micro",
                        "cache.t2.medium"
                    ]
                }
            }
        }
    ]
}
```

4. elasticache:NumNodeGroups: Membuat grup replikasi dengan kurang dari 20 grup simpul.

```
{
    "Version": "2012-10-17",
    "Statement": [
         {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
```

```
            "Effect": "Allow",
            "Action": [
             "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
             "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "NumericLessThanEquals": {
                    "elasticache:NumNodeGroups": "20"
                }
            }
        }
    ]
}
```

5. elasticache:ReplicasPerNodeGroup: Menentukan replika per simpul antara 5 dan 10.

```
{
    "Version": "2012-10-17",
    "Statement": [
         {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "NumericGreaterThanEquals": {
                    "elasticache:ReplicasPerNodeGroup": "5"
                },
```

```
                "NumericLessThanEquals": {
                    "elasticache:ReplicasPerNodeGroup": "10"
                }
            }
        }
    ]
}
```

6. elasticache:EngineVersion: Menentukan penggunaan mesin versi 5.0.6.

```
{
    "Version": "2012-10-17",
    "Statement": [
     {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
           "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*",
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "StringEquals": {
                    "elasticache:EngineVersion": "5.0.6"
                }
            }
        }
    ]
}
```

7. elasticache:EngineType: Menentukan penggunaan mesin Redis saja.

```
{
    "Version": "2012-10-17",
    "Statement": [
         {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*",
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "StringEquals": {
                    "elasticache:EngineType": "redis"
                }
            }
        }
    ]
}
```

8. elasticache:AtRestEncryptionEnabled: Menentukan bahwa grup replikasi akan dibuat hanya dengan enkripsi aktif.

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
         {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "Bool": {
                    "elasticache:AtRestEncryptionEnabled": "true"
                }
            }
        }
    ]
}
```

9. elasticache:TransitEncryptionEnabled

   a. Atur kunci syarat `elasticache:TransitEncryptionEnabled` ke `false` untuk tindakan CreateReplicationGroup untuk menentukan bahwa grup replikasi hanya dapat dibuat ketika TLS tidak digunakan:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
```

```
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "Bool": {
                    "elasticache:TransitEncryptionEnabled": "false"
                }
            }
        }
    ]
}
```

Ketika kunci syarat `elasticache:TransitEncryptionEnabled` disetel ke `false` dalam kebijakan untuk tindakan CreateReplicationGroup, permintaan `CreateReplicationGroup` akan diizinkan hanya jika TLS tidak digunakan (yaitu, jika permintaan tidak menyertakan parameter `TransitEncryptionEnabled` yang disetel ke `true` atau parameter `TransitEncryptionMode` yang disetel ke `required`.

b. Atur kunci syarat `elasticache:TransitEncryptionEnabled` ke `true` untuk tindakan CreateReplicationGroup untuk menentukan bahwa grup replikasi hanya dapat dibuat ketika TLS sedang digunakan:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
```

```
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "Bool": {
                    "elasticache:TransitEncryptionEnabled": "true"
                }
            }
        }
    ]
}
```

Ketika kunci syarat `elasticache:TransitEncryptionEnabled` disetel ke `true` dalam kebijakan untuk tindakan CreateReplicationGroup, permintaan `CreateReplicationGroup` akan diizinkan hanya jika permintaan menyertakan parameter `TransitEncryptionEnabled` diatur ke `true` dan parameter `TransitEncryptionMode` diatur ke `required`.

c. Atur `elasticache:TransitEncryptionEnabled` ke `true` untuk tindakan `ModifyReplicationGroup` guna menentukan bahwa grup replikasi hanya dapat dimodifikasi ketika TLS sedang digunakan:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticache:ModifyReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "BoolIfExists": {
                    "elasticache:TransitEncryptionEnabled": "true"
```

```
                }
            }
        }
    ]
}
```

Ketika kunci syarat `elasticache:TransitEncryptionEnabled` disetel ke `true` dalam kebijakan untuk tindakan ModifyReplicationGroup, permintaan `ModifyReplicationGroup` akan diizinkan hanya jika permintaan menyertakan parameter `TransitEncryptionMode` yang diatur ke `required`. Parameter `TransitEncryptionEnabled` yang disetel ke `true` juga dapat disertakan, tetapi tidak diperlukan dalam kasus ini untuk mengaktifkan TLS.

10. elasticache:AutomaticFailoverEnabled: Menentukan bahwa grup replikasi akan dibuat hanya dengan failover otomatis yang aktif.

```
{
    "Version": "2012-10-17",
    "Statement": [
         {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "Bool": {
                    "elasticache:AutomaticFailoverEnabled": "true"
                }
            }
        }
```

```
    ]
}
```

11. elasticache:MultiAZEnabled: Menentukan bahwa grup replikasi tidak dapat dibuat jika Multi-AZ dinonaktifkan.

```
{
    "Version": "2012-10-17",
    "Statement": [
         {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },
        {
            "Effect": "Deny",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*",
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "Bool": {
                    "elasticache:MultiAZEnabled": "false"
                }
            }
        }
    ]
}
```

12. elasticache:ClusterModeEnabled: Menentukan bahwa grup replikasi hanya dapat dibuat dengan mode klaster aktif.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
         {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "Bool": {
                    "elasticache:ClusterModeEnabled": "true"
                }
            }
        }
    ]
}
```

13. elasticache:AuthTokenEnabled: Menentukan bahwa grup replikasi akan dibuat hanya dengan token AUTH yang aktif.

```
{
    "Version": "2012-10-17",
    "Statement": [

         {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
```

```
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*",
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "Bool": {
                    "elasticache:AuthTokenEnabled": "true"
                }
            }
        }
    ]
}
```

14. elasticache:SnapshotRetentionLimit: Menentukan jumlah hari (atau minimum/maksimum) untuk menyimpan snapshot. Kebijakan di bawah ini memberlakukan penyimpanan cadangan setidaknya 30 hari.

```
{
    "Version": "2012-10-17",
    "Statement": [

         {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
```

```
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup",
                "elasticache:CreateServerlessCache"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*",
                "arn:aws:elasticache:*:*:replicationgroup:*",
                "arn:aws:elasticache:*:*:serverlesscache:*"
            ],
            "Condition": {
                "NumericGreaterThanEquals": {
                    "elasticache:SnapshotRetentionLimit": "30"
                }
            }
        }
    ]
}
```

15. elasticache:KmsKeyId: Menentukan penggunaan kunci KMS AWS yang dikelola pelanggan.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
        "Sid": "AllowDependentResources",
        "Effect": "Allow",
        "Action": [
            "elasticache:CreateServerlessCache"
        ],
        "Resource": [
            "arn:aws:elasticache:*:*:serverlesscachesnapshot:*",
            "arn:aws:elasticache:*:*:snapshot:*",
            "arn:aws:elasticache:*:*:usergroup:*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "elasticache:CreateServerlessCache"
```

```
        ],
        "Resource": [
            "arn:aws:elasticache:*:*:serverlesscache:*"
        ],
        "Condition": {
            "StringEquals": {
                "elasticache:KmsKeyId": "my-key"
            }
        }
    }
  ]
}
```

16. elasticache:CacheParameterGroupName: Menentukan grup parameter non default dengan parameter tertentu dari organisasi pada klaster Anda. Anda juga dapat menentukan pola penamaan untuk grup parameter Anda atau memblokir dan menghapus nama grup parameter tertentu. Berikut ini adalah contoh pembatasan penggunaan “my-org-param-group” saja.

```
{
    "Version": "2012-10-17",
    "Statement": [

         {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*",
                "arn:aws:elasticache:*:*:replicationgroup:*"
```

```
            ],
            "Condition": {
                "StringEquals": {
                    "elasticache:CacheParameterGroupName": "my-org-param-group"
                }
            }
        }
    ]
}
```

17. elasticache:CreateCacheCluster: Menolak tindaka `CreateCacheCluster` jika tag permintaan `Project` hilang atau tidak sama dengan `Dev`, `QA` atau `Prod`.

```
{
    "Version": "2012-10-17",
    "Statement": [
          {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*",
                "arn:aws:elasticache:*:*:securitygroup:*",
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ]
        },
        {
            "Effect": "Deny",
            "Action": [
                "elasticache:CreateCacheCluster"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*"
            ],
            "Condition": {
                "Null": {
                    "aws:RequestTag/Project": "true"
                }
            }
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:AddTagsToResource"
            ],
            "Resource": "arn:aws:elasticache:*:*:cluster:*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/Project": [
                        "Dev",
                        "Prod",
                        "QA"
                    ]
                }
            }
        }
    ]
}
```

18. elasticache:CacheNodeType: Mengizinkan `CreateCacheCluster` dengan `cacheNodeType` cache.r5.large atau cache.r6g.4xlarge dan tag `Project=XYZ`.

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
      "Effect": "Allow",
      "Action": [
        "elasticache:CreateCacheCluster",
        "elasticache:CreateReplicationGroup"
      ],
      "Resource": [
        "arn:aws:elasticache:*:*:parametergroup:*",
        "arn:aws:elasticache:*:*:subnetgroup:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "elasticache:CreateCacheCluster"
      ],
      "Resource": [
        "arn:aws:elasticache:*:*:cluster:*"
```

```
      ],
      "Condition": {
        "StringEqualsIfExists": {
          "elasticache:CacheNodeType": [
            "cache.r5.large",
            "cache.r6g.4xlarge"
          ]
        },
        "StringEquals": {
          "aws:RequestTag/Project": "XYZ"
        }
      }
    }
  ]
}
```

> **Note**
>
> Saat membuat kebijakan untuk menegakkan tag dan kunci syarat lainnya secara bersama, syarat `IfExists` mungkin diperlukan pada elemen kunci syarat karena persyaratan kebijakan tambahan `elasticache:AddTagsToResource` untuk permintaan pembuatan dengan parameter `--tags`.

## Menggunakan Peran Tertaut Layanan untuk Amazon ElastiCache

Amazon ElastiCache menggunakan peran tertaut layanan AWS Identity and Access Management (IAM). Peran tertaut layanan adalah tipe peran IAM unik yang tertaut langsung ke layanan AWS, seperti Amazon ElastiCache. Peran tertaut layanan Amazon ElastiCache telah ditetapkan sebelumnya oleh Amazon ElastiCache. Mereka menyertakan semua izin yang diperlukan layanan untuk memanggil layanan AWS atas nama klaster Anda.

Peran tertaut layanan mempermudah pengaturan Amazon ElastiCache karena Anda tidak perlu menambahkan izin yang diperlukan secara manual. Peran sudah ada di dalam akun AWS Anda, tetapi tertaut ke kasus penggunaan Amazon ElastiCache dan memiliki izin yang telah ditetapkan. Hanya Amazon ElastiCache yang dapat mengambil peran ini, dan hanya peran ini yang dapat menggunakan kebijakan izin yang telah ditetapkan. Anda dapat menghapus peran tersebut hanya setelah pertama kali menghapus sumber dayanya yang terkait. Hal ini melindungi sumber daya

Amazon ElastiCache karena Anda tidak dapat menghapus izin untuk mengakses sumber daya secara tidak sengaja.

Untuk informasi tentang layanan lain yang mendukung peran tertaut layanan, lihat Layanan AWS yang bisa digunakan dengan IAM dan cari layanan yang memiliki opsi Ya di kolom Peran Tertaut Layanan. Pilih Yes (Ya) bersama tautan untuk melihat dokumentasi peran tertaut layanan untuk layanan tersebut.

Daftar Isi



## Izin Peran Tertaut Layanan untuk Amazon ElastiCache

### Izin untuk membuat peran tertaut layanan

Untuk mengizinkan entitas IAM membuat AWSperan tertaut layanan ServiceRoleForElastiCache

Tambahkan pernyataan kebijakan berikut ini ke izin untuk entitas IAM:

```
{
    "Effect": "Allow",
    "Action": [
```

```
        "iam:CreateServiceLinkedRole",
        "iam:PutRolePolicy"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/elasticache.amazonaws.com/
AWSServiceRoleForElastiCache*",
    "Condition": {"StringLike": {"iam:AWSServiceName": "elasticache.amazonaws.com"}}
}
```

Untuk mengizinkan entitas IAM menghapus peran tertaut layanan AWSServiceRoleForElastiCache

Tambahkan pernyataan kebijakan berikut ini ke izin untuk entitas IAM:

```
{
    "Effect": "Allow",
    "Action": [
        "iam:DeleteServiceLinkedRole",
        "iam:GetServiceLinkedRoleDeletionStatus"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/elasticache.amazonaws.com/
AWSServiceRoleForElastiCache*",
    "Condition": {"StringLike": {"iam:AWSServiceName": "elasticache.amazonaws.com"}}
}
```

Anda juga dapat menggunakan kebijakan AWS terkelola untuk memberikan akses penuh ke Amazon ElastiCache.

Membuat Peran Tertaut Layanan (IAM)

Anda dapat membuat peran tertaut layanan menggunakan konsol IAM, CLI, atau API.

Membuat Peran Tertaut Layanan (Konsol IAM)

Anda dapat menggunakan konsol IAM untuk membuat peran tertaut layanan.

Untuk membuat peran tertaut layanan (konsol)

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi konsol IAM, pilih Peran. Kemudian pilih Buat peran baru.
3. Di bagian Pilih tipe entitas tepercaya, pilih Layanan AWS.
4. Di bagian Atau pilih layanan untuk melihat kasus penggunaannya, pilih ElastiCache.

5. Pilih Selanjutnya: Izin.
6. Di bagian Nama kebijakan, perhatikan bahwa `ElastiCacheServiceRolePolicy` diperlukan untuk peran ini. Pilih Selanjutnya: Tanda.
7. Perhatikan bahwa tanda tidak didukung untuk peran Tertaut-Layanan. Pilih Selanjutnya: Tinjau.
8. (Opsional) Untuk Deskripsi peran, edit deskripsi untuk peran tertaut layanan baru.
9. Tinjau peran, lalu pilih Buat peran.

### Membuat Peran Tertaut Layanan (CLI IAM)

Anda dapat menggunakan operasi IAM dari AWS Command Line Interface untuk membuat peran tertaut layanan. Peran ini dapat mencakup kebijakan kepercayaan dan kebijakan terkait yang diperlukan oleh layanan untuk mengambil peran tersebut.

Untuk membuat peran tertaut layanan (CLI)

Gunakan operasi berikut:

```
$ aws iam create-service-linked-role --aws-service-name elasticache.amazonaws.com
```

### Membuat Peran Tertaut Layanan (API IAM)

Anda dapat menggunakan API IAM untuk membuat peran tertaut layanan. Peran ini dapat berisi kebijakan kepercayaan dan kebijakan terkait yang diperlukan oleh layanan untuk mengambil peran tersebut.

Untuk membuat peran tertaut layanan (API)

Gunakan panggilan API CreateServiceLinkedRole. Dalam permintaan, sebutkan nama layanan `elasticache.amazonaws.com`.

### Mengedit Deskripsi Peran Tertaut Layanan untuk Amazon ElastiCache

Amazon ElastiCache tidak mengizinkan Anda untuk mengedit peran tertaut layanan AWSServiceRoleForElastiCache. Setelah Anda membuat peran terkait layanan, Anda tidak dapat mengubah nama peran karena berbagai entitas mungkin mereferensikan peran tersebut. Namun, Anda dapat mengedit penjelasan peran menggunakan IAM.

Mengedit Deskripsi Peran Tertaut Layanan (Kebijakan IAM)

Anda dapat menggunakan konsol IAM untuk mengedit deskripsi peran tertaut layanan.

Untuk mengedit deskripsi peran tertaut layanan (konsol)

1. Di panel navigasi konsol IAM, pilih Peran.
2. Pilih nama peran yang akan diubah.
3. Di ujung kanan Deskripsi peran, pilih Edit.
4. Masukkan deskripsi baru di kotak, lalu pilih Simpan.

### Mengedit Deskripsi Peran Tertaut Layanan (CLI IAM)

Anda dapat menggunakan operasi IAM dari AWS Command Line Interface untuk mengedit deskripsi peran tertaut layanan.

Untuk mengubah deskripsi peran tertaut layanan (CLI)

1. (Opsional) Untuk melihat deskripsi peran saat ini, gunakan AWS CLI untuk operasi IAM `get-role`.

   Example

   ```
   $ aws iam get-role --role-name AWSServiceRoleForElastiCache
   ```

   Gunakan nama peran, bukan ARN, untuk merujuk ke peran dengan operasi CLI. Misalnya, jika peran memiliki ARN berikut: `arn:aws:iam::123456789012:role/myrole`, peran akan dirujuk sebagai **`myrole`**.
2. Untuk memperbarui deskripsi peran tertaut layanan, gunakan AWS CLI untuk operasi IAM `update-role-description`.

   Untuk Linux, macOS, atau Unix:

   ```
   $ aws iam update-role-description \
       --role-name AWSServiceRoleForElastiCache \
       --description "new description"
   ```

   Untuk Windows:

   ```
   $ aws iam update-role-description ^
       --role-name AWSServiceRoleForElastiCache ^
       --description "new description"
   ```

### Mengedit Deskripsi Peran Tertaut Layanan (API IAM)

Anda dapat menggunakan API IAM untuk mengedit deskripsi peran tertaut layanan.

Untuk mengubah deskripsi peran tertaut layanan (API)

1. (Opsional) Untuk melihat deskripsi peran saat ini, gunakan operasi API IAM GetRole.

   Example

   ```
   https://iam.amazonaws.com/
      ?Action=GetRole
      &RoleName=AWSServiceRoleForElastiCache
      &Version=2010-05-08
      &AUTHPARAMS
   ```

2. Untuk memperbarui deskripsi peran, gunakan operasi API IAM UpdateRoleDescription.

   Example

   ```
   https://iam.amazonaws.com/
      ?Action=UpdateRoleDescription
      &RoleName=AWSServiceRoleForElastiCache
      &Version=2010-05-08
      &Description="New description"
   ```

### Menghapus Peran Tertaut Layanan untuk Amazon ElastiCache

Jika Anda tidak perlu lagi menggunakan fitur atau layanan yang memerlukan peran tertaut layanan, sebaiknya hapus peran tersebut. Dengan begitu, Anda tidak perlu lagi memantau atau memelihara entitas yang tidak digunakan. Namun, Anda harus membersihkan peran tertaut layanan sebelum dapat menghapusnya.

Amazon ElastiCache tidak menghapus peran tertaut layanan untuk Anda.

#### Membersihkan Peran Tertaut Layanan

Sebelum Anda dapat menggunakan IAM untuk menghapus peran tertaut layanan, pastikan terlebih dahulu bahwa peran tersebut tidak memiliki sumber daya (klaster atau grup replikasi) yang terkait dengannya.

Untuk memastikan peran tertaut layanan memiliki sesi aktif di konsol IAM

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi konsol IAM, pilih Peran. Lalu pilih nama (bukan kotak centang) peran AWSServiceRoleForElastiCache.
3. Di halaman Ringkasan untuk peran yang dipilih, pilih tab Penasihat Akses.
4. Di tab Penasihat Akses, tinjau aktivitas terbaru untuk peran tertaut layanan tersebut.

Untuk menghapus sumber daya Amazon ElastiCache yang memerlukan AWSServiceRoleForelAticache

- Untuk menghapus klaster, lihat referensi berikut:
  - Menggunakan AWS Management Console
  - Menggunakan AWS CLI
  - Menggunakan API ElastiCache
- Untuk menghapus grup replikasi, lihat referensi berikut:
  - Menghapus Grup Replikasi (Konsol)
  - Menghapus Grup Replikasi (AWS CLI)
  - Menghapus grup replikasi (API ElastiCache)

Menghapus Peran Tertaut Layanan (Konsol IAM)

Anda dapat menggunakan konsol IAM untuk menghapus peran tertaut layanan.

Untuk menghapus peran tertaut layanan (konsol)

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi konsol IAM, pilih Peran. Kemudian, pilih kotak centang di sebelah nama peran yang ingin dihapus, bukan nama atau baris itu sendiri.
3. Untuk Tindakan peran di bagian atas halaman, pilih Hapus peran.
4. Di kotak dialog konfirmasi, tinjau data akses terakhir layanan, yang menunjukkan waktu terakhir setiap peran yang dipilih mengakses layanan AWS. Hal ini membantu Anda mengonfirmasi aktif

tidaknya peran tersebut saat ini. Jika Anda ingin melanjutkan, pilih Ya, Hapus guna mengirimkan peran tertaut layanan untuk penghapusan.

5. Perhatikan notifikasi konsol IAM untuk memantau progres penghapusan peran tertaut layanan. Karena penghapusan peran tertaut layanan IAM bersifat asinkron, setelah Anda mengirimkan peran tersebut untuk penghapusan, tugas penghapusan dapat berhasil atau gagal. Jika tugas tersebut gagal, Anda dapat memilih Lihat detail atau Lihat Sumber Daya dari notifikasi untuk mempelajari alasan gagalnya penghapusan.

### Menghapus Peran Tertaut Layanan (IAM CLI)

Anda dapat menggunakan operasi IAM dari AWS Command Line Interface untuk menghapus peran tertaut layanan.

Untuk menghapus peran yang terhubung dengan layanan (CLI)

1. Jika Anda tidak tahu nama peran tertaut layanan yang ingin dihapus, masukkan perintah berikut. Perintah ini menampilkan daftar peran dan Amazon Resource Names (ARN) di akun Anda.

   ```
   $ aws iam get-role --role-name role-name
   ```

   Gunakan nama peran, bukan ARN, untuk merujuk ke peran dengan operasi CLI. Misalnya, jika peran memiliki ARN `arn:aws:iam::123456789012:role/myrole`, peran akan dirujuk sebagai **`myrole`**.

2. Karena peran tertaut layanan tidak dapat dihapus jika sedang digunakan atau memiliki sumber daya terkait, Anda harus mengirimkan permintaan penghapusan. Permintaan tersebut dapat ditolak jika syarat ini tidak terpenuhi. Anda harus menangkap `deletion-task-id` dari tanggapan untuk memeriksa status tugas penghapusan. Masukkan perintah berikut untuk mengirimkan permintaan penghapusan peran tertaut layanan:

   ```
   $ aws iam delete-service-linked-role --role-name role-name
   ```

3. Masukkan perintah berikut untuk memeriksa status tugas penghapusan:

   ```
   $ aws iam get-service-linked-role-deletion-status --deletion-task-id deletion-task-id
   ```

Kemungkinan status tugas penghapusan dapat berupa `NOT_STARTED`, `IN_PROGRESS`, `SUCCEEDED`, atau `FAILED`. Jika penghapusan gagal, panggilan akan mengembalikan alasan kegagalan panggilan agar Anda dapat memecahkan masalah.

### Menghapus Peran Terkait Layanan (API IAM)

Anda dapat menggunakan API IAM untuk menghapus peran tertaut layanan.

Untuk menghapus peran tertaut layanan (API)

1. Untuk mengirimkan permintaan penghapusan untuk peran tertaut layanan, panggil DeleteServiceLinkedRole. Di permintaan tersebut, tentukan nama peran.

   Karena peran tertaut layanan tidak dapat dihapus jika sedang digunakan atau memiliki sumber daya terkait, Anda harus mengirimkan permintaan penghapusan. Permintaan tersebut dapat ditolak jika syarat ini tidak terpenuhi. Anda harus menangkap `DeletionTaskId` dari tanggapan untuk memeriksa status tugas penghapusan.
2. Untuk memeriksa status penghapusan, panggil GetServiceLinkedRoleDeletionStatus. Di permintaan tersebut, tentukan `DeletionTaskId`.

   Kemungkinan status tugas penghapusan dapat berupa `NOT_STARTED`, `IN_PROGRESS`, `SUCCEEDED`, atau `FAILED`. Jika penghapusan gagal, panggilan akan mengembalikan alasan kegagalan panggilan agar Anda dapat memecahkan masalah.

### Izin API ElastiCache: Referensi tindakan, sumber daya, dan kondisi

Saat Anda menyiapkan kontrol akses dan menulis kebijakan izin untuk melampirkan kebijakan IAM (baik berbasis identitas atau berbasis sumber daya), gunakan tabel berikut sebagai referensi. Tabel berikut ini mencantumkan setiap operasi API Amazon ElastiCache dan tindakan-tindakan terkait yang dapat Anda berikan izin untuk melakukan tindakan tersebut. Anda menentukan tindakan dalam bidang `Action` kebijakan, dan Anda menentukan nilai sumber daya pada bidang `Resource` kebijakan. Kecuali dinyatakan sebaliknya, sumber daya wajib ditentukan. Beberapa bidang mencakup baik sumber daya yang diperlukan maupun sumber daya opsional. Jika tidak terdapat ARN sumber daya, sumber daya dalam kebijakan adalah wildcard (*).

Anda dapat menggunakan kunci syarat di seluruh di kebijakan ElastiCache untuk menyatakan syarat. Untuk melihat daftar berisi kunci kondisi khusus ElastiCache bersama dengan tindakan dan jenis sumber daya tujuan penerapannya, lihat Menggunakan kunci syarat. Untuk daftar lengkap kunci seluruh AWS, lihat Kunci konteks syarat global AWS AWS dalam Panduan Pengguna IAM.

> **Note**
>
> Untuk menentukan tindakan, gunakan awalan `elasticache:` diikuti dengan nama operasi API (misalnya, `elasticache:DescribeCacheClusters`).

Untuk melihat daftar tindakan ElastiCache, lihat Tindakan yang ditentukan oleh Amazon ElastiCache di Rujukan Otorisasi Layanan.

## Validasi kepatuhan untuk Amazon ElastiCache

Auditor pihak ketiga menilai keamanan dan kepatuhan layanan AWS sebagai bagian dari beberapa program kepatuhan AWS, seperti SOC, PCI, FedRAMP, dan HIPAA.

Untuk mempelajari apakah Layanan AWS berada dalam lingkup program kepatuhan khusus, lihat Layanan AWS di Scope oleh Program Program Kepatuhan yang Anda minati. Untuk informasi umum, lihat Program Kepatuhan AWS .

Anda dapat mengunduh laporan audit pihak ketiga menggunakan AWS Artifact. Untuk informasi selengkapnya, lihat Mengunduh Laporan di AWS Artifact.

Tanggung jawab kepatuhan Anda saat menggunakan Layanan AWS ditentukan berdasarkan sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, serta hukum dan peraturan yang berlaku. AWS menyediakan sumber daya berikut untuk membantu kepatuhan:

- Panduan Mulai Cepat Keamanan dan Kepatuhan – Panduan deployment ini membahas pertimbangan arsitektur dan menyediakan langkah-langkah untuk melakukan deployment lingkungan dasar di AWS yang menjadi fokus keamanan dan kepatuhan.
- Merancang Keamanan dan Kepatuhan HIPAA di Amazon Web Services – Laporan resmi ini menjelaskan cara perusahaan dapat menggunakan AWS untuk membuat aplikasi yang memenuhi syarat HIPAA.

> **Note**
>
> Tidak semua Layanan AWS memenuhi syarat HIPAA. Untuk informasi selengkapnya, lihat Referensi Layanan yang Memenuhi Syarat HIPAA.

- Sumber Daya Kepatuhan AWS – Kumpulan buku kerja dan panduan ini mungkin berlaku untuk industri dan lokasi Anda.
- Panduan Kepatuhan Pelanggan AWS – Pahami model tanggung jawab bersama melalui lensa kepatuhan. Panduan ini merangkum praktik terbaik untuk mengamankan Layanan AWS dan memetakan panduan kontrol keamanan di banyak kerangka kerja (termasuk National Institute of Standards and Technology (NIST), Payment Card Industry Security Standards Council (PCI), dan International Organization for Standardization (ISO)).
- Mengevaluasi Sumber Daya dengan Aturan dalam Panduan Developer AWS Config – Layanan AWS Config menilai seberapa baik konfigurasi sumber daya Anda dalam mematuhi praktik-praktik internal, pedoman industri, dan regulasi internal.
- AWS Security Hub – Layanan AWS ini memberikan gambaran komprehensif tentang status keamanan Anda di dalam AWS. Security Hub menggunakan kontrol keamanan untuk sumber daya AWS Anda serta untuk memeriksa kepatuhan Anda terhadap standar industri keamanan dan praktik terbaik. Untuk daftar layanan dan kontrol yang didukung, lihat Referensi kontrol Security Hub.
- AWS Audit Manager – Layanan AWS ini akan membantu Anda untuk terus-menerus mengaudit penggunaan AWS untuk menyederhanakan cara Anda mengelola risiko dan kepatuhan terhadap regulasi dan standar industri.

## Informasi selengkapnya

Untuk informasi umum tentang kepatuhan AWS Cloud, lihat hal berikut:

- Titik Akhir FIPS berdasarkan Layanan
- Pembaruan layanan di ElastiCache
- Kepatuhan AWS Cloud
- Model Tanggung Jawab Bersama
- Program Kepatuhan AWS PCI DSS

# Ketahanan di Amazon ElastiCache

Infrastruktur global AWS dibangun di sekitar Wilayah dan Zona Ketersediaan AWS. AWS Wilayah menyediakan beberapa Zona Ketersediaan yang terpisah dan terisolasi secara fisik, yang terhubung dengan jaringan latensi rendah, throughput tinggi, dan sangat redundan. Dengan Zona Ketersediaan, Anda dapat merancang serta mengoperasikan aplikasi dan basis data yang secara otomatis melakukan fail over di antara Zona Ketersediaan tanpa gangguan. Zona Ketersediaan memiliki ketersediaan dan toleransi kesalahan yang lebih baik, dan dapat diskalakan dibandingkan infrastruktur pusat data tunggal atau multi tradisional.

Untuk informasi selengkapnya tentang Wilayah AWS dan Zona Ketersediaan, silakan lihat Infrastruktur Global AWS.

Selain infrastruktur global AWS, Amazon ElastiCache memberikan penawaran beberapa fitur untuk membantu support ketahanan data dan kebutuhan Backup Anda.

Topik

- Mitigasi Kegagalan

## Mitigasi Kegagalan

Saat merencanakan ElastiCache implementasi Amazon Anda, Anda harus merencanakan sehingga kegagalan memiliki dampak minimal pada aplikasi dan data Anda. Topik pada bagian ini mencakup pendekatan yang dapat Anda ambil untuk melindungi aplikasi dan data Anda dari kegagalan.

Topik

- Mitigasi Kegagalan saat Menjalankan Redis

- Rekomendasi

## Mitigasi Kegagalan saat Menjalankan Redis

Jika menjalankan mesin Redis, Anda memiliki opsi berikut untuk meminimalkan dampak kegagalan simpul atau Zona Ketersediaan.

### Mitigasi Kegagalan Simpul

Cache nirserver secara otomatis memitigasi kegagalan simpul dengan arsitektur Multi-AZ yang direplikasi sehingga kegagalan simpul terlihat jelas untuk aplikasi Anda. Klaster yang dirancang sendiri harus dikonfigurasi dengan tepat untuk mengurangi kegagalan simpul individu.

Untuk mengurangi dampak kegagalan simpul Redis pada klaster yang dirancang sendiri, Anda memiliki opsi berikut:

Topik

- Mitigasi Kegagalan: Grup Replikasi Redis

### Mitigasi Kegagalan: Grup Replikasi Redis

Grup replikasi Redis terdiri dari simpul primer tunggal yang digunakan untuk baca-tulis aplikasi Anda, dan dari 1 hingga 5 simpul replika baca-saja. Setiap kali data ditulis ke simpul primer, data juga diperbarui secara asinkron ke simpul replika baca.

Saat salah satu replika baca gagal

1. ElastiCache mendeteksi replika baca yang gagal.
2. ElastiCache mengambil node yang gagal off line.
3. ElastiCache meluncurkan dan menyediakan node pengganti di AZ yang sama.
4. Simpul baru melakukan sinkronisasi dengan simpul primer.

Selama proses ini, aplikasi Anda dapat terus melakukan baca-tulis menggunakan simpul lain.

### Multi-AZ Redis

Anda dapat mengaktifkan Multi-AZ pada grup replikasi Redis Anda. Terlepas dari apakah Anda mengaktifkan Multi-AZ atau tidak, primer yang gagal akan terdeteksi dan diganti secara otomatis. Namun prosesnya akan berbeda tergantung apakah Multi-AZ aktif atau tidak.

Jika Multi-AZ aktif

1. ElastiCache mendeteksi kegagalan node primer.
2. ElastiCache mempromosikan simpul replika baca dengan lag replikasi paling sedikit ke simpul utama.
3. Replika lain melakukan sinkronisasi dengan simpul primer yang baru.
4. ElastiCache memutar replika baca di AZ primer yang gagal.
5. Simpul baru disinkronkan dengan primer yang baru dipromosikan.

Melakukan fail over ke simpul replika umumnya lebih cepat daripada membuat dan memprovisikan simpul primer baru. Ini berarti aplikasi Anda dapat melanjutkan kembali proses penulisan ke simpul primer Anda lebih cepat daripada ketika Multi-AZ tidak diaktifkan.

Untuk informasi selengkapnya, lihat Meminimalkan waktu henti di ElastiCache for Redis dengan Multi-AZ.

Bila Multi-AZ tidak aktif

1. ElastiCache mendeteksi kegagalan primer.
2. ElastiCache mengambil offline utama.
3. ElastiCache membuat dan menyediakan simpul utama baru untuk menggantikan primer yang gagal.
4. ElastiCache menyinkronkan primer baru dengan salah satu replika yang ada.
5. Setelah sinkronisasi selesai, simpul baru berfungsi sebagai simpul primer klaster.

Selama langkah 1 sampai 4 proses ini, aplikasi Anda tidak dapat menulis ke simpul primer. Namun, aplikasi Anda dapat terus membaca dari simpul replika Anda.

Untuk perlindungan tambahan, sebaiknya Anda meluncurkan simpul pada grup replikasi Anda di Zona Ketersediaan (AZ) yang berbeda. Jika Anda melakukan ini, kegagalan AZ hanya akan berdampak pada simpul di AZ tersebut dan bukan yang lain.

Untuk informasi selengkapnya, lihat Ketersediaan tinggi menggunakan grup replikasi.

Mitigasi Kegagalan Zona Ketersediaan

Cache nirserver secara otomatis memitigasi kegagalan simpul dengan arsitektur Multi-AZ yang direplikasi sehingga kegagalan AZ terlihat jelas untuk aplikasi Anda.

Untuk mengurangi dampak kegagalan Zona Ketersediaan di klaster yang dirancang sendiri, tempatkan simpul untuk setiap serpihan di sebanyak mungkin Zona Ketersediaan.

Sebanyak apa pun simpul yang Anda miliki di sebuah serpihan, jika semua simpul terletak di Zona Ketersediaan yang sama, kegagalan fatal pada AZ tersebut akan membuat semua data serpihan Anda hilang. Namun, jika Anda menempatkan simpul Anda di beberapa AZ, maka kegagalan dari satu AZ hanya mengakibatkan kehilangan simpul di AZ tersebut.

Setiap kali kehilangan simpul, Anda dapat mengalami penurunan kinerja karena operasi baca menjadi terbagi ke simpul yang lebih sedikit. Penurunan kinerja ini akan berlanjut hingga simpul tersebut diganti.

Untuk informasi penentuan Zona Ketersediaan untuk simpul Redis, lihat Membuat klaster Redis (Mode Klaster Dinonaktifkan) (Konsol).

Untuk informasi selengkapnya wilayah dan Zona Ketersediaan, lihat Memilih wilayah dan zona ketersediaan.

## Rekomendasi

Sebaiknya buat cache nirserver alih-alih klaster yang dirancang sendiri, karena Anda secara otomatis mendapatkan toleransi kesalahan yang lebih baik tanpa konfigurasi tambahan. Saat membuat klaster yang dirancang sendiri, ada dua jenis kegagalan yang perlu direncanakan: kegagalan simpul tunggal dan kegagalan Zona Ketersediaan yang luas. Rencana mitigasi kegagalan terbaik akan mengatasi kedua jenis kegagalan tersebut.

### Meminimalkan Dampak Kegagalan Simpul

Untuk meminimalkan dampak dari kegagalan simpul, sebaiknya implementasi Anda menggunakan beberapa simpul di setiap serpihan dan mendistribusikan simpul di beberapa Zona Ketersediaan. Ini dilakukan secara otomatis untuk cache nirserver.

Untuk cluster yang dirancang sendiri, kami menyarankan Anda mengaktifkan Multi-AZ pada grup replikasi Anda sehingga secara otomatis ElastiCache akan gagal ke replika jika node utama gagal.

### Meminimalkan Dampak Kegagalan Zona Ketersediaan

Untuk meminimalkan dampak kegagalan Zona Ketersediaan, sebaiknya Anda meluncurkan simpul Anda di sebanyak mungkin Availability Zone yang tersedia. Menyebarkan simpul Anda secara merata

di seluruh AZ akan meminimalkan dampak jika terjadi kegagalan AZ. Ini dilakukan secara otomatis untuk cache nirserver.

Tindakan pencegahan lain

Jika Anda menjalankan Redis, maka selain cara di atas, sebaiknya Anda menjadwalkan pencadangan rutin untuk klaster Anda. Cadangan (snapshot) membuat file .rdb yang dapat Anda gunakan untuk memulihkan cache Anda jika terjadi kegagalan atau kerusakan. Lihat informasi yang lebih lengkap di Melakukan snapshot dan pemulihan.

# Keamanan infrastruktur di AWS ElastiCache

Sebagai layanan terkelola, AWS ElastiCache dilindungi oleh prosedur keamanan jaringan global AWS yang dijelaskan pada bagian Keamanan dan Kepatuhan di Pusat Arsitektur AWS.

Gunakan panggilan API AWS yang dipublikasikan untuk mengakses ElastiCache melalui jaringan. Klien harus mendukung Keamanan Lapisan Pengangkutan (TLS) 1.2 atau versi yang lebih baru. Kami merekomendasikan TLS 1.3 atau versi yang lebih baru. Klien juga harus mendukung cipher suite dengan perfect forward secrecy (PFS) seperti Ephemeral Diffie-Hellman (DHE) atau Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Sebagian besar sistem-sistem modern seperti Java 7 dan versi yang lebih baru mendukung mode-mode ini.

Selain itu, permintaan harus ditandatangani menggunakan ID kunci akses dan kunci akses rahasia yang dikaitkan dengan prinsipal IAM. Atau Anda dapat menggunakan AWS Security Token Service (AWS STS) untuk membuat kredensial keamanan sementara untuk menandatangani permintaan.

# Pembaruan layanan di ElastiCache

ElastiCache secara otomatis memonitor armada cache, cluster, dan node Anda untuk menerapkan pembaruan layanan saat tersedia. Pembaruan layanan untuk cache nirserver diterapkan secara otomatis dan transparan. Untuk cluster yang dirancang sendiri, Anda menyiapkan jendela pemeliharaan yang telah ditentukan sehingga ElastiCache dapat menerapkan pembaruan ini. Namun, dalam beberapa kasus Anda mungkin menganggap cara ini terlalu kaku dan cenderung menghambat alur bisnis Anda.

Dengan pembaruan layanan, Anda mengontrol waktu dan jenis pembaruan yang diterapkan pada klaster yang dirancang sendiri. Anda juga dapat memantau kemajuan pembaruan ini ke ElastiCache cluster yang Anda pilih secara real time.

# Mengelola pembaruan layanan

ElastiCache pembaruan layanan untuk cluster yang dirancang sendiri dirilis secara teratur. Jika Anda memiliki satu atau beberapa cluster yang dirancang sendiri yang memenuhi syarat untuk pembaruan layanan tersebut, Anda menerima pemberitahuan melalui email, SNS, Personal Health Dashboard (PHD), dan CloudWatch acara Amazon saat pembaruan dirilis. Pembaruan juga ditampilkan di halaman Pembaruan Layanan di ElastiCache konsol. Dengan menggunakan dasbor ini, Anda dapat melihat semua pembaruan layanan dan statusnya untuk ElastiCache armada Anda. Pembaruan layanan untuk cache nirserver diterapkan secara transparan dan tidak dapat dikelola melalui Pembaruan Layanan.

Anda mengontrol waktu penerapan pembaruan sebelum pembaruan otomatis dimulai. Kami sangat menyarankan agar Anda menerapkan pembaruan jenis pembaruan keamanan sesegera mungkin untuk memastikan bahwa ElastiCache cluster Anda selalu up-to-date dengan patch keamanan saat ini.

Bagian berikut membahas opsi-opsi tersebut secara terperinci.

Topik



## Menerapkan pembaruan layanan

Anda dapat memulai menerapkan pembaruan layanan untuk armada Anda sejak pembaruan berstatus tersedia. Pembaruan layanan bersifat kumulatif. Dengan kata lain, pembaruan apa pun yang belum diterapkan akan disertakan dalam pembaruan terbaru Anda.

Jika pembaruan layanan telah mengaktifkan pembaruan otomatis, Anda dapat memilih untuk tidak mengambil tindakan apa pun saat pembaruan tersebut tersedia. ElastiCache akan menjadwalkan untuk menerapkan pembaruan selama salah satu jendela pemeliharaan klaster Anda yang akan datang setelah tanggal mulai pembaruan Otomatis. Anda akan menerima notifikasi terkait untuk setiap tahap pembaruan.

> **Note**
>
> Anda dapat menerapkan hanya pembaruan layanan yang berstatus tersedia atau terjadwal.

Untuk informasi selengkapnya tentang meninjau dan menerapkan pembaruan khusus layanan apa pun ke ElastiCache kluster yang berlaku, lihat. Menerapkan pembaruan layanan menggunakan konsol

Jika pembaruan layanan baru tersedia untuk satu atau beberapa ElastiCache cluster, Anda dapat menggunakan ElastiCache konsol, API, atau AWS CLI untuk menerapkan pembaruan. Bagian berikut menjelaskan opsi yang dapat Anda gunakan untuk menerapkan pembaruan.

Menerapkan pembaruan layanan menggunakan konsol

Untuk melihat daftar pembaruan layanan yang tersedia, bersama informasi lainnya, buka halaman Pembaruan Layanan pada konsol.

1. Masuk ke AWS Management Console dan buka ElastiCache konsol Amazon di https://console.aws.amazon.com/elasticache/.
2. Pada panel navigasi, pilih Pembaruan Layanan.
3. Di bagian Pembaruan layanan, Anda dapat melihat hal berikut:

    - Nama pembaruan layanan: Nama unik pembaruan layanan
    - Jenis pembaruan: Jenis pembaruan layanan, yaitu pembaruan keamanan atau pembaruan mesin
    - Kepelikan pembaruan: Prioritas penerapan pembaruan:
        - kritis: Sebaiknya Anda menerapkan pembaruan ini segera (dalam waktu 14 hari atau kurang).
        - penting: Sebaiknya Anda menerapkan pembaruan ini secepatnya saat alur bisnis Anda memungkinkan (dalam 30 hari atau kurang).
        - sedang: Sebaiknya Anda menerapkan pembaruan ini secepatnya saat Anda bisa (dalam 60 hari atau kurang).
        - rendah: Sebaiknya Anda menerapkan pembaruan ini secepatnya saat Anda bisa (dalam 90 hari atau kurang).
    - Versi mesin: Jika jenis pembaruan adalah pembaruan mesin, yang diperbarui adalah versi mesin.

- Tanggal Rilis: Waktu saat pembaruan dirilis dan tersedia untuk diterapkan pada klaster Anda.
- Direkomendasikan Terapkan Berdasarkan Tanggal: tanggal ElastiCache panduan untuk menerapkan pembaruan oleh.
- Status: Status pembaruan, yang merupakan salah satu dari berikut ini:
  - tersedia: Pembaruan tersedia untuk klaster yang diperlukan.
  - selesai: Pembaruan telah diterapkan.
  - dibatalkan: Pembaruan telah dibatalkan dan tidak diperlukan lagi.
  - kedaluwarsa: Pembaruan tidak tersedia lagi untuk diterapkan.

4. Pilih pembaruan individual (bukan tombol di sebelah kirinya) untuk melihat detail pembaruan layanan.

   Di bagian Status pembaruan klaster, Anda dapat melihat daftar klaster yang berisi pembaruan yang belum diterapkan atau baru saja diterapkan. Untuk setiap klaster, Anda dapat melihat hal berikut:

   - Nama klaster: Nama dari klaster
   - Simpul diperbarui: Rasio simpul individual dalam klaster tertentu yang telah diperbarui atau tetap tersedia setelah pembaruan layanan tertentu.
   - Jenis Pembaruan: Jenis pembaruan layanan, yaitu pembaruan keamanan atau pembaruan mesin
   - Status: Status pembaruan layanan pada klaster, yang merupakan salah satu dari berikut ini:
     - tersedia: Pembaruan ini tersedia untuk klaster yang diperlukan.
     - sedang berlangsung: Pembaruan sedang diterapkan ke klaster ini.
     - dijadwalkan: Tanggal pembaruan telah dijadwalkan.
     - selesai: Pembaruan telah berhasil diterapkan. Klaster dengan status selesai akan ditampilkan selama 7 hari setelah selesai.

   Jika Anda memilih salah satu atau semua klaster dengan status tersedia atau dijadwalkan, lalu memilih Terapkan sekarang, pembaruan akan mulai diterapkan pada klaster tersebut.

### Menerapkan pembaruan layanan menggunakan AWS CLI

Setelah Anda menerima notifikasi bahwa pembaruan layanan telah tersedia, Anda dapat memeriksa dan menerapkannya menggunakan AWS CLI:

- Untuk mendapatkan deskripsi pembaruan layanan yang tersedia, jalankan perintah berikut:

  ```
  aws elasticache describe-service-updates --service-update-status
  available
  ```

  Untuk informasi lebih lanjut, lihat describe-service-updates.

- Untuk menerapkan pembaruan layanan pada daftar klaster, jalankan perintah berikut:

  ```
  aws elasticache batch-apply-update-action --service-update
  ServiceUpdateNameToApply=sample-service-update --cluster-names cluster-1
  cluster2
  ```

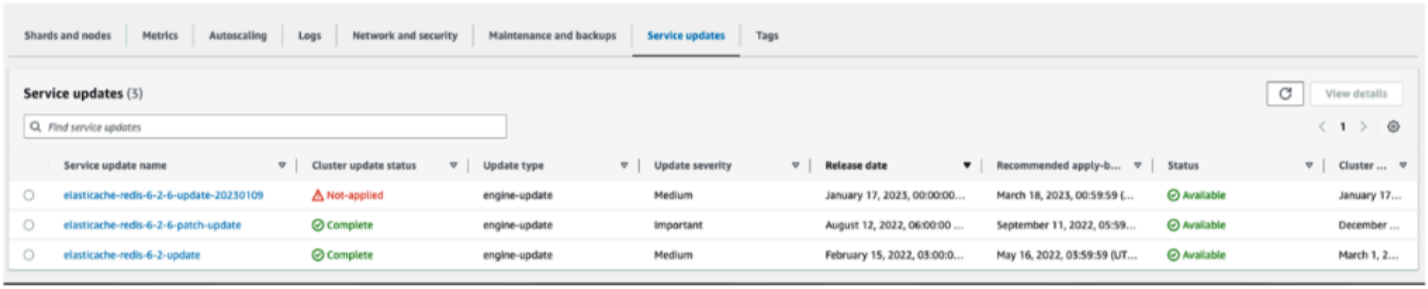
  Untuk informasi lebih lanjut, lihat batch-apply-update-action.

## Memverifikasi bahwa Anda memiliki Pembaruan Layanan terbaru yang Diterapkan menggunakan AWS konsol

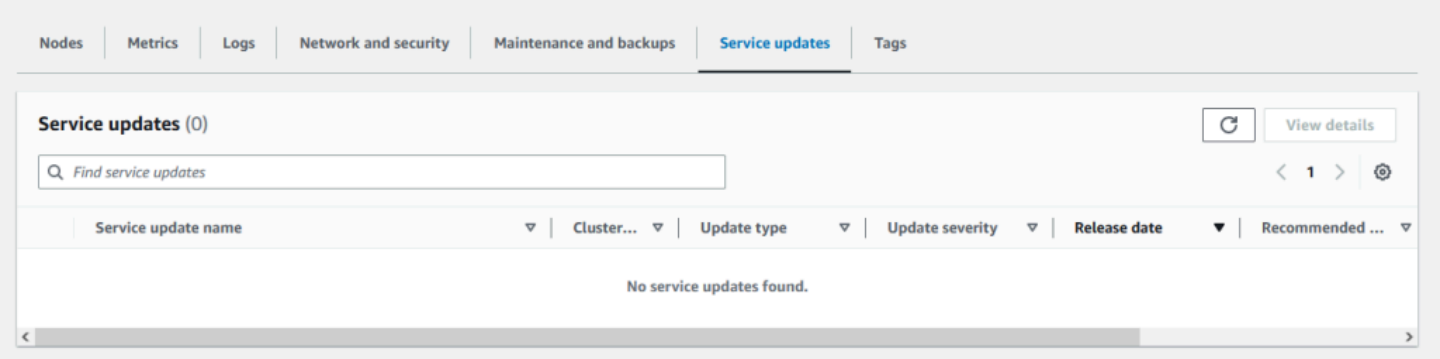
Anda dapat ElastiCache memverifikasi bahwa klaster Redis menjalankan pembaruan layanan terbaru dengan mengikuti langkah-langkah berikut:

1. Pilih klaster yang berlaku di halaman Redis Clusters
2. Pilih Pembaruan layanan di panel navigasi untuk melihat pembaruan layanan yang berlaku untuk klaster tersebut, jika ada.

Jika konsol menampilkan daftar pembaruan layanan, Anda dapat memilih pembaruan layanan dan memilih Terapkan sekarang.




Jika konsol menampilkan “Tidak ada pembaruan layanan yang ditemukan”, itu berarti klaster ElastiCache for Redis sudah menerapkan pembaruan layanan terbaru.




## Menghentikan pembaruan layanan

Anda dapat menghentikan pembaruan pada klaster jika diperlukan. Misalnya, Anda mungkin ingin menghentikan pembaruan jika Anda mengalami lonjakan tak terduga pada klaster yang sedang diperbarui. Atau Anda mungkin ingin menghentikan pembaruan yang memakan waktu terlalu lama dan mengganggu alur bisnis Anda pada jam sibuk.

Operasi Menghentikan akan segera menghentikan semua pembaruan pada klaster dan setiap simpul yang belum diperbarui. Operasi untuk simpul yang berstatus sedang berlangsung akan terus dilanjutkan hingga selesai. Namun, pembaruan akan dihentikan pada simpul lain di klaster yang sama yang berstatus pembaruan tersedia dan mengubah statusnya ke Menghentikan.

Saat alur kerja Menghentikan selesai, simpul yang berstatus Menghentikan akan berubah menjadi Dihentikan. Tergantung pada alur kerja pembaruan, simpul pada beberapa klaster mungkin tidak akan diperbarui sama sekali. Klaster lain mungkin akan menyertakan beberapa simpul yang sudah diperbarui dan simpul lain yang masih berstatus pembaruan tersedia.

Anda dapat kembali nanti untuk menyelesaikan proses pembaruan ketika alur bisnis Anda memungkinkan. Dalam kasus ini, pilih klaster yang sesuai yang ingin diselesaikan pembaruannya, lalu pilih Terapkan Sekarang. Untuk informasi selengkapnya, lihat Menerapkan pembaruan layanan.

### Menggunakan konsol

Anda dapat mengganggu pembaruan layanan menggunakan ElastiCache konsol. Contoh berikut menunjukkan cara melakukannya:

- Setelah pembaruan layanan berlangsung pada cluster yang dipilih, ElastiCache konsol menampilkan tab Lihat/Berhenti Pembaruan di bagian atas dasbor. ElastiCache
- Untuk menghentikan pembaruan, pilih Hentikan Pembaruan.

- Saat Anda menghentikan pembaruan, pilih klaster dan periksa statusnya. Status klaster akan berubah menjadi Menghentikan dan pada akhirnya berstatus Dihentikan.

Menggunakan AWS CLI

Anda dapat menghentikan pembaruan layanan menggunakan AWS CLI. Contoh kode berikut ini menunjukkan cara untuk melakukannya.

Untuk grup replikasi, lakukan hal berikut:

```
aws elasticache batch-stop-update-action --service-update-name sample-service-update --replication-group-ids my-replication-group-1 my-replication-group-2
```

Untuk klaster cache, lakukan hal berikut:

```
aws elasticache batch-stop-update-action --service-update-name sample-service-update --cache-cluster-ids my-cache-cluster-1 my-cache-cluster-2
```

Untuk informasi lebih lanjut, lihat BatchStopUpdateAction.

# Kerentanan dan Eksposur Umum (CVE): Kerentanan keamanan yang ditangani untuk Redis ElastiCache

Kerentanan dan Eksposur Umum (CVE) adalah daftar entri untuk kerentanan keamanan siber yang diketahui publik. Setiap entri adalah tautan yang berisi nomor identifikasi, deskripsi, dan setidaknya satu referensi publik. Anda dapat menemukan di halaman ini daftar kerentanan keamanan yang telah ditangani ElastiCache untuk Redis.

Kami menyarankan agar Anda selalu meningkatkan ke versi terbaru ElastiCache untuk Redis agar terlindungi dari kerentanan yang diketahui. Saat mengoperasikan Cache ElastiCache Tanpa Server, perbaikan CVE secara otomatis diterapkan ke cache Anda. Saat mengoperasikan cluster yang dirancang sendiri, ElastiCache untuk Redis mengekspos komponen PATCH. Misalnya, ketika menggunakan ElastiCache untuk Redis versi 6.2.6, versi utama adalah 6, versi minor adalah 2, dan versi patch adalah 6. Versi PATCH adalah untuk perbaikan bug yang kompatibel ke belakang, perbaikan keamanan, dan perubahan non-fungsional.

Anda dapat menggunakan halaman ini untuk memverifikasi apakah versi tertentu ElastiCache untuk Redis memiliki perbaikan untuk kerentanan keamanan tertentu. Jika klaster ElastiCache for Redis

Anda menjalankan versi tanpa perbaikan keamanan, lihat tabel di bawah ini dan ambil tindakan. Anda dapat meningkatkan ke versi Redis yang lebih baru ElastiCache yang berisi perbaikan, atau jika Anda menggunakan versi ElastiCache untuk Redis yang berisi perbaikan, pastikan Anda memiliki pembaruan layanan terbaru yang diterapkan dengan merujuk ke. Mengelola pembaruan layanan Untuk informasi selengkapnya tentang versi mesin Redis yang didukung ElastiCache dan cara memutakhirkan, lihatVersi mesin dan peningkatannya.

> **Note**
>
> - Jika CVE ditujukan dalam versi ElastiCache untuk Redis, itu berarti CVE juga dibahas dalam versi yang lebih baru. Jadi misalnya jika kerentanan ditangani ElastiCache untuk Redis Versi 6.0.5, ini terus berlanjut untuk Versi 6.2.6, 7.0.7, dan 7.1.
> - Tanda bintang (*) dalam tabel berikut menunjukkan Anda harus memiliki pembaruan layanan terbaru yang diterapkan ElastiCache untuk Redis Cluster yang menjalankan Versi Redis ElastiCache untuk ditentukan untuk mengatasi kerentanan keamanan. Untuk informasi selengkapnya tentang cara memverifikasi bahwa Anda telah menerapkan pembaruan layanan terbaru untuk versi Redis ElastiCache untuk versi Redis yang dijalankan klaster Anda, lihatMengelola pembaruan layanan.

| ElastiCache untuk versi Redis | CVE Dialamatkan |
|---|---|
| Redis 6.0.5 | CVE-2022-24735 *, CVE-2022-24736 * |
| Redis 6.2.6 | CVE-2022-24834 *, CVE-2022-35977 *, CVE-2022-36021 *, CVE-2022-24735, CVE-2022-24736 |
| Redis 7.0.7 | CVE-2023-41056 *, CVE-2022-24834* , CVE-2022-35977, CVE-2022-36021, CVE-2022-24735, CVE-2022-24736 |
| Redis 7.1.0 | CVE-2023-41056, CVE-2022-24834, CVE-2022-35977, CVE-2022-36021, CVE-2022-24735 , CVE-2022-24736 |

# Logging dan pemantauan di Amazon ElastiCache

Untuk mengelola cache Anda, penting bagi Anda untuk mengetahui performa cache Anda. ElastiCache menghasilkan metrik yang diterbitkan ke Log Amazon CloudWatch untuk memantau performa cache Anda. Selain itu, ElastiCache menghasilkan peristiwa ketika perubahan signifikan terjadi pada sumber daya cache Anda (misalnya cache baru dibuat, atau cache dihapus).

Topik



## Metrik dan peristiwa nirserver

Bagian ini menjelaskan metrik dan peristiwa yang dapat Anda pantau saat bekerja dengan cache nirserver.

Topik



### Metrik cache nirserver

Namespace `AWS/ElastiCache` menyertakan metrik CloudWatch berikut untuk cache nirserver Redis.

| Metrik | Deskripsi | Unit |
| --- | --- | --- |
| BytesUsedForCache | Jumlah total byte yang digunakan oleh data yang disimpan dalam cache Anda. | Bytes |
| ElastiCacheProcessingUnits | Jumlah total ElastiCac heProcessingUnits (ECPUs) yang dikonsumsi oleh | Jumlah |

| Metrik | Deskripsi | Unit |
| --- | --- | --- |
| | permintaan yang dijalankan di cache Anda | |
| SuccessfulReadRequestLatency | Latensi permintaan baca yang berhasil. | Mikrodetik |
| SuccessfulWriteRequestLatency | Latensi permintaan tulis yang berhasil | Mikrodetik |
| TotalCmdsCount | Jumlah total semua perintah yang dijalankan pada cache Anda | Jumlah |
| CacheHitRate | Menunjukkan laju keberhasilan cache Anda. Ini dihitung menggunakan statistik `cache_hits` dan `cache_misses` dengan cara berikut: `cache_hits /(cache_hits + cache_misses)` . | Persen |
| CacheHits | Jumlah pencarian kunci baca saja yang berhasil di cache. | Jumlah |
| CurrConnections | Jumlah koneksi klien ke cache Anda. | Jumlah |
| ThrottledCmds | Jumlah permintaan yang mendapat throttling oleh ElastiCache karena beban kerja menskalakan lebih cepat daripada yang dapat diskalakan oleh ElastiCache. | Jumlah |

| Metrik | Deskripsi | Unit |
| --- | --- | --- |
| NewConnections | Jumlah seluruh koneksi yang telah diterima oleh server selama periode ini. | Jumlah |
| CurrItems | Jumlah item dalam cache. | Jumlah |
| CurrVolatileItems | Jumlah item dalam cache dengan TTL. | Jumlah |
| NetworkBytesIn | Total byte yang ditransfer ke cache | Bytes |
| NetworkBytesOut | Total byte yang ditransfer ke cache | Bytes |
| Pengosongan | Hitungan kunci yang dilakukan pengosongan oleh cache | Jumlah |
| IamAuthenticationExpirations | Jumlah total koneksi Redis yang diautentikasi IAM yang telah kedaluwarsa. Anda dapat menemukan informasi lebih lanjut tentang Autentikasi dengan IAM di panduan pengguna. | Jumlah |
| IAMAuthenticationThrottling | Jumlah total permintaan Redis AUTH atau HELLO yang diautentikasi IAM yang mendapat throttling. Anda dapat menemukan informasi lebih lanjut tentang Autentikasi dengan IAM di panduan pengguna. | Jumlah |

| Metrik | Deskripsi | Unit |
| --- | --- | --- |
| KeyAuthorizationFailures | Jumlah seluruh upaya pengguna yang gagal untuk mengakses kunci akses karena tidak mempunyai izin akses. Sebaiknya mengatur alarm untuk ini untuk mendeteksi upaya akses yang tidak sah. | Jumlah |
| AuthenticationFailures | Jumlah seluruh upaya gagal untuk autentikasi ke Redis menggunakan perintah AUTH. Sebaiknya mengatur alarm untuk ini untuk mendeteksi upaya akses yang tidak sah. | Jumlah |
| CommandAuthorizationFailures | Jumlah seluruh upaya pengguna yang gagal untuk menjalankan perintah karena tidak memiliki izin untuk memanggil perintah itu. Sebaiknya mengatur alarm untuk ini untuk mendeteksi upaya akses yang tidak sah. | Jumlah |

Metrik tingkat perintah

ElastiCache juga memancarkan metrik tingkat perintah berikut. Untuk setiap jenis perintah, ElastiCache memancarkan jumlah total perintah dan jumlah ECPU yang dikonsumsi oleh jenis perintah tersebut.

| Metrik | Deskripsi | Unit |
| --- | --- | --- |
| EvalBasedCmds | Jumlah perintah get yang telah diterima oleh cache. | Jumlah |

| Metrik | Deskripsi | Unit |
| --- | --- | --- |
| EvalBasedCmdsECPUs | ECPU dikonsumsi oleh perintah berbasis eval. | Jumlah |
| GeoSpatialBasedCmds | Jumlah seluruh perintah untuk perintah berbasis geospasial. Ini berasal dari statistik commandstats Redis. Ini berasal dengan menjumlahkan semua perintah jenis geo: geoadd, geodist, geohash, geopos, georadius, dan georadiusbymember. | Jumlah |
| GeoSpatialBasedCmdsECPUs | ECPU dikonsumsi oleh perintah berbasis geospasial. | Jumlah |
| GetTypeCmds | Jumlah seluruh perintah jenis baca saja. Ini berasal dari statistik commandstats Redis dengan menjumlahkan semua jenis perintah baca saja (get, hget, scard, dan sebagainya.) | Jumlah |
| GetTypeCmdseECPUs | ECPU dikonsumsi oleh perintah baca. | Jumlah |
| HashBasedCmds | Jumlah seluruh perintah yang berbasis hash. Ini berasal dari statistik commandstats Redis dengan menjumlahkan semua perintah yang bekerja atas satu atau lebih hash (hget, hkeys, hvals, hdel, dan sebagainya). | Jumlah |

| Metrik | Deskripsi | Unit |
| --- | --- | --- |
| HashBasedCmdsECPUs | ECPU dikonsumsi oleh perintah berbasis hash. | Jumlah |
| HyperLogLogBasedCmds | Jumlah seluruh perintah berbasis HyperLogLog. Ini berasal dari statistik commandstats Redis dengan menjumlahkan semua jenis perintah pf (pfadd, pfcount, pfmerge, dan sebagainya.). | Jumlah |
| HyperLogLogBasedCmdsECPUs | ECPU dikonsumsi oleh perintah berbasis HyperLogLog. | Jumlah |
| JsonBasedCmds | Jumlah total perintah JSON, termasuk perintah baca dan tulis. Ini berasal dari statistik commandstats Redis dengan menjumlahkan semua perintah JSON yang bekerja atas kunci JSON. | Jumlah |
| JsonBasedCmdsECPUs | ECPU dikonsumsi oleh semua perintah JSON, termasuk perintah baca dan tulis. | Jumlah |
| JsonBasedGetCmds | Jumlah seluruh perintah JSON jenis baca saja. Ini berasal dari statistik commandstats Redis dengan menjumlahkan semua perintah baca JSON yang bekerja atas kunci JSON. | Jumlah |

| Metrik | Deskripsi | Unit |
| --- | --- | --- |
| JsonBasedGetCmdsECPUs | ECPU dikonsumsi oleh perintah baca-saja JSON. | Jumlah |
| JsonBasedSetCmds | Jumlah seluruh perintah JSON jenis tulis. Ini berasal dari statistik commandstats Redis dengan menjumlahkan semua perintah tulis JSON yang bekerja atas kunci JSON. | Jumlah |
| JsonBasedSetCmdsECPUs | ECPU dikonsumsi oleh perintah tulis JSON. | Jumlah |
| KeybasedCmds | Jumlah seluruh perintah yang berbasis kunci. Ini berasal dari statistik commandstats Redis dengan menjumlah kan semua perintah yang bekerja atas satu atau lebih kunci di berbagai struktur data (del, expire, rename, dan sebagainya.). | Jumlah |
| KeyBasedCmdsECPUs | ECPUs dikonsumsi oleh perintah berbasis kunci. | Jumlah |
| ListBasedCmds | Jumlah seluruh perintah yang berbasis daftar. Ini berasal dari statistik commandstats Redis dengan menjumlahkan semua perintah yang bekerja atas satu atau lebih daftar (lindex, lrange, lpush, ltrim, dan sebagainya). | Jumlah |

| Metrik | Deskripsi | Unit |
| --- | --- | --- |
| ListBasedCmdsECPUs | ECPUs dikonsumsi oleh perintah berbasis daftar. | Jumlah |
| NonKeyTypeCmds | Jumlah seluruh perintah yang tidak berbasis kunci. Ini berasal dari statistik commandstats Redis dengan menjumlahkan semua perintah yang tidak bekerja atas kunci, misalnya, acl, dbsize atau info. | Jumlah |
| NonKeyTypeCmdsECPUs | ECPUs dikonsumsi oleh perintah yang tidak berbasis kunci. | Jumlah |
| PubSubBasedCmds | Jumlah seluruh perintah untuk fungsionalitas pub/sub. Ini berasal dari commandst atsstatistics Redis dengan menjumlahkan semua perintah yang digunakan untuk fungsionalitas pub/sub: psubscribe, publish, pubsub, punsubscribe, ssubscrib e, sunsubscribe, spublish, subscribe, dan unsubscribe. | Jumlah |
| PubSubBasedCmdsECPUs | ECPUs dikonsumsi oleh perintah berbasis pub/sub. | Jumlah |

| Metrik | Deskripsi | Unit |
| --- | --- | --- |
| SetBasedCmds | Jumlah seluruh perintah yang berbasis set. Ini berasal dari statistik commandstats Redis dengan menjumlah kan semua perintah yang bekerja atas satu atau lebih set (scard, sdiff, sadd, sunion, dan sebagainya). | Jumlah |
| SetBasedCmdsECPUs | ECPUs dikonsumsi oleh perintah berbasis set. | Jumlah |
| SetTypeCmds | Jumlah seluruh perintah jenis tulis. Ini berasal dari statistik commandstats Redis dengan menjumlahkan semua perintah jenis mutative yang bekerja pada data (set, hset, sadd, lpop, dan sebagainya.) | Jumlah |
| SetTypeCmdsECPUs | ECPUs dikonsumsi oleh perintah tulis. | Jumlah |
| SortedSetBasedCmds | Jumlah seluruh perintah yang diurutkan berbasis set. Ini berasal dari statistik commandstats Redis dengan menjumlahkan semua perintah yang bekerja atas satu atau lebih urutan set (zcount, zrange, zrank, zadd, dan sebagainya). | Jumlah |
| SortedSetBasedCmdsECPUs | ECPUs dikonsumsi berdasark an perintah berbasis urutan. | Jumlah |

| Metrik | Deskripsi | Unit |
| --- | --- | --- |
| StringBasedCmds | Jumlah seluruh perintah yang berbasis string. Ini berasal dari statistik commandstats Redis dengan menjumlahkan semua perintah yang bekerja atas satu atau lebih string (strlen, setex, setrange, dan sebagainya). | Jumlah |
| StringBasedCmdsECPUs | ECPUs dikonsumsi oleh perintah berbasis string. | Jumlah |
| StreamBasedCmds | Jumlah seluruh perintah yang berbasis stream. Ini berasal dari statistik commandstats Redis dengan menjumlahkan semua perintah yang bekerja atas satu atau lebih jenis data stream (xrange, xlen, xadd, xdel, dan sebagainya). | Jumlah |
| StreamBasedCmdsECPUs | ECPUs dikonsumsi oleh perintah berbasis stream. | Jumlah |

## Peristiwa cache nirserver

ElastiCache mencatat log peristiwa yang berhubungan dengan cache nirserver Anda. Informasi ini mencakup tanggal dan waktu peristiwa, nama sumber dan jenis sumber peristiwa, serta deskripsi dari peristiwa itu. Anda dapat dengan mudah mengambil peristiwa dari log menggunakan konsol ElastiCache, perintah describe-events AWS CLI, atau tindakan `DescribeEvents` API ElastiCache.

Anda dapat memilih untuk memantau, menyerap, mengubah, dan bertindak pada peristiwa ElastiCache menggunakan Amazon EventBridge. Pelajari lebih lanjut di Amazon EventBridge https://docs.aws.amazon.com/eventbridge/latest/userguide/.

### Melihat peristiwa ElastiCache (Konsol)

Untuk melihat peristiwa menggunakan konsol ElastiCache:

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/
2. Untuk melihat daftar dari semua peristiwa yang tersedia, pada panel navigasi, pilih Peristiwa.
3. Pada layar Peristiwa setiap baris dari daftar mewakili satu peristiwa dan menampilkan sumber peristiwa, jenis peristiwa, waktu GMT peristiwa, serta deskripsi dari peristiwa itu. Dengan menggunakan Filter Anda dapat menentukan apakah Anda ingin melihat semua peristiwa, atau hanya peristiwa dari jenis tertentu dalam daftar peristiwa.

### Melihat peristiwa ElastiCache (AWS CLI)

Untuk menghasilkan daftar peristiwa ElastiCache menggunakan AWS CLI, gunakan perintah describe-events. Anda dapat menggunakan parameter opsional untuk mengontrol jenis peristiwa yang tercantum, kerangka waktu peristiwa yang tercantum, jumlah maksimum peristiwa untuk dicantumkan, dan banyak lagi.

Kode berikut mencantumkan hingga 40 peristiwa cache nirserver.

```
aws elasticache describe-events --source-type serverless-cache --max-items 40
```

Kode berikut mencantumkan semua peristiwa untuk cache nirserver selama 24 jam terakhir (1440 menit).

```
aws elasticache describe-events --source-type serverless-cache --duration 1440
```

### Peristiwa Nirserver

Bagian ini mendokumentasikan berbagai jenis peristiwa yang mungkin Anda terima untuk cache nirserver.

#### Peristiwa Pembuatan Cache Nirserver

| Jenis-Detail | Deskripsi | Unit | Sumber | Pesan |
|---|---|---|---|---|
| Cache dibuat | Cache arn | pembuatan | cache-nirserver | Cache <cache-name> dibuat |

| Jenis-Detail | Deskripsi | Unit | Sumber | Pesan |
|---|---|---|---|---|
| | | | | dan siap digunakan. |
| Cache dibuat | Cache arn<br>Snapshot arn | pembuatan | cache-nirserver | Cache <cache-name> dibuat dan data dipulihkan dari snapshot. Cache Anda siap digunakan. |
| Pembuatan cache gagal | Cache arn | kegagalan | cache-nirserver | Gagal membuat cache <cache-name>. Alamat IP gratis tidak cukup untuk membuat titik akhir VPC. |
| Pembuatan cache gagal | Cache arn | kegagalan | cache-nirserver | Gagal membuat cache <cache-name>. Subnet tidak valid yang disediakan dalam permintaan. |
| Pembuatan cache gagal | Cache arn | kegagalan | cache-nirserver | Gagal membuat cache <cache-name>. Batas kuota tercapai untuk membuat titik akhir VPC. |

| Jenis-Detail | Deskripsi | Unit | Sumber | Pesan |
| --- | --- | --- | --- | --- |
| Pembuatan cache gagal | Cache arn | kegagalan | cache-nirserver | Gagal membuat cache <cache-name>. Anda tidak memiliki izin untuk membuat titik akhir VPC. |
| Pembuatan cache gagal | Cache arn | kegagalan | cache-nirserver | Gagal membuat cache <cache-name>. Pengguna dengan versi Redis yang tidak kompatibel tersedia dalam grup pengguna <user-group-name>. |
| Pembuatan cache gagal | Cache arn<br><br>Cache snapshot arn | kegagalan | cache-nirserver | Gagal membuat cache <cache-name>. Grup pengguna <user-group-name> yang disediakan tidak ada. |

| Jenis-Detail | Deskripsi | Unit | Sumber | Pesan |
| --- | --- | --- | --- | --- |
| Pembuatan cache gagal | Cache arn | kegagalan | cache-nirserver | Gagal membuat cache <cache-name>. Pemulihan data dari snapshot gagal karena <reason>.<br><br>Alasan kegagalan:<br><br>• gagal mengambil file dari S3.<br>• md5 yang diharapkan tidak cocok dengan md5 aktual.<br>• file RDB yang disediakan memiliki versi yang tidak didukung. |

Peristiwa Pembaruan Cache Nirserver

| Jenis-Detail | Daftar sumber daya | Kategori | Sumber | Pesan |
| --- | --- | --- | --- | --- |
| Cache diperbarui | Cache arn | perubahan konfigurasi | cache-nirserver | SecurityGroups diperbarui untuk <cache-name>. |

| Jenis-Detail | Daftar sumber daya | Kategori | Sumber | Pesan |
| --- | --- | --- | --- | --- |
| Cache diperbarui | Cache arn | perubahan konfigurasi | cache-nirserver | Tag diperbarui untuk cache <cache-name>. |
| Pembaruan cache gagal | Cache arn | perubahan konfigurasi | cache-nirserver | Pembaruan ke cache <cache-name> gagal. Pengguna dengan versi Redis yang tidak kompatibel tersedia dalam grup pengguna <user-group-name>. |
| Pembaruan cache gagal | Cache arn | perubahan konfigurasi | cache-nirserver | Pembaruan ke cache <cache-name> gagal. Pembaruan GrupKeamanan gagal. |
| Pembaruan cache gagal | Cache arn | perubahan konfigurasi | cache-nirserver | Pembaruan ke cache <cache-name> gagal. Pembaruan GrupKeamanan gagal karena izin yang tidak mencukupi. |

| Jenis-Detail | Daftar sumber daya | Kategori | Sumber | Pesan |
| --- | --- | --- | --- | --- |
| Pembaruan cache gagal | Cache arn | perubahan konfigurasi | cache-nirserver | Pembaruan ke cache <cache-name> gagal. Pembaruan GrupKeamanan gagal karena GrupKeamanan tidak valid. |

Peristiwa Penghapusan Cache Nirserver

| Jenis-Detail | Daftar sumber daya | Kategori | Sumber | Pesan |
| --- | --- | --- | --- | --- |
| Cache dihapus | Cache arn | penghapusan | cache-nirserver | Cache <cache-name> telah dihapus. |

Peristiwa Batas Penggunaan Cache Nirserver

| Jenis-Detail | Deskripsi | Unit | Sumber | Pesan |
| --- | --- | --- | --- | --- |
| Cache diperbarui | Cache arn | perubahan konfigurasi | cache-nirserver | Batas diperbarui untuk cache <cache-name>. |
| Batas cache semakin dekat | Cache arn | pemberitahuan | cache-nirserver | Slot <X> menggunakan lebih <Y> % dari batas per slot 32 GB. Misalnya, Slot 10 menggunakan |

| Jenis-Detail | Deskripsi | Unit | Sumber | Pesan |
| --- | --- | --- | --- | --- |
| | | | | lebih 90% dari batas per slot 32 GB. |
| Pembaruan cache gagal | Cache arn | kegagalan | cache-nirserver | Pembaruan batas ke cache <cache-name> gagal karena cache telah dihapus. |
| Pembaruan cache gagal | Cache arn | kegagalan | cache-nirserver | Pembaruan batas ke cache <cache-name> gagal karena konfigurasi yang tidak valid. |
| Pembaruan cache gagal | Cache arn | kegagalan | cache-nirserver | Pembaruan batas ke cache <cache-name> gagal karena data cache saat ini melebihi batas baru. Harap bersihkan beberapa data sebelum menerapkan batas. |

## Peristiwa Snapshot Cache Nirserver

| Jenis-Detail | Daftar-sumber daya | Kategori | Sumber | Pesan |
| --- | --- | --- | --- | --- |
| Snapshot dibuat | Cache arn<br>Snapshot arn | pembuatan | snapshot-cache-nirserver | Snapshot <snapshot-name> dibuat untuk cache <cache-name>. |
| Pembuatan snapshot gagal | Cache arn<br>Snapshot arn | kegagalan | snapshot-cache-nirserver | Gagal membuat snapshot untuk cache <cache-name>. Pembuatan snapshot <snapshot-name> gagal dengan Kunci yang Dikelola Pelanggan<key-id> <reason>.<br>Pesan alasan kegagalan:<br>• karena Kunci yang Dikelola Pelanggan dinonaktifkan<br>• karena Kunci yang Dikelola Pelanggan tidak dapat ditemukan<br>• karena waktu permintaan habis |

| Jenis-Detail | Daftar-sumber daya | Kategori | Sumber | Pesan |
| --- | --- | --- | --- | --- |
| Pembuatan snapshot gagal | Cache arn<br>Snapshot arn | kegagalan | snapshot-cache-nirserver | Gagal membuat snapshot untuk cache <cache-name>. Pembuatan snapshot <snapshot-name> gagal karena <reason>.<br>Alasan default:<br>• karena kesalahan internal |
| Ekspor snapshot gagal | Snapshot arn | kegagalan | snapshot-cache-nirserver | Gagal mengekspor snapshot untuk cache <cache-name>. Tidak dapat mengekspor snapshot ke bucket %s karena ElastiCache tidak memiliki izin ke bucket. |

| Jenis-Detail | Daftar-sumber daya | Kategori | Sumber | Pesan |
| --- | --- | --- | --- | --- |
| Ekspor snapshot gagal | Snapshot arn | kegagalan | snapshot-cache-nirserver | Gagal mengekspo r snapshot untuk cache <cache-name>. Tidak dapat mengekspor snapshot ke bucket '%s' karena sudah ada objek dengan nama yang sama di bucket. |
| Ekspor snapshot gagal | Snapshot arn | kegagalan | snapshot-cache-nirserver | Gagal mengekspo r snapshot untuk cache <cache-name>. Tidak dapat mengekspor snapshot ke bucket '%s' karena Id akun pemilik bucket telah berubah. |

| Jenis-Detail | Daftar-sumber daya | Kategori | Sumber | Pesan |
| --- | --- | --- | --- | --- |
| Ekspor snapshot gagal | Snapshot arn | kegagalan | snapshot-cache-nirserver | Gagal mengekspo r snapshot untuk cache <cache-name>. Tidak dapat mengekspor snapshot ke bucket '%s' karena bucket S3 tidak dapat diakses. |
| Ekspor snapshot gagal | Snapshot arn | kegagalan | snapshot-cache-nirserver | Gagal mengekspo r snapshot untuk cache <cache-name>. Tidak dapat mengekspor snapshot ke bucket '%s' karena bucket tidak dapat diakses. |

| Jenis-Detail | Daftar-sumber daya | Kategori | Sumber | Pesan |
| --- | --- | --- | --- | --- |
| Ekspor snapshot gagal | Snapshot arn | kegagalan | snapshot-cache-nirserver | Gagal mengekspo r snapshot untuk cache <cache-name>. Tidak dapat mengekspor snapshot ke bucket '%s' karena bucket tidak tersedia. |
| Ekspor snapshot gagal | Snapshot arn | kegagalan | snapshot-cache-nirserver | Gagal mengekspo r snapshot untuk cache <cache-name>. Tidak dapat mengekspo r snapshot ke bucket '%s' dengan snapshot sumber Kunci Terkelola Pelanggan %s <reason>. |

| Jenis-Detail | Daftar-sumber daya | Kategori | Sumber | Pesan |
| --- | --- | --- | --- | --- |
| Ekspor snapshot gagal | Snapshot arn | kegagalan | snapshot-cache-nirserver | Gagal mengekspor snapshot untuk cache <cache-name>. Tidak dapat mengekspor snapshot ke bucket '%s'. |
| Salinan snapshot gagal | Snapshot arn-1<br>Snapshot arn-2 | kegagalan | snapshot-cache-nirserver | Gagal menyalin snapshot <snapshot-name>. Tidak dapat menyalin snapshot '%s' ke snapshot '%s' dengan snapshot sumber Kunci Terkelola Pelanggan <key-id> <reason-name>. |

| Jenis-Detail | Daftar-sumber daya | Kategori | Sumber | Pesan |
| --- | --- | --- | --- | --- |
| Salinan snapshot gagal | Snapshot arn-1<br><br>Snapshot arn-2 | kegagalan | snapshot-cache-nirserver | Gagal menyalin snapshot <snapshot-name>. Tidak dapat menyalin snapshot '%s' ke snapshot '%s' dengan snapshot target Kunci Terkelola Pelanggan '%s' '%s'. |

# Metrik dan acara klaster yang dirancang sendiri

Bagian ini menjelaskan metrik, peristiwa, dan log yang dapat Anda lihat saat Anda menangani klaster yang dirancang sendiri.

Topik

- Metrik dan peristiwa klaster yang dirancang sendiri
- Peristiwa untuk klaster yang dirancang sendiri
- Pengiriman log
- Memantau penggunaan dengan Metrik CloudWatch
- Pemantauan Amazon SNS dari peristiwa ElastiCache

## Metrik dan peristiwa klaster yang dirancang sendiri

Saat Anda merancang klaster sendiri, ElastiCache menerbitkan metrik di setiap tingkat simpul, termasuk metrik tingkat host dan metrik cache.

Untuk informasi selengkapnya tentang metrik tingkat host, lihat Metrik Tingkat Host.

Untuk informasi selengkapnya tentang metrik tingkat simpul, lihat Metrik untuk Redis.

# Peristiwa untuk klaster yang dirancang sendiri

Peristiwa log ElastiCache yang berkaitan dengan cache yang Anda rancang sendiri. Saat menangani klaster yang dirancang sendiri, Anda dapat melihat peristiwa klaster di konsol ElastiCache, menggunakan AWS CLI, atau menggunakan Amazon Simple Notification Service (SNS). Peristiwa klaster yang dirancang sendiri tidak diterbitkan ke Amazon EventBridge.

Informasi peristiwa klaster yang dirancang sendiri menyertakan tanggal dan waktu peristiwa, nama sumber dan jenis sumber peristiwa, serta deskripsi peristiwa. Anda dapat dengan mudah mengambil peristiwa dari log menggunakan konsol ElastiCache, perintah describe-events AWS CLI, atau tindakan DescribeEvents API ElastiCache.

Melihat peristiwa ElastiCache (Konsol)

Prosedur berikut menampilkan peristiwa menggunakan konsol ElastiCache.

Untuk melihat peristiwa menggunakan konsol ElastiCache:

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/
2. Untuk melihat daftar semua peristiwa yang tersedia, pada panel navigasi, pilih Peristiwa.
3. Pada layar Peristiwa, setiap baris daftar merepresentasikan satu peristiwa dan menampilkan sumber peristiwa, jenis peristiwa, waktu GMT peristiwa, serta deskripsi peristiwa tersebut. Dengan menggunakan Filter, Anda dapat menentukan apakah Anda ingin melihat semua peristiwa, atau hanya peristiwa dari jenis tertentu dalam daftar peristiwa.

Melihat peristiwa ElastiCache (AWS CLI)

Untuk menghasilkan daftar peristiwa ElastiCache menggunakan AWS CLI, gunakan perintah describe-events. Anda dapat menggunakan parameter opsional untuk mengontrol jenis peristiwa yang tercantum, jangka waktu peristiwa yang dicantumkan, jumlah maksimum peristiwa yang akan dicantumkan, dan lainnya.

Kode berikut mencantumkan hingga 40 peristiwa klaster yang dirancang sendiri.

```
aws elasticache describe-events --source-type cache-cluster --max-items 40
```

Kode berikut mencantumkan semua peristiwa untuk cache nirserver selama 24 jam terakhir (1440 menit).

```
aws elasticache describe-events --source-type cache-cluster --duration 1440
```

Peristiwa klaster yang dirancang sendiri

Bagian ini berisi daftar peristiwa yang dapat Anda terima untuk klaster yang dirancang sendiri.

Peristiwa ElastiCache berikut memicu notifikasi Amazon SNS. Untuk informasi tentang detail peristiwa, lihat Melihat peristiwa ElastiCache.

| Nama Peristiwa | Pesan | Deskripsi |
|---|---|---|
| ElastiCache:AddCacheNodeComplete | `ElastiCache:AddCacheNodeComplete : cache-cluster` | Simpul cache telah ditambahkan ke klaster cache dan siap untuk digunakan. |
| ElastiCache:AddCacheNodeFailed karena alamat IP yang bebas tidak mencukupi | `ElastiCache:AddCacheNodeFailed : cluster-name` | Simpul cache tidak dapat ditambahkan karena alamat IP yang tersedia tidak cukup. |
| ElastiCache:CacheClusterParametersChanged | `ElastiCache:CacheClusterParametersChanged : cluster-name` | Satu atau beberapa parameter klaster cache telah berubah. |
| ElastiCache:CacheClusterProvisioningComplete | `ElastiCache:CacheClusterProvisioningComplete cluster-name-0001-005` | Penyediaan klaster cache selesai, dan simpul cache dalam klaster cache siap untuk digunakan. |
| ElastiCache:CacheClusterProvisioningFailed karena status jaringan yang tidak kompatibel | `ElastiCache:CacheClusterProvisioningFailed : cluster-name` | Percobaan dilakukan untuk meluncurkan klaster cache baru ke cloud privat virtual (VPC) yang tidak ada. |
| ElastiCache:CacheClusterScalingComplete | `CacheClusterScalingComplete : cluster-name` | Penskalaan untuk klaster cache berhasil diselesaikan. |

| Nama Peristiwa | Pesan | Deskripsi |
| --- | --- | --- |
| ElastiCache:CacheClusterScalingFailed | ElastiCache:CacheClusterScalingFailed : *nama klaster* | Operasi peningkatan skala pada klaster cache gagal. |
| ElastiCache:CacheClusterSecurityGroupModified | `ElastiCache:CacheClusterSecurityGroupModified :` *cluster-name* | Salah satu peristiwa berikut telah terjadi:<br>• Daftar grup keamanan cache yang diotorisasi untuk klaster cache telah berubah.<br>• Satu atau beberapa grup keamanan EC2 telah diotorisasi pada grup keamanan cache yang terkait dengan klaster cache.<br>• Satu atau beberapa grup keamanan EC2 baru telah dicabut otorisasinya dari salah satu grup keamanan cache yang terkait dengan klaster cache. |

| Nama Peristiwa | Pesan | Deskripsi |
| --- | --- | --- |
| ElastiCache:CacheNodeReplaceStarted | `ElastiCache:CacheNodeReplaceStarted : cluster-name` | ElastiCache telah mendeteksi bahwa host yang menjalankan simpul cache mengalami penurunan performa atau tidak dapat dijangkau dan telah mulai mengganti simpul cache.<br><br>**Note**<br>Entri DNS untuk simpul cache yang diganti tidak berubah.<br><br>Pada kebanyakan kasus, Anda tidak perlu menyegarkan daftar server untuk klien Anda ketika peristiwa ini terjadi. Namun, beberapa pustaka klien cache mungkin berhenti menggunakan simpul cache bahkan setelah ElastiCache mengganti simpul cache; dalam hal ini, aplikasi harus menyegarkan daftar server ketika peristiwa ini terjadi. |

| Nama Peristiwa | Pesan | Deskripsi |
| --- | --- | --- |
| ElastiCache:CacheNodeReplaceComplete | `ElastiCache:CacheNodeReplaceComplete : cluster-name` | ElastiCache mendeteksi bahwa host yang menjalankan simpul cache terdegradasi atau tidak terjangkau dan telah selesai mengganti simpul cache.<br><br>**Note**<br>Entri DNS untuk simpul cache yang diganti tidak berubah.<br><br>Pada kebanyakan kasus, Anda tidak perlu menyegarkan daftar server untuk klien Anda ketika peristiwa ini terjadi. Namun, beberapa pustaka klien cache mungkin berhenti menggunakan simpul cache bahkan setelah ElastiCache mengganti simpul cache; dalam hal ini, aplikasi harus menyegarkan daftar server ketika peristiwa ini terjadi. |
| ElastiCache:CacheNodesRebooted | `ElastiCache:CacheNodesRebooted : cluster-name` | Satu atau beberapa simpul cache telah di-boot ulang.<br><br>Pesan (Memcached): `"Cache node %s shutdown"` Kemudian pesan kedua: `"Cache node %s restarted"` |

| Nama Peristiwa | Pesan | Deskripsi |
| --- | --- | --- |
| ElastiCache:CertificateRenewalComplete (hanya Redis) | `ElastiCache:CertificateRenewalComplete` | Sertifikat Amazon CA berhasil diperbarui. |
| ElastiCache:CreateReplicationGroupComplete | `ElastiCache:CreateReplicationGroupComplete : cluster-name` | Grup replikasi berhasil dibuat. |
| ElastiCache:DeleteCacheClusterComplete | `ElastiCache:DeleteCacheClusterComplete : cluster-name` | Penghapusan klaster cache dan semua simpul cache terkait telah selesai. |
| ElastiCache:FailoverComplete (hanya Redis) | `ElastiCache:FailoverComplete : mycluster` | Failover ke simpul replika telah berhasil. |
| ElastiCache:ReplicationGroupIncreaseReplicaCountFinished | `ElastiCache:ReplicationGroupIncreaseReplicaCountFinished : cluster-name-0001-005` | Jumlah replika dalam klaster telah meningkat. |
| ElastiCache:ReplicationGroupIncreaseReplicaCountStarted | `ElastiCache:ReplicationGroupIncreaseReplicaCountStarted : cluster-name-0003-004` | Proses penambahan replika ke klaster Anda telah dimulai. |
| ElastiCache:NodeReplacementCanceled | `ElastiCache:NodeReplacementCanceled : cluster-name` | Sebuah simpul di klaster Anda yang dijadwalkan akan diganti tidak lagi dijadwalkan akan diganti. |

| Nama Peristiwa | Pesan | Deskripsi |
| --- | --- | --- |
| ElastiCache:NodeReplacementRescheduled | `ElastiCache:NodeReplacementRescheduled : cluster-name` | Sebuah simpul di klaster Anda yang sebelumnya dijadwalkan akan diganti telah dijadwalkan ulang untuk diganti selama periode baru yang dijelaskan dalam notifikasi.<br><br>Untuk informasi tentang tindakan yang dapat Anda ambil, lihat Mengganti simpul. |
| ElastiCache:NodeReplacementScheduled | `ElastiCache:NodeReplacementScheduled : cluster-name` | Sebuah simpul di klaster Anda dijadwalkan akan diganti selama periode yang dijelaskan dalam notifikasi.<br><br>Untuk informasi tentang tindakan yang dapat Anda ambil, lihat Mengganti simpul. |
| ElastiCache:RemoveCacheNodeComplete | `ElastiCache:RemoveCacheNodeComplete : cluster-name` | Sebuah simpul cache telah dihapus dari klaster cache. |
| ElastiCache:ReplicationGroupScalingComplete | `ElastiCache:ReplicationGroupScalingComplete : cluster-name` | Operasi peningkatan skala pada grup replikasi berhasil diselesaikan. |
| ElastiCache:ReplicationGroupScalingFailed | `"Failed applying modification to cache node type to %s."` | Operasi peningkatan skala pada grup replikasi gagal. |
| ElastiCache:ServiceUpdateAvailableForNode | `"Service update is available for cache node %s."` | Pembaruan mandiri tersedia untuk simpul. |

| Nama Peristiwa | Pesan | Deskripsi |
| --- | --- | --- |
| ElastiCache:SnapshotComplete (hanya Redis) | `ElastiCache:SnapshotComplete : cluster-name` | Sebuah snapshot cache telah berhasil diselesaikan. |
| ElastiCache:SnapshotFailed (hanya Redis) | `SnapshotFailed : cluster-name` | Sebuah snapshot cache telah gagal. Lihat peristiwa klaster cache untuk penyebab yang lebih terperinci.<br><br>Jika Anda mendeskripsikan snapshot, lihat `DescribeSnapshots` , statusnya adalah `failed`. |

# Pengiriman log

> **Note**
>
> Log Lambat Redis didukung untuk klaster cache Redis dan grup replikasi menggunakan mesin versi 6.0 seterusnya.
> Log Lambat Redis didukung untuk klaster cache Redis dan grup replikasi menggunakan mesin versi 6.2 seterusnya.

Pengiriman log memungkinkan Anda melakukan streaming SLOWLOG Redis atau Redis Engine Log ke salah satu dari dua destinasi:

- Amazon Data Firehose
- CloudWatch Log Amazon

Anda mengaktifkan dan mengonfigurasi pengiriman log saat membuat atau memodifikasi klaster menggunakan ElastiCache API. Setiap entri log akan dikirimkan ke destinasi yang ditentukan dalam salah satu dari dua format berikut: JSON atau TEXT.

Entri berjumlah tetap log Slow diambil dari mesin Redis secara berkala. Tergantung pada nilai yang ditentukan untuk parameter mesin `slowlog-max-len`, entri log lambat tambahan mungkin tidak dikirimkan ke destinasi.

Anda dapat memilih untuk mengubah konfigurasi pengiriman atau menonaktifkan pengiriman log kapan saja menggunakan AWS konsol atau salah satu API modifikasi, salah satu modify-cache-clusteratau modify-replication-group.

Anda harus menetapkan parameter `apply-immediately` untuk semua modifikasi pengiriman log.

> **Note**
>
> Biaya Amazon CloudWatch Logs berlaku saat pengiriman log diaktifkan, bahkan saat log dikirim langsung ke Amazon Data Firehose. Untuk informasi selengkapnya, lihat bagian Log Terjual di CloudWatch Harga Amazon.

## Isi dari entri log lambat

The ElastiCache for Redis Slow Log berisi informasi berikut:

- CacheClusterId— ID dari cluster cache
- CacheNodeId— ID dari node cache
- Id – Pengidentifikasi progresif unik untuk setiap entri log lambat
- Timestamp – Stempel waktu Unix menunjukkan saat perintah yang dicatat ke log diproses
- Duration – Jumlah waktu yang diperlukan untuk eksekusinya, dalam mikrodetik
- Command – Perintah yang digunakan oleh klien. Misalnya, `set foo bar` di `foo` mana kuncinya dan `bar` nilainya. ElastiCache untuk Redis menggantikan nama kunci dan nilai yang sebenarnya dengan `(2 more arguments)` untuk menghindari mengekspos data sensitif.
- ClientAddress— Alamat IP klien dan port
- ClientName— Nama klien jika diatur melalui `CLIENT SETNAME` perintah

## Isi entri log mesin

The ElastiCache for Redis Engine Log berisi informasi berikut:

- CacheClusterId— ID dari cluster cache
- CacheNodeId— ID dari node cache
- Level log — LogLevel bisa salah satu dari yang berikut:`VERBOSE("-")`,`NOTICE("*")`,`WARNING("#")`.
- Time - Waktu UTC dari pesan yang dicatat. Waktu berada dalam format berikut: `"DD MMM YYYY hh:mm:ss.ms UTC"`
- Role — Peran simpul asal log dipancarkan. Itu bisa menjadi salah satu dari yang berikut: “M” untuk Primer, “S” untuk replika, “C” untuk proses turunan penulis yang bekerja pada RDB/AOF atau “X” untuk sentinel.
- Message — Pesan log mesin Redis.

## Izin untuk mengonfigurasi pencatatan log

Anda perlu menyertakan izin IAM berikut dalam kebijakan pengguna/peran IAM:

- `logs:CreateLogDelivery`

- `logs:UpdateLogDelivery`
- `logs:DeleteLogDelivery`
- `logs:GetLogDelivery`
- `logs:ListLogDeliveries`

Untuk informasi selengkapnya, lihat Gambaran umum manajemen akses: Izin dan kebijakan.

## Spesifikasi format log dan jenis log

### Log lambat

Log lambat mendukung JSON dan TEXT

Contoh berikut menunjukkan format JSON:

```
{
  "CacheClusterId": "logslowxxxxmsxj",
  "CacheNodeId": "0001",
  "Id": 296,
  "Timestamp": 1605631822,
  "Duration (us)": 0,
  "Command": "GET ... (1 more arguments)",
  "ClientAddress": "192.168.12.104:55452",
  "ClientName": "logslowxxxxmsxj##"
}
```

Contoh berikut menunjukkan format TEXT:

```
logslowxxxxmsxj,0001,1605631822,30,GET ... (1 more
 arguments),192.168.12.104:55452,logslowxxxxmsxj##
```

### Log mesin

Log mesin mendukung JSON dan TEXT

Contoh berikut menunjukkan format JSON:

```
{
  "CacheClusterId": "xxxxxxxxxzy-engine-log-test",
  "CacheNodeId": "0001",
  "LogLevel": "VERBOSE",
```

```
  "Role": "M",
  "Time": "12 Nov 2020 01:28:57.994 UTC",
  "Message": "Replica is waiting for next BGSAVE before synchronizing with the primary.
 Check back later"
}
```

Contoh berikut menunjukkan format TEXT:

```
xxxxxxxxxxxzy-engine-log-test/0001:M 29 Oct 2020 20:12:20.499 UTC * A slow-running Lua
 script detected that is still in execution after 10000 milliseconds.
```

## ElastiCache tujuan pencatatan

Bagian ini menjelaskan tujuan pencatatan yang dapat Anda pilih untuk ElastiCache log Anda. Setiap bagian menyediakan panduan untuk mengonfigurasi pencatatan log untuk jenis tujuan dan informasi tentang semua perilaku yang spesifik untuk jenis destinasi. Setelah mengonfigurasi tujuan pencatatan, Anda dapat memberikan spesifikasinya ke konfigurasi ElastiCache logging untuk mulai masuk ke sana.

Topik

- CloudWatch Log Amazon
- Amazon Data Firehose

### CloudWatch Log Amazon

- Anda menentukan grup CloudWatch log Log tempat log akan dikirimkan.
- Log dari beberapa klaster dan grup replikasi Redis dapat dikirimkan ke grup log yang sama.
- Log stream baru akan dibuat untuk setiap simpul di dalam klaster cache atau grup replikasi dan log akan dikirimkan ke masing-masing log stream itu. Nama log stream akan menggunakan format berikut: `elasticache/${engine-name}/${cache-cluster-id}/${cache-node-id}/${log-type}`

#### Izin untuk mempublikasikan log ke CloudWatch Log

Anda harus memiliki pengaturan izin berikut ElastiCache untuk mengonfigurasi agar Redis mengirim log ke grup log Log: CloudWatch

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "logs:CreateLogDelivery",
                "logs:GetLogDelivery",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:ListLogDeliveries"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow",
            "Sid": "ElastiCacheLogging"
        },
        {
            "Sid": "ElastiCacheLoggingCWL",
            "Action": [
                "logs:PutResourcePolicy",
                "logs:DescribeResourcePolicies",
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Untuk informasi selengkapnya, lihat Log yang dikirim ke CloudWatch Log.

Amazon Data Firehose

- Anda menentukan aliran pengiriman Firehose tempat log akan dikirimkan.
- Log dari beberapa klaster dan grup replikasi Redis dapat dikirimkan ke aliran pengiriman yang sama.
- Log dari setiap simpul di dalam klaster cache atau grup replikasi akan dikirimkan ke aliran pengiriman yang sama. Anda dapat membedakan pesan log dari simpul cache yang berbeda berdasarkan `cache-cluster-id` dan `cache-node-id` yang disertakan di tiap pesan log.

- Pengiriman log ke Firehose saat ini tidak tersedia di Wilayah Asia Pasifik (Osaka).

Izin untuk mempublikasikan log ke Firehose

Anda harus memiliki izin berikut ElastiCache untuk mengonfigurasi agar Redis mengirim log ke aliran pengiriman Amazon Kinesis Data Firehose.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "logs:CreateLogDelivery",
                "logs:GetLogDelivery",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:ListLogDeliveries"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow",
            "Sid": "ElastiCacheLogging"
        },
        {
            "Sid": "ElastiCacheLoggingFHSLR",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Sid": "ElastiCacheLoggingFH",
            "Action": [
                "firehose:TagDeliveryStream"
            ],
            "Resource": "Amazon Kinesis Data Firehose delivery stream ARN",
            "Effect": "Allow"
        }
    ]
}
```

## Menentukan pengiriman log menggunakan Konsol

Menggunakan AWS Management Console Anda dapat membuat klaster Redis (mode klaster dinonaktifkan) dengan mengikuti langkah-langkah di Membuat klaster Redis (Mode Klaster Dinonaktifkan) (Konsol) atau membuat klaster Redis (mode klaster diaktifkan) menggunakan langkah-langkah di Membuat klaster Redis (mode klaster diaktifkan) (Konsol). Dalam kedua kasus, Anda membuat konfigurasi pengiriman log dengan melakukan hal berikut;

1. Di bagian pengaturan Advanced Redis, pilih Log, lalu periksa Log lambat atau Log mesin.
2. Di bagian Format log, pilih Teks atau JSON.
3. Di bagian Jenis tujuan, pilih CloudWatch Logs atau Kinesis Firehose.
4. Di bagian Tujuan log, pilih Buat baru dan masukkan salah satu dari nama bucket Amazon S3, nama grup log CloudWatchLogs atau nama aliran Kinesis Data Firehose Anda, atau pilih Pilih yang ada, lalu pilih salah satu dari nama grup CloudWatchLogs atau nama aliran Kinesis Data Firehose Anda,

Saat mengubah klaster:

Anda dapat memilih untuk mengaktifkan/menonaktifkan pengiriman log atau mengubah salah satu dari jenis tujuan, format atau tujuan:

1. Masuk ke Konsol dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Dari panel navigasi, pilih Klaster Redis.
3. Dari daftar klaster, pilih klaster yang ingin diubah. Pilih Nama klaster, bukan kotak centang di sampingnya.
4. Pada halaman Nama klaster, pilih tab Log.
5. Untuk mengaktifkan/menonaktifkan log lambat, pilih Aktifkan log lambat atau Nonaktifkan log lambat.
6. Untuk mengaktifkan/menonaktifkan log mesin, pilih Aktifkan log mesin atau Nonaktifkan log mesin.
7. Untuk mengubah konfigurasi Anda, pilih Ubah log lambat atau Ubah log mesin:
   - Di bagian Jenis tujuan, pilih CloudWatch Logs atau Kinesis Firehose.

- Di bagian Tujuan log, pilih salah satu dari Buat baru dan masukkan nama grup log CloudWatchLogs atau nama aliran Kinesis Data Firehose Anda. Atau pilih Pilih yang ada lalu pilih nama grup log CloudWatchLogs atau nama aliran Kinesis Data Firehose Anda.

## Menentukan pengiriman log menggunakan AWS CLI

### Log Lambat

Buat grup replikasi dengan pengiriman log lambat ke CloudWatch Log.

Untuk Linux, macOS, atau Unix:

```
aws elasticache create-replication-group \
    --replication-group-id test-slow-log \
    --replication-group-description test-slow-log \
    --engine redis \
    --cache-node-type cache.r5.large \
    --num-cache-clusters 2 \
    --log-delivery-configurations '{
        "LogType":"slow-log",
        "DestinationType":"cloudwatch-logs",
        "DestinationDetails":{
          "CloudWatchLogsDetails":{
            "LogGroup":"my-log-group"
          }
        },
        "LogFormat":"json"
      }'
```

Untuk Windows:

```
aws elasticache create-replication-group ^
    --replication-group-id test-slow-log ^
    --replication-group-description test-slow-log ^
    --engine redis ^
    --cache-node-type cache.r5.large ^
    --num-cache-clusters 2 ^
    --log-delivery-configurations '{
        "LogType":"slow-log",
        "DestinationType":"cloudwatch-logs",
        "DestinationDetails":{
          "CloudWatchLogsDetails":{
```

```
            "LogGroup":"my-log-group"
          }
        },
        "LogFormat":"json"
      }'
```

Ubah grup replikasi untuk mengirimkan log lambat ke CloudWatch Log

Untuk Linux, macOS, atau Unix:

```
aws elasticache modify-replication-group \
    --replication-group-id test-slow-log \
    --apply-immediately \
    --log-delivery-configurations '
    {
      "LogType":"slow-log",
      "DestinationType":"cloudwatch-logs",
      "DestinationDetails":{
        "CloudWatchLogsDetails":{

          "LogGroup":"my-log-group"
        }
      },
      "LogFormat":"json"
    }'
```

Untuk Windows:

```
aws elasticache modify-replication-group ^
    --replication-group-id test-slow-log ^
    --apply-immediately ^
    --log-delivery-configurations '
    {
      "LogType":"slow-log",
      "DestinationType":"cloudwatch-logs",
      "DestinationDetails":{
        "CloudWatchLogsDetails":{
          "LogGroup":"my-log-group"
        }
      },
      "LogFormat":"json"
    }'
```

Ubah grup replikasi untuk menonaktifkan pengiriman log lambat

Untuk Linux, macOS, atau Unix:

```
aws elasticache modify-replication-group \
    --replication-group-id test-slow-log \
    --apply-immediately \
    --log-delivery-configurations '
    {
      "LogType":"slow-log",
      "Enabled":false
    }'
```

Untuk Windows:

```
aws elasticache modify-replication-group ^
    --replication-group-id test-slow-log ^
    --apply-immediately ^
    --log-delivery-configurations '
    {
      "LogType":"slow-log",
      "Enabled":false
    }'
```

Log Mesin

Buat grup replikasi dengan pengiriman log mesin ke CloudWatch Log.

Untuk Linux, macOS, atau Unix:

```
aws elasticache create-replication-group \
    --replication-group-id test-slow-log \
    --replication-group-description test-slow-log \
    --engine redis \
    --cache-node-type cache.r5.large \
    --num-cache-clusters 2 \
    --log-delivery-configurations '{
        "LogType":"engine-log",
        "DestinationType":"cloudwatch-logs",
        "DestinationDetails":{
          "CloudWatchLogsDetails":{
            "LogGroup":"my-log-group"
```

```
          }
        },
        "LogFormat":"json"
      }'
```

Untuk Windows:

```
aws elasticache create-replication-group ^
    --replication-group-id test-slow-log ^
    --replication-group-description test-slow-log ^
    --engine redis ^
    --cache-node-type cache.r5.large ^
    --num-cache-clusters 2 ^
    --log-delivery-configurations '{
        "LogType":"engine-log",
        "DestinationType":"cloudwatch-logs",
        "DestinationDetails":{
          "CloudWatchLogsDetails":{
            "LogGroup":"my-log-group"
          }
        },
        "LogFormat":"json"
      }'
```

Ubah grup replikasi untuk mengirimkan log engine ke Firehose

Untuk Linux, macOS, atau Unix:

```
aws elasticache modify-replication-group \
    --replication-group-id test-slow-log \
    --apply-immediately \
    --log-delivery-configurations '
    {
      "LogType":"engine-log",
      "DestinationType":"kinesis-firehose",
      "DestinationDetails":{
      "KinesisFirehoseDetails":{
         "DeliveryStream":"test"
       }
     },
      "LogFormat":"json"
    }'
```

Untuk Windows:

```
aws elasticache modify-replication-group ^
    --replication-group-id test-slow-log ^
    --apply-immediately ^
    --log-delivery-configurations '
    {
      "LogType":"engine-log",
      "DestinationType":"kinesis-firehose",
      "DestinationDetails":{
      "KinesisFirehoseDetails":{
         "DeliveryStream":"test"
       }
      },
      "LogFormat":"json"
    }'
```

Ubah grup replikasi untuk beralih ke format mesin

Untuk Linux, macOS, atau Unix:

```
aws elasticache modify-replication-group \
    --replication-group-id test-slow-log \
    --apply-immediately \
    --log-delivery-configurations '
    {
       "LogType":"engine-log",
       "LogFormat":"json"
    }'
```

Untuk Windows:

```
aws elasticache modify-replication-group ^
    --replication-group-id test-slow-log ^
    --apply-immediately ^
    --log-delivery-configurations '
    {
       "LogType":"engine-log",
       "LogFormat":"json"
    }'
```

Ubah grup replikasi untuk menonaktifkan pengiriman log mesin

Untuk Linux, macOS, atau Unix:

```
aws elasticache modify-replication-group \
    --replication-group-id test-slow-log \
    --apply-immediately \
    --log-delivery-configurations '
    {
      "LogType":"engine-log",
      "Enabled":false
    }'
```

Untuk Windows:

```
aws elasticache modify-replication-group ^
    --replication-group-id test-slow-log ^
    --apply-immediately ^
    --log-delivery-configurations '
    {
      "LogType":"engine-log",
      "Enabled":false
    }'
```

# Memantau penggunaan dengan Metrik CloudWatch

ElastiCache menyediakan metrik yang memungkinkan Anda memantau klaster Anda. Anda dapat mengakses metrik ini melalui CloudWatch. Untuk informasi selengkapnya tentang CloudWatch, lihat dokumentasi CloudWatch.

ElastiCache menyediakan metrik tingkat host (misalnya, penggunaan CPU) dan metrik yang khusus untuk perangkat lunak mesin cache (misalnya, pengambilan cache dan kesalahan cache). Metrik ini diukur dan diterbitkan untuk setiap simpul Cache dalam interval waktu 60 detik.

> **Important**
>
> Sebaiknya atur alarm CloudWatch di beberapa metrik kunci tertentu, sehingga Anda akan diberitahu jika performa klaster cache Anda mulai menurun. Untuk informasi selengkapnya, lihat Metrik Apa Yang Harus Dipantau? dalam panduan ini.

Topik



## Metrik Tingkat Host

Namespace `AWS/ElastiCache` mencakup metrik tingkat host berikut untuk simpul cache individual. Metrik ini diukur dan diterbitkan untuk setiap simpul Cache dalam interval waktu 60 detik.

Lihat Juga

- Metrik untuk Redis

| Metrik | Deskripsi | Unit |
| --- | --- | --- |
| `CPUUtilization` | Persentase pemanfaatan CPU untuk keseluruh an host. Karena Redis adalah thread tunggal, | Persen |

| Metrik | Deskripsi | Unit |
| --- | --- | --- |
| | sebaiknya pantau metrik `EngineCPU Utilization` untuk simpul dengan 4 vCPU atau lebih. | |
| `CPUCreditBalance` | Jumlah kredit CPU yang diperoleh yang diakumulasi oleh instans sejak diluncurkan atau dimulai. Untuk T2 Standar, CPUCreditBalance juga mencakup jumlah kredit peluncuran yang telah diakumulasi.<br><br>Kredit diakumulasikan ke saldo kredit setelah diperoleh, dan dihapus dari saldo kredit saat digunakan. Saldo kredit memiliki batas maksimum, yang ditentukan oleh ukuran instans. Setelah batas tercapai, setiap kredit yang baru diperoleh akan dibuang. Untuk T2 Standar, kredit peluncuran tidak termasuk dalam penghitungan batas.<br><br>Kredit dalam CPUCreditBalance tersedia untuk instans untuk digunakan hingga melonjak melebihi pemanfaatan CPU acuan.<br><br>Metrik kredit CPU tersedia pada frekuensi lima menit saja.<br><br>Metrik ini tidak tersedia untuk instans performa yang dapat melonjak T2. | Kredit (Menit vCPU) |

| Metrik | Deskripsi | Unit |
| --- | --- | --- |
| CPUCreditUsage | Jumlah kredit CPU yang digunakan oleh instans untuk pemanfaatan CPU. Satu kredit CPU sama dengan satu vCPU yang berjalan dengan pemanfaatan 100% selama satu menit atau kombinasi yang setara dari vCPU, pemanfaatan, dan waktu (misalnya, satu vCPU yang berjalan dengan pemanfaatan 50% selama dua menit atau dua vCPU yang berjalan dengan pemanfaatan 25% selama dua menit).<br><br>Metrik kredit CPU tersedia pada frekuensi lima menit saja. Jika Anda menentukan periode lebih dari lima menit, gunakan statistik Sum, bukan statistik Average.<br><br>Metrik ini tidak tersedia untuk instans performa yang dapat melonjak T2. | Kredit (Menit vCPU) |
| FreeableMemory | Jumlah memori kosong yang tersedia di host. Angka ini berasal dari RAM, buffer, dan cache yang dilaporkan oleh OS sebagai memori yang dapat dibebaskan. | Byte |
| NetworkBytesIn | Jumlah byte yang telah dibaca oleh host dari jaringan. | Byte |
| NetworkBytesOut | Jumlah byte yang dikirimkan ke semua antarmuka jaringan oleh instans. | Byte |
| NetworkPacketsIn | Jumlah paket yang diterima di semua antarmuka jaringan oleh instans. Metrik ini mengidentifikasi volume lalu lintas yang masuk dari segi jumlah paket pada instans tunggal. | Jumlah |

| Metrik | Deskripsi | Unit |
| --- | --- | --- |
| NetworkPacketsOut | Jumlah paket yang dikirimkan di semua antarmuka jaringan oleh instans. Metrik ini mengidentifikasi volume lalu lintas yang keluar dari segi jumlah paket pada instans tunggal. | Jumlah |
| NetworkBandwidthInAllowanceExceeded | Jumlah paket antri atau dijatuhkan karena kumpulan bandwidth yang masuk melebihi maksimum untuk instans. | Jumlah |
| NetworkConntrackAllowanceExceeded | Jumlah paket turun karena pelacakan koneksi melebihi maksimum untuk instans dan koneksi baru tidak dapat dibuat. Hal ini dapat mengakibatkan hilangnya paket untuk lalu lintas ke atau dari instans. | Jumlah |
| NetworkBandwidthOutAllowanceExceeded | Jumlah paket antre atau dijatuhkan karena bandwidth agregat yang keluar melebihi maksimum untuk instans. | Jumlah |
| NetworkPacketsPerSecondAllowanceExceeded | Jumlah paket yang diantrekan atau dijatuhkan karena paket per detik dua arah melebihi maksimum untuk instans. | Jumlah |
| NetworkMaxBytesIn | Semburan maksimum byte yang diterima dalam setiap menit. | Byte |
| NetworkMaxBytesOut | Semburan maksimum byte yang ditransmisikan dalam setiap menit. | Byte |
| NetworkMaxPacketsIn | Semburan maksimum paket yang diterima dalam setiap menit. | Jumlah |
| NetworkMaxPacketsOut | Semburan maksimum paket yang ditransmisikan dalam setiap menit. | Jumlah |
| SwapUsage | Jumlah swap yang digunakan oleh host. | Byte |

## Metrik untuk Redis

Namespace `AWS/ElastiCache` mencakup metrik Redis berikut.

Dengan pengecualian `ReplicationLag` dan `EngineCPUUtilization`, metrik ini berasal dari perintah info Redis. Setiap metrik dihitung di tingkat simpul cache.

Untuk dokumentasi lengkap tentang perintah info Redis, lihat http://redis.io/commands/info.

Lihat Juga

- Metrik Tingkat Host

| Metrik | Deskripsi | Unit |
|---|---|---|
| `ActiveDefragHits` | Jumlah realokasi nilai per menit yang dilakukan oleh proses defragmentasi aktif. Metrik ini berasal dari statistik `active_defrag_hits` di Redis INFO. | Bilangan |
| `AuthenticationFailures` | Jumlah seluruh upaya yang gagal untuk autentikasi ke Redis menggunakan perintah AUTH. Anda dapat menemukan informasi selengkapnya tentang setiap kegagalan autentikasi menggunakan perintah ACL LOG. Sebaiknya atur peringatan untuk hal ini guna mendeteksi upaya akses yang tidak sah. | Hitungan |
| `BytesUsedForCache` | Jumlah seluruh byte yang dialokasikan oleh Redis untuk semua tujuan, termasuk set data, buffer, dan lain-lain. | Byte |
| | `Dimension: Tier=Memory` untuk klaster Redis yang menggunakan Tingkatan data: Jumlah total byte yang digunakan untuk cache oleh memori. Ini adalah nilai statistik `used_memory` di Redis INFO. | Byte |

| Metrik | Deskripsi | Unit |
| --- | --- | --- |
| | `Dimension: Tier=SSD` untuk cluster Redis yang menggunakan Tingkatan data: Jumlah total byte yang digunakan untuk cache oleh SSD. | Byte |
| `BytesReadFromDisk` | Jumlah total byte yang dibaca dari disk per menit. Didukung hanya untuk klaster yang menggunakan Tingkatan data. | Byte |
| `BytesWrittenToDisk` | Total jumlah byte yang ditulis ke disk per menit. Didukung hanya untuk klaster yang menggunakan Tingkatan data. | Byte |
| `CacheHits` | Jumlah pencarian kunci hanya-baca yang berhasil di kamus utama. Metrik ini berasal dari statistik `keyspace_hits` di Redis INFO. | Hitungan |
| `CacheMisses` | Jumlah pencarian kunci hanya-baca yang tidak berhasil di kamus utama. Metrik ini berasal dari statistik `keyspace_misses` di Redis INFO. | Jumlah |
| `CommandAuthorizationFailures` | Jumlah seluruh upaya pengguna yang gagal untuk menjalankan perintah karena tidak memiliki izin untuk memanggil perintah itu. Anda dapat menemukan informasi selengkapnya tentang setiap kegagalan autentikasi menggunakan perintah ACL LOG. Sebaiknya atur peringatan untuk hal ini guna mendeteksi upaya akses yang tidak sah. | Jumlah |

| Metrik | Deskripsi | Unit |
| --- | --- | --- |
| `CacheHitRate` | Menunjukkan efisiensi penggunaan instans Redis. Jika rasio cache lebih rendah dari sekitar 0,8, hal ini berarti bahwa sejumlah besar kunci telah dikosongkan, kedaluwarsa, atau tidak ada. Rasio ini dihitung menggunakan statistik `cache_hits` dan `cache_misses` dengan cara berikut: `cache_hits / (cache_hits + cache_misses)` . | Persen |
| `ChannelAuthorizationFailures` | Jumlah seluruh upaya pengguna yang gagal untuk mengakses saluran akses karena tidak mempunyai izin akses. Anda dapat menemukan informasi selengkapnya tentang setiap kegagalan autentikasi menggunakan perintah ACL LOG. Sebaiknya atur peringatan untuk metrik ini guna mendeteksi upaya akses yang tidak sah. | Jumlah |
| `CurrConnections` | Jumlah koneksi klien, tidak termasuk koneksi dari replika baca. ElastiCache menggunakan dua hingga empat koneksi untuk memantau cluster dalam setiap kasus. Metrik ini berasal dari statistik `connected_clients` di Redis INFO. | Jumlah |
| `CurrItems` | Jumlah item dalam cache. Metrik ini berasal dari statistik `keyspace` Redis, yang menjumlahkan semua kunci di seluruh keyspace. | Jumlah |
| | `Dimension: Tier=Memory` untuk klaster menggunakan Tingkatan data. Jumlah item dalam memori. | Jumlah |
| | `Dimension: Tier=SSD` (solid state drive) untuk klaster yang menggunakan Tingkatan data. Jumlah item dalam SSD. | Jumlah |

| Metrik | Deskripsi | Unit |
| --- | --- | --- |
| `CurrVolatileItems` | Jumlah total kunci di semua basis data yang memiliki set ttl. Metrik ini berasal dari statistik `expires` Redis, yang menjumlahkan semua kunci dengan set ttl di seluruh keyspace. | Jumlah |
| `DatabaseCapacityUsagePercentage` | Persentase kapasitas data total untuk klaster yang sedang digunakan. Metrik ini dihitung sebagai:<br><br>`used_memory/maxmemory`<br><br>Pada instans Berjenjang Data, metrik dihitung sebagai berikut:<br><br>`(used_memory - mem_not_counted_for_evict + SSD used) / (maxmemory + SSD total capacity)`<br><br>dengan `used_memory` dan `maxmemory` diambil dari Redis INFO. | Persen |

| Metrik | Deskripsi | Unit |
| --- | --- | --- |
| DatabaseCapacityUsageCountedForEvictPercentage | Persentase kapasitas data total untuk klaster yang sedang digunakan, tidak termasuk memori yang digunakan untuk overhead dan COB. Metrik ini dihitung sebagai:<br>`used_memory - mem_not_counted_for_evict/maxmemory`<br><br>Pada instans Berjenjang Data, metrik dihitung sebagai berikut:<br>`(used_memory + SSD used) / (maxmemory + SSD total capacity)`<br>dengan `used_memory` dan `maxmemory` diambil dari Redis INFO | Persen |
| DatabaseMemoryUsagePercentage | Persentase memori untuk klaster yang sedang digunakan. Persentase ini dihitung dengan menggunakan `used_memory/maxmemory` dari Redis INFO. | Persen |
| DatabaseMemoryUsageCountedForEvictPercentage | Persentase memori untuk klaster yang sedang digunakan, tidak termasuk memori yang digunakan untuk overhead dan COB. Persentase ini dihitung dengan menggunakan `used_memory-mem_not_counted_for_evict/maxmemory` dari Redis INFO. | Persen |

| Metrik | Deskripsi | Unit |
| --- | --- | --- |
| DB0AverageTTL | Mengekspos avg_ttl dari DBO, dari statistik keyspace perintah Redis INFO. Replika tidak menghentikan masa berlaku kunci. Sebaliknya, replikas menunggu simpul primer untuk menghentikan masa berlaku kunci. Ketika simpul primer menghentikan masa berlaku kunci (atau mengosongkannya karena LRU), simpul tersebut mensintesis perintah DEL, yang ditransmisikan ke semua replika. Oleh karena itu, DB0AverageTTL bernilai 0 untuk simpul replika, karena ia tidak membuat kunci kedaluwarsa sehingga tidak melacak TTL. | Milidetik |

| Metrik | Deskripsi | Unit |
| --- | --- | --- |
| `EngineCPUUtilization` | Menyediakan pemanfaatan CPU dari thread mesin Redis. Karena Redis adalah single-threaded, Anda dapat menggunak an metrik ini untuk menganalisis beban dari proses Redis itu sendiri. Metrik `EngineCPU Utilization` memberikan visibilitas yang lebih tepat dari proses Redis. Anda dapat menggunakannya dalam hubungannya dengan metrik `CPUUtilization` . `CPUUtiliz ation` mengungkapkan pemanfaatan CPU untuk instans server secara keseluruhan, termasuk sistem operasi lain dan proses manajemen. Untuk jenis simpul yang lebih besar dengan empat vCPUs atau lebih, gunakan metrik `EngineCPUUtilization` untuk memantau dan menetapkan ambang batas untuk penskalaan.<br><br>**Note**<br>Di host ElastiCache, proses latar belakang memantau host untuk memberikan pengalaman basis data terkelola. Proses latar belakang ini dapat menimbulkan beban kerja CPU yang cukup besar. Akibat ini tidak signifikan pada host yang lebih besar dengan vCPU lebih dari dua. Tetapi dapat mempengaruhi host yang lebih kecil dengan 2vCPU atau lebih sedikit. Jika Anda hanya memantau metrik `EngineCPUUtilization` , Anda tidak akan menyadari situasi ketika host kelebihan beban baik akibat penggunaa n CPU yang tinggi dari Redis maupun | Persen |

| Metrik | Deskripsi | Unit |
| --- | --- | --- |
| | penggunaan CPU yang tinggi oleh proses pemantauan latar belakang. Oleh karena itu, sebaiknya pemantaua n metrik `CPUUtilization` dilakukan untuk host dengan dua vCPU atau kurang. | |
| `Evictions` | Jumlah kunci yang telah dikosongkan karena batas `maxmemory` . Metrik ini berasal dari statistik `evicted_keys` di Redis INFO. | Jumlah |
| `GlobalDatastoreReplicationLag` | Ini adalah lag antara simpul primer dari Wilayah sekunder dan simpul primer dari Wilayah primer. Untuk Redis yang mengaktifkan mode klaster, lag ini menunjukkan penundaan maksimum di antara serpihan. | Detik |
| `IamAuthenticationExpirations` | Jumlah total koneksi Redis yang diautenti kasi IAM yang telah kedaluwarsa. Anda dapat menemukan informasi lebih lanjut tentang Autentikasi dengan IAM di panduan pengguna. | Jumlah |
| `IamAuthenticationThrottling` | Jumlah total permintaan Redis AUTH atau HELLO yang diautentikasi IAM yang mendapat throttling. Anda dapat menemukan informasi lebih lanjut tentang Autentikasi dengan IAM di panduan pengguna. | Jumlah |
| `IsMaster` | Menunjukkan apakah simpul tersebut adalah simpul primer dari serpihan/klaster saat ini. Metrik ini dapat bernilai 0 (bukan primer) atau 1 (primer). | Jumlah |

| Metrik | Deskripsi | Unit |
| --- | --- | --- |
| KeyAuthorizationFailures | Jumlah seluruh upaya pengguna yang gagal untuk mengakses kunci akses karena tidak mempunyai izin akses. Anda dapat menemukan informasi selengkapnya tentang setiap kegagalan autentikasi menggunakan perintah ACL LOG. Sebaiknya atur peringatan untuk hal ini guna mendeteksi upaya akses yang tidak sah. | Jumlah |
| KeysTracked | Jumlah kunci yang dilacak oleh pelacakan kunci Redis sebagai persentase dari `tracking-table-max-keys` . Pelacakan kunci digunakan untuk membantu caching sisi klien dan memberitahukan klien jika kunci diubah. | Jumlah |
| MemoryFragmentationRatio | Menunjukkan efisiensi dalam pengalokasian memori mesin Redis. Ambang batas tertentu menandakan perilaku yang berbeda. Nilai yang disarankan adalah memiliki fragmentasi di atas 1,0. Nilai ini dihitung dari `mem_fragmentation_ratio statistic` Redis INFO. | Bilangan |

| Metrik | Deskripsi | Unit |
| --- | --- | --- |
| `NewConnections` | Jumlah seluruh koneksi yang telah diterima oleh server selama periode ini. Metrik ini berasal dari statistik `total_connections_received` di Redis INFO.<br><br>**Note**<br>Jika Anda menggunakan ElastiCache for Redis versi 5 atau yang lebih lama, antara dua dan empat koneksi yang dilaporkan oleh metrik ini akan digunakan oleh ElastiCache untuk memantau klaster. Namun, saat menggunakan ElastiCache for Redis versi 6 atau yang lebih baru, koneksi yang digunakan oleh ElastiCache untuk memantau klaster tidak termasuk dalam metrik ini. | Jumlah |
| `NumItemsReadFromDisk` | Jumlah total item yang diambil dari disk per menit. Didukung hanya untuk klaster yang menggunakan Tingkatan data. | Jumlah |
| `NumItemsWrittenToDisk` | Jumlah total item yang ditulis ke disk per menit. Didukung hanya untuk klaster yang menggunakan Tingkatan data. | Jumlah |
| `MasterLinkHealthStatus` | Status ini memiliki dua nilai: 0 atau 1. Nilai 0 menunjukkan bahwa data di simpul primer ElastiCache tidak sinkron dengan Redis di EC2. Nilai 1 menunjukkan bahwa data sudah sinkron. Untuk menyelesaikan migrasi, gunakan operasi API CompleteMigration. | Boolean |

| Metrik | Deskripsi | Unit |
| --- | --- | --- |
| `Reclaimed` | Jumlah seluruh peristiwa kedaluwarsa kunci. Metrik ini berasal dari statistik `expired_keys` di Redis INFO. | Jumlah |
| `ReplicationBytes` | Untuk simpul dalam konfigurasi yang direplikasi, `ReplicationBytes` melaporkan jumlah byte yang dikirimkan oleh primer ke semua replikanya. Metrik ini mewakili beban tulis pada grup replikasi. Metrik ini berasal dari statistik `master_repl_offset` di Redis INFO. | Byte |
| `ReplicationLag` | Metrik ini hanya berlaku untuk simpul yang bekerja sebagai replika baca. Hal ini menunjukkan seberapa jauh ketinggalan, dalam detik, suatu replika dalam menerapkan perubahan dari simpul primer. Untuk Redis dengan versi mesin 5.0.6 dan seterusnya, ketertinggalan tersebut dapat diukur dalam milidetik. | Detik |
| `SaveInProgress` | Metrik biner ini menghasilkan 1 jika penyimpanan latar belakang (bercabang atau tak bercabang) sedang berlangsung, dan 0 jika sebaliknya. Proses simpan di latar belakang biasanya digunakan selama snapshot dan sinkronisasi. Operasi ini dapat menyebabkan kinerja menurun. Menggunakan metrik `SaveInProgress`, Anda dapat mendiagnosis apakah kinerja yang menurun ini disebabkan oleh proses penyimpanan di latar belakang. Metrik ini berasal dari statistik `rdb_bgsave_in_progress` di Redis INFO. | Boolean |

| Metrik | Deskripsi | Unit |
| --- | --- | --- |
| `TrafficManagementActive` | Menunjukkan apakah ElastiCache for Redis secara aktif mengelola lalu lintas dengan menyesuaikan lalu lintas yang dialokasikan ke perintah, pemantauan, atau replikasi yang masuk. Lalu lintas dikelola ketika lebih banyak perintah dikirim ke simpul daripada yang dapat diproses oleh Redis dan digunakan untuk menjaga stabilitas dan pengoperasian mesin yang optimal. Setiap titik data 1 dapat menunjukkan bahwa simpul diskalakan untuk beban kerja yang disediakan.<br><br>**Note**<br>Jika metrik ini tetap aktif, evaluasi cluster klaster memutuskan apakah penaikan skala atau penskalaan keluar diperlukan. Metrik terkait termasuk `NetworkBandwidthOutAllowanceExceeded` dan `EngineCPUUtilization` . | Boolean |

Ketersediaan EngineCPUUtilization

Wilayah AWS yang tercantum berikut ini tersedia di semua jenis simpul yang didukung.

| Wilayah | Nama wilayah |
| --- | --- |
| `us-east-2` | AS Timur (Ohio) |
| `us-east-1` | AS Timur (Virginia Utara) |
| `us-west-1` | AS Barat (California Utara) |
| `us-west-2` | AS Barat (Oregon) |

| Wilayah | Nama wilayah |
| --- | --- |
| ap-northeast-1 | Asia Pasifik (Tokyo) |
| ap-northeast-2 | Asia Pasifik (Seoul) |
| ap-northeast-3 | Asia Pasifik (Osaka) |
| ap-east-1 | Asia Pasifik (Hong Kong) |
| ap-south-1 | Asia Pasifik (Mumbai) |
| ap-southeast-1 | Asia Pasifik (Singapura) |
| ap-southeast-2 | Asia Pasifik (Sydney) |
| ap-southeast-3 | Asia Pasifik (Jakarta) |
| ca-central-1 | Kanada (Pusat) |
| cn-north-1 | Tiongkok (Beijing) |
| cn-northwest-2 | Tiongkok (Ningxia) |
| me-south-1 | Timur Tengah (Bahrain) |
| eu-central-1 | Eropa (Frankfurt) |
| eu-west-1 | Eropa (Irlandia) |
| eu-west-2 | Eropa (London) |
| eu-west-3 | EU (Paris) |
| eu-south-1 | Eropa (Milan) |
| af-south-1 | Afrika (Cape Town) |
| eu-north-1 | Eropa (Stockholm) |
| sa-east-1 | Amerika Selatan (Sao Paulo) |

| Wilayah | Nama wilayah |
| --- | --- |
| us-gov-west-1 | AWS GovCloud (US-West) |
| us-gov-east-1 | AWS GovCloud (US-East) |

Berikut ini adalah kumpulan jenis perintah tertentu, yang berasal dari info commandstats. Bagian commandstats menyediakan statistik berdasarkan jenis perintah, termasuk jumlah panggilan, jumlah waktu CPU yang dikonsumsi oleh perintah ini, dan CPU rata-rata yang dikonsumsi per eksekusi perintah. Untuk setiap jenis perintah, baris berikut ditambahkan: `cmdstat_XXX: calls=XXX,usec=XXX,usec_per_call=XXX`.

Metrik latensi yang tercantum berikut dihitung menggunakan statistik commandstats dari INFO Redis. Metrik dihitung dengan cara berikut: `delta(usec)/delta(calls)` .`delta` dihitung sebagai diff dalam satu menit. Latensi didefinisikan sebagai waktu CPU yang dibutuhkan oleh ElastiCache untuk memproses perintah. Perhatikan bahwa untuk klaster yang menggunakan tingkatan data, waktu yang dibutuhkan untuk mengambil item dari SSD tidak termasuk dalam pengukuran ini.

Untuk daftar lengkap perintah yang tersedia, lihat perintah redis di dokumentasi Redis.

| Metrik | Deskripsi | Unit |
| --- | --- | --- |
| `ClusterBasedCmds` | Jumlah seluruh perintah yang berbasis klaster. Ini berasal dari `commandstats` statistik Redis dengan menjumlahkan semua perintah yang bekerja atas klaster (`cluster slot`,`cluster info`, dan sebagainya). | Jumlah |
| `ClusterBasedCmdsLatency` | Latensi perintah berbasis klaster. | Mikrodetik |
| `EvalBasedCmds` | Jumlah seluruh perintah untuk perintah berbasis eval. Ini berasal dari statistik `commandstats` Redis dengan menjumlahkan eval, evalsha. | Jumlah |
| `EvalBasedCmdsLatency` | Latensi perintah berbasis eval. | Mikrodetik |

| Metrik | Deskripsi | Unit |
| --- | --- | --- |
| GeoSpatialBasedCmds | Jumlah seluruh perintah untuk perintah berbasis geospasial. Ini berasal dari statistik commandstats Redis. Ini diperoleh dengan menjumlahkan semua perintah jenis geo: geoadd, geodist, geohash, geopos, georadius, dan georadiusbymember. | Jumlah |
| GeoSpatialBasedCmdsLatency | Latensi perintah berbasis geospasial. | Mikrodetik |
| GetTypeCmds | Jumlah seluruh perintah jenis read-only. Ini berasal dari statistik commandstats Redis dengan menjumlahkan semua jenis perintah read-only (get, hget, scard, lrange, dan sebagainya.) | Jumlah |
| GetTypeCmdsLatency | Latensi perintah baca. | Mikrodetik |
| HashBasedCmds | Jumlah seluruh perintah yang berbasis hash. Ini berasal dari statistik commandstats Redis dengan menjumlahkan semua perintah yang bekerja atas satu atau lebih hash (hget, hkeys, hvals, hdel, dan sebagainya). | Jumlah |
| HashBasedCmdsLatency | Latensi perintah berbasis hash. | Mikrodetik |
| HyperLogLogBasedCmds | Jumlah seluruh perintah berbasis HyperLog . Ini berasal dari statistik commandstats Redis dengan menjumlahkan semua jenis perintah pf (pfadd, pfcount, pfmerge, dan sebagainya.). | Jumlah |
| HyperLogLogBasedCmdsLatency | Latensi perintah berbasis HyperLogLog. | Mikrodetik |

| Metrik | Deskripsi | Unit |
| --- | --- | --- |
| JsonBasedCmds | Jumlah total perintah JSON, termasuk perintah baca dan tulis. Ini berasal dari statistik commandstats Redis dengan menjumlahkan semua perintah JSON yang bekerja atas kunci JSON. | Jumlah |
| JsonBasedCmdsLatency | Latensi semua perintah JSON, termasuk perintah baca dan tulis. | Mikrodetik |
| JsonBasedGetCmds | Jumlah seluruh perintah JSON hanya-baca. Ini berasal dari statistik commandstats Redis dengan menjumlahkan semua perintah baca JSON yang bekerja atas kunci JSON. | Jumlah |
| JsonBasedGetCmdsLatency | Latensi perintah hanya-baca JSON. | Mikrodetik |
| JsonBasedSetCmds | Jumlah seluruh perintah JSON jenis tulis. Ini berasal dari statistik commandstats Redis dengan menjumlahkan semua perintah tulis JSON yang bekerja atas kunci JSON. | Jumlah |
| JsonBasedSetCmdsLatency | Latensi perintah tulis JSON. | Mikrodetik |
| KeyBasedCmds | Jumlah seluruh perintah yang berbasis kunci. Ini berasal dari statistik commandstats Redis dengan menjumlahkan semua perintah yang bekerja atas satu atau lebih kunci di berbagai struktur data (del, expire, rename, dan sebagainya.). | Jumlah |
| KeyBasedCmdsLatency | Latensi perintah berbasis kunci. | Mikrodetik |

| Metrik | Deskripsi | Unit |
| --- | --- | --- |
| ListBasedCmds | Jumlah seluruh perintah yang berbasis daftar. Ini berasal dari statistik commandstats Redis dengan menjumlahkan semua perintah yang bekerja atas satu atau beberapa daftar (lindex, lrange, lpush, ltrim, dan sebagainya). | Jumlah |
| ListBasedCmdsLatency | Latensi perintah berbasis daftar. | Mikrodetik |
| NonKeyTypeCmds | Jumlah seluruh perintah yang tidak berbasis kunci. Ini berasal dari statistik commandstats Redis dengan menjumlahkan semua perintah yang tidak bekerja atas kunci, misalnya acl, dbsize atau info. | Jumlah |
| NonKeyTypeCmdsLatency | Latensi perintah berbasis kunci. | Mikrodetik |
| PubSubBasedCmds | Jumlah seluruh perintah untuk fungsionalitas pub/sub. Ini berasal dari statistik commandstats Redis dengan menjumlahkan semua perintah yang digunakan untuk fungsionalitas pub/sub: psubscribe, publish, pubsub, punsubscribe, ssubscribe, sunsubscribe, spublish, subscribe, dan unsubscribe. | Jumlah |
| PubSubBasedCmdsLatency | Latensi perintah berbasis pub/sub. | Mikrodetik |
| SetBasedCmds | Jumlah seluruh perintah yang berbasis set. Ini berasal dari statistik commandstats Redis dengan menjumlahkan semua perintah yang bekerja atas satu atau lebih set (scard, sdiff, sadd, sunion, dan sebagainya). | Jumlah |
| SetBasedCmdsLatency | Latensi perintah berbasis set. | Mikrodetik |

| Metrik | Deskripsi | Unit |
| --- | --- | --- |
| SetTypeCmds | Jumlah seluruh perintah jenis write. Ini berasal dari statistik commandstats Redis dengan menjumlahkan semua perintah jenis mutative yang bekerja pada data (set, hset, sadd, lpop, dan sebagainya.) | Jumlah |
| SetTypeCmdsLatency | Latensi perintah tulis. | Mikrodetik |
| SortedSetBasedCmds | Jumlah seluruh perintah yang diurutkan berbasis set. Ini berasal dari statistik commandstats Redis dengan menjumlahkan semua perintah yang bekerja atas satu atau beberapa set yang diurutkan (zcount, zrange, zrank, zadd, dan sebagainya). | Jumlah |
| SortedSetBasedCmdsLatency | Latensi perintah berbasis urutan. | Mikrodetik |
| StringBasedCmds | Jumlah seluruh perintah yang berbasis string. Ini berasal dari statistik commandstats Redis dengan menjumlahkan semua perintah yang bekerja atas satu atau beberapa string (strlen, setex, setrange, dan sebagainya). | Jumlah |
| StringBasedCmdsLatency | Latensi perintah berbasis string. | Mikrodetik |
| StreamBasedCmds | Jumlah seluruh perintah yang berbasis stream. Ini berasal dari statistik commandstats Redis dengan menjumlahkan semua perintah yang bekerja atas satu atau beberapa tipe data stream (xrange, xlen, xadd, xdel, dan sebagainya). | Jumlah |
| StreamBasedCmdsLatency | Latensi perintah berbasis stream. | Mikrodetik |

## Metrik Apa Yang Harus Dipantau?

Metrik CloudWatch berikut menawarkan wawasan yang bermanfaat tentang performa ElastiCache. Pada umumnya, kami merekomendasikan Anda untuk mengatur alarm CloudWatch untuk metrik ini agar Anda dapat mengambil tindakan korektif sebelum masalah performa terjadi.

Metrik yang Perlu Dipantau

- CPUUtilization
- EngineCPUUtilization
- SwapUsage
- Pengosongan
- CurrConnections
- Memori
- Jaringan
- Latensi
- Replikasi
- Manajemen Lalu Lintas

CPUUtilization

Ini adalah metrik tingkat host yang dilaporkan sebagai persentase. Untuk informasi selengkapnya, lihat Metrik Tingkat Host.

Untuk jenis simpul yang lebih kecil dengan 2vCPU atau kurang, gunakan metrik `CPUUtilization` untuk memantau beban kerja Anda.

Secara umum, sebaiknya atur ambang batas Anda sebesar 90% dari CPU yang tersedia. Karena Redis bersifat single-threaded, nilai ambang batas yang sebenarnya harus dihitung sebagai bagian kecil dari seluruh kapasitas simpul ini. Sebagai contoh, misalkan Anda menggunakan jenis simpul yang memiliki dua core. Dalam hal ini, ambang batas untuk CPUUtilization akan menjadi 90/2, atau 45%.

Anda akan perlu menentukan ambang batas Anda sendiri, berdasarkan jumlah core pada simpul cache yang Anda gunakan. Jika Anda melampaui ambang batas ini, dan beban kerja utama Anda adalah dari permintaan baca, skalakan keluar klaster cache Anda dengan menambahkan replika baca. Jika beban kerja utama dari permintaan tulis, bergantung pada konfigurasi klaster Anda, sebaiknya Anda:

- Klaster Redis (mode klaster dinonaktifkan): menaikkan skala dengan menggunakan tipe instans cache yang lebih besar.
- Klaster Redis (mode klaster diaktifkan): menambahkan lebih banyak serpihan untuk mendistribusikan beban kerja tulis ke lebih banyak simpul primer.

> **Tip**
>
> Alih-alih menggunakan metrik `CPUUtilization` Tingkat Host, pengguna Redis mungkin dapat menggunakan metrik `EngineCPUUtilization` Redis, yang melaporkan persentase penggunaan di core mesin Redis. Untuk melihat apakah metrik ini tersedia di simpul Anda dan untuk informasi selengkapnya, lihat Metrik untuk Redis.

Untuk jenis simpul yang lebih besar dengan 4vCPU atau lebih, sebaiknya gunakan metrik `EngineCPUUtilization`, yang melaporkan persentase penggunaan di core mesin Redis. Untuk melihat apakah metrik ini tersedia di simpul Anda dan untuk informasi selengkapnya, lihat Metrik untuk Redis.

EngineCPUUtilization

Untuk jenis simpul yang lebih besar dengan 4vCPU atau lebih, sebaiknya gunakan metrik `EngineCPUUtilization`, yang melaporkan persentase penggunaan di core mesin Redis. Untuk melihat apakah metrik ini tersedia di simpul Anda dan untuk informasi selengkapnya, lihat Metrik untuk Redis.

Untuk informasi selengkapnya, lihat bagian CPU di Praktik terbaik pemantauan dengan Amazon ElastiCache for Redis menggunakan Amazon CloudWatch.

SwapUsage

Ini adalah metrik tingkat host yang dilaporkan dalam byte. Untuk informasi selengkapnya, lihat Metrik Tingkat Host.

Metrik `FreeableMemory` CloudWatch yang mendekati 0 (yaitu, di bawah 100MB) atau metrik `SwapUsage` yang lebih besar dari metrik `FreeableMemory` menunjukkan simpul berada dalam tekanan memori. Jika tidak, lihat topik berikut:

- Memastikan bahwa Anda memiliki cukup memori untuk membuat snapshot Redis

- Mengelola Memori Terpesan

Pengosongan

Ini adalah metrik mesin cache. Sebaiknya tentukan ambang batas alarm Anda sendiri untuk metrik ini berdasarkan kebutuhan aplikasi Anda.

CurrConnections

Ini adalah metrik mesin cache. Sebaiknya tentukan ambang batas alarm Anda sendiri untuk metrik ini berdasarkan kebutuhan aplikasi Anda.

Peningkatan jumlah CurrConnections mungkin menunjukkan masalah di aplikasi Anda; Anda perlu menyelidiki perilaku aplikasi untuk mengatasi masalah ini.

Untuk informasi selengkapnya, lihat bagian Koneksi di Praktik terbaik pemantauan dengan Amazon ElastiCache for Redis menggunakan Amazon CloudWatch.

Memori

Memori adalah aspek utama dari Redis. Memahami pemanfaatan memori dari klaster Anda diperlukan untuk menghindari kehilangan data dan mengakomodasi pertumbuhan set data Anda pada masa mendatang. Statistik tentang pemanfaatan memori dari simpul tersedia di bagian memori pada perintah INFO Redis.

Untuk informasi selengkapnya, lihat bagian Memori di Praktik terbaik pemantauan dengan Amazon ElastiCache for Redis menggunakan Amazon CloudWatch.

Jaringan

Salah satu faktor penentu untuk kapasitas bandwidth jaringan dari klaster Anda adalah jenis simpul yang telah Anda pilih. Untuk informasi selengkapnya tentang kapasitas jaringan simpul Anda, lihat Harga Amazon ElastiCache.

Untuk informasi selengkapnya lihat bagian Jaringan di Praktik terbaik pemantauan dengan Amazon ElastiCache for Redis menggunakan Amazon CloudWatch.

Latensi

Anda dapat mengukur latensi perintah dengan serangkaian metrik CloudWatch yang menyediakan latensi gabungan per struktur data. Metrik latensi ini dihitung menggunakan statistik `commandstats` dari perintah INFO Redis.

Untuk informasi selengkapnya, lihat bagian Latensi di Praktik terbaik pemantauan dengan Amazon ElastiCache for Redis menggunakan Amazon CloudWatch.

Replikasi

Volume data yang direplikasi akan terlihat melalui metrik `ReplicationBytes`. Metrik ini tidak memberikan wawasan mengenai kesehatan replikasi, meskipun mewakili beban tulis pada grup replikasi. Untuk tujuan ini, Anda dapat menggunakan metrik `ReplicationLag`.

Untuk informasi selengkapnya, lihat bagian Replikasi di Praktik terbaik pemantauan dengan Amazon ElastiCache for Redis menggunakan Amazon CloudWatch.

Manajemen Lalu Lintas

ElastiCache for Redis secara otomatis mengelola lalu lintas terhadap simpul ketika lebih banyak perintah masuk dikirim ke simpul daripada yang dapat diproses oleh Redis. Hal ini dilakukan untuk menjaga operasi dan stabilitas mesin yang optimal.

Ketika lalu lintas dikelola secara aktif pada simpul, metrik `TrafficManagementActive` akan memancarkan titik data 1. Hal ini menunjukkan bahwa simpul mungkin kurang diskalakan untuk beban kerja yang disediakan. Jika metrik ini tetap 1 untuk jangka waktu yang lama, evaluasi klaster untuk memutuskan apakah penaikan skala atau penskalaan ke luar diperlukan.

Untuk informasi selengkapnya, lihat metrik `TrafficManagementActive` di halaman Metrik.

## Memilih Statistik dan Periode Metrik

Meskipun CloudWatch memungkinkan Anda memilih statistik dan periode apa pun untuk setiap metrik, tidak semua kombinasi akan berguna. Misalnya, statistik Rata-Rata, Minimum, dan Maksimum untuk pemanfaatan CPU akan bermanfaat, tetapi statistik Jumlah tidak.

Semua sampel ElastiCache dipublikasikan selama 60 detik untuk setiap simpul cache. Untuk periode 60 detik, metrik simpul cache hanya akan berisi satu sampel tunggal.

Untuk informasi lebih lanjut tentang cara mengambil metrik untuk simpul cache, lihat Memantau Metrik Klaster dan Simpul CloudWatch.

## Memantau Metrik Klaster dan Simpul CloudWatch

ElastiCache dan CloudWatch terintegrasi agar Anda dapat mengumpulkan berbagai metrik. Anda dapat memantau metrik ini menggunakan CloudWatch.

> **Note**
>
> Contoh berikut memerlukan alat baris perintah CloudWatch. Untuk informasi selengkapnya tentang CloudWatch dan untuk mengunduh alat developer, lihat halaman produk CloudWatch.

Prosedur berikut menunjukkan cara menggunakan CloudWatch guna mengumpulkan statistik ruang penyimpanan untuk klaster cache selama satu jam terakhir.

> **Note**
>
> Nilai `StartTime` dan `EndTime` yang diberikan pada contoh di bawah ini adalah untuk tujuan ilustrasi. Anda harus mengganti nilai waktu mulai dan akhir yang sesuai untuk simpul cache Anda.

Untuk informasi tentang batas ElastiCache, lihat Kuota Layanan AWS untuk ElastiCache.

Memantau Metrik Klaster dan Simpul CloudWatch (Konsol)

Untuk mengumpulkan statistik pemanfaatan CPU untuk klaster cache

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Pilih simpul cache yang metriknya ingin Anda lihat.

   > **Note**
   >
   > Memilih lebih dari 20 simpul akan menonaktifkan tampilan metrik pada konsol.

   a. Di halaman Klaster Cache pada Konsol Manajemen AWS, klik nama satu atau beberapa klaster cache.

      Halaman detail untuk klaster cache akan muncul.

   b. Klik tab Simpul di bagian atas jendela.

   c. Di tab Simpul pada jendela detail, pilih simpul cache yang ingin Anda lihat metriknya.

      Daftar Metrik CloudWatch yang tersedia muncul di bagian bawah jendela konsol.

   d. Klik metrik Pemanfaatan CPU.

      Konsol CloudWatch akan terbuka, yang menampilkan metrik pilihan Anda. Anda dapat menggunakan kotak daftar drop-down Statistik dan Periode, serta tab Rentang waktu untuk mengubah metrik yang ditampilkan.

Memantau Metrik Klaster dan Simpul CloudWatch menggunakan CloudWatch CLI

Untuk mengumpulkan statistik pemanfaatan CPU untuk klaster cache

- Untuk Linux, macOS, atau Unix:

```
aws cloudwatch get-metric-statistics \
    --namespace AWS/ElastiCache \
    --metric-name CPUUtilization \
    --dimensions='[{"Name":"CacheClusterId","Value":"test"},
{"Name":"CacheNodeId","Value":"0001"}]' \
    --statistics=Average \
    --start-time 2018-07-05T00:00:00 \
    --end-time 2018-07-06T00:00:00 \
    --period=3600
```

Untuk Windows:

```
aws cloudwatch get-metric-statistics ^
    --namespace AWS/ElastiCache ^
    --metric-name CPUUtilization ^
    --dimensions='[{"Name":"CacheClusterId","Value":"test"},
{"Name":"CacheNodeId","Value":"0001"}]' ^
    --statistics=Average ^
    --start-time 2018-07-05T00:00:00 ^
    --end-time 2018-07-06T00:00:00 ^
    --period=3600
```

Memantau Metrik Klaster dan Simpul CloudWatch menggunakan API CloudWatch

Untuk mengumpulkan statistik pemanfaatan CPU untuk klaster cache

- Panggil API CloudWatch `GetMetricStatistics` dengan parameter berikut (perhatikan bahwa waktu mulai dan akhir ditampilkan sebagai contoh saja; Anda perlu mengganti waktu awal dan akhir yang sesuai dengan waktu Anda sendiri):
  - `Statistics.member.1=Average`
  - `Namespace=AWS/ElastiCache`
  - `StartTime=2013-07-05T00:00:00`
  - `EndTime=2013-07-06T00:00:00`
  - `Period=60`
  - `MeasureName=CPUUtilization`
  - `Dimensions=CacheClusterId=mycachecluster,CacheNodeId=0002`

  Example

```
http://monitoring.amazonaws.com/
    ?Action=GetMetricStatistics
    &SignatureVersion=4
    &Version=2014-12-01
    &StartTime=2018-07-05T00:00:00
    &EndTime=2018-07-06T23:59:00
```

```
&Period=3600
&Statistics.member.1=Average
&Dimensions.member.1="CacheClusterId=mycachecluster"
&Dimensions.member.2="CacheNodeId=0002"
&Namespace=&AWS;/ElastiCache
&MeasureName=CPUUtilization
&Timestamp=2018-07-07T17%3A48%3A21.746Z
&AWS;AccessKeyId=<&AWS; Access Key ID>
&Signature=<Signature>
```

# Pemantauan Amazon SNS dari peristiwa ElastiCache

Saat peristiwa penting terjadi untuk klaster, ElastiCache mengirimkan notifikasi ke topik Amazon SNS tertentu. Contohnya meliputi kegagalan menambahkan simpul, keberhasilan menambahkan simpul, perubahan grup keamanan, dan lainnya. Dengan memantau peristiwa penting, Anda dapat mengetahui status klaster terbaru Anda dan dapat mengambil tindakan korektif sesuai peristiwa tersebut.

Topik

- Mengelola notifikasi ElastiCache Amazon SNS
- Melihat peristiwa ElastiCache
- Notifikasi Peristiwa dan Amazon SNS

## Mengelola notifikasi ElastiCache Amazon SNS

Anda dapat mengkonfigurasi ElastiCache untuk mengirim notifikasi untuk peristiwa klaster penting menggunakan Amazon Simple Notification Service (Amazon SNS). Dalam contoh ini, Anda akan mengonfigurasi klaster dengan Amazon Resource Name (ARN) dari topik Amazon SNS untuk menerima notifikasi.

> **Note**
>
> Topik ini mengasumsikan bahwa Anda telah mendaftar ke Amazon SNS dan telah mengatur serta berlangganan topik Amazon SNS. Untuk informasi lebih lanjut tentang cara melakukannya, lihat Panduan Developer Amazon Simple Notification Service.

### Menambahkan topik Amazon SNS

Bagian berikut menunjukkan cara menambahkan topik Amazon SNS menggunakan konsol AWS, AWS CLI, atau API ElastiCache.

#### Menambahkan topik Amazon SNS (Konsol)

Prosedur berikut menunjukkan cara menambahkan topik Amazon SNS untuk klaster. Untuk menambahkan topik Amazon SNS untuk grup replikasi, pada langkah 2, alih-alih memilih klaster, pilih grup replikasi, lalu ikuti langkah yang sama tersisa.

> **Note**
>
> Proses ini juga dapat digunakan untuk mengubah topik Amazon SNS.

Untuk menambahkan atau mengubah topik Amazon SNS untuk klaster (Konsol)

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Di  Klaster, pilih laster yang ingin Anda tambahkan atau ubah ARN topik Amazon SNS-nya.
3. Pilih Ubah.
4. Di Ubah Klaster di bagian Topik untuk Notifikasi SNS, pilih topik SNS yang ingin Anda tambahkan, atau pilih Masukan ARN manual dan ketik ARN topik Amazon SNS.
5. Pilih Ubah.

#### Menambahkan topik Amazon SNS (AWS CLI)

Untuk menambahkan atau mengubah topik Amazon SNS untuk klaster, gunakan perintah AWS CLI `modify-cache-cluster`.

Contoh kode berikut menambahkan arn topik Amazon SNS ke my-cluster.

Untuk Linux, macOS, atau Unix:

```
aws elasticache modify-cache-cluster \
    --cache-cluster-id my-cluster \
    --notification-topic-arn arn:aws:sns:us-
west-2:123456789xxx:ElastiCacheNotifications
```

Untuk Windows:

```
aws elasticache modify-cache-cluster ^
    --cache-cluster-id my-cluster ^
    --notification-topic-arn arn:aws:sns:us-west-2:123456789xx:ElastiCacheNotifications
```

Untuk informasi selengkapnya, lihat modify-cache-cluster.

Menambahkan topik Amazon SNS (ElastiCache API)

Untuk menambah atau mengubah topik Amazon SNS untuk klaster, panggil tindakan `ModifyCacheCluster` dengan parameter berikut:

- `CacheClusterId=my-cluster`
- `TopicArn=arn%3Aaws%3Asns%3Aus-west-2%3A565419523791%3AElastiCacheNotifications`

Example

```
https://elasticache.amazon.com/
    ?Action=ModifyCacheCluster
    &ApplyImmediately=false
    &CacheClusterId=my-cluster
    &NotificationTopicArn=arn%3Aaws%3Asns%3Aus-
west-2%3A565419523791%3AElastiCacheNotifications
    &Version=2014-12-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20141201T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

Untuk informasi selengkapnya, lihat ModifyCacheCluster.

### Mengaktifkan dan nonaktifkan notifikasi Amazon SNS

Anda dapat mengaktifkan atau menonaktifkan notifikasi untuk klaster. Prosedur berikut menunjukkan cara nonaktifkan notifikasi Amazon SNS.

#### Mengaktifkan dan nonaktifkan notifikasi Amazon SNS (Konsol)

Untuk menonaktifkan notifikasi Amazon SNS menggunakan AWS Management Console

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Untuk melihat daftar klaster Anda yang menjalankan Redis, pada panel navigasi, pilih Redis.
3. Pilih kotak di sebelah kiri klaster yang ingin diubah notifikasinya.
4. Pilih Ubah.
5. Di Ubah Klaster di bagian Topik untuk Notifikasi SNS, pilih Nonaktifkan Notifikasi.
6. Pilih Ubah.

#### Mengaktifkan dan nonaktifkan notifikasi Amazon SNS (AWS CLI)

Untuk nonaktifkan notifikasi Amazon SNS, gunakan perintah `modify-cache-cluster` dengan parameter berikut:

Untuk Linux, macOS, atau Unix:

```
aws elasticache modify-cache-cluster \
    --cache-cluster-id my-cluster \
    --notification-topic-status inactive
```

Untuk Windows:

```
aws elasticache modify-cache-cluster ^
    --cache-cluster-id my-cluster ^
    --notification-topic-status inactive
```

#### Mengaktifkan dan nonaktifkan notifikasi Amazon SNS (API ElastiCache)

Untuk nonaktifkan notifikasi Amazon SNS, panggil tindakan `ModifyCacheCluster` dengan parameter berikut:

- `CacheClusterId=my-cluster`
- `NotificationTopicStatus=inactive`

Panggilan ini menghasilkan output seperti yang berikut ini:

Example

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=ModifyCacheCluster
    &ApplyImmediately=false
    &CacheClusterId=my-cluster
    &NotificationTopicStatus=inactive
    &Version=2014-12-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20141201T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

## Melihat peristiwa ElastiCache

ElastiCache membuat log acara yang berhubungan dengan instans klaster, grup keamanan, dan grup parameter Anda. Informasi ini mencakup tanggal dan waktu peristiwa, nama sumber dan jenis sumber peristiwa, serta deskripsi dari peristiwa tersebut. Anda dapat dengan mudah menarik peristiwa dari log menggunakan konsol ElastiCache, perintah AWS CLI `describe-events` , atau tindakan API ElastiCache `DescribeEvents`.

Prosedur berikut menunjukkan kepada Anda cara untuk melihat semua peristiwa ElastiCache dalam 24 jam terakhir (1440 menit).

### Melihat peristiwa ElastiCache (Konsol)

Prosedur berikut menampilkan peristiwa menggunakan konsol ElastiCache.

Untuk melihat peristiwa menggunakan konsol ElastiCache:

1. Masuk ke AWS Management Console dan buka konsol ElastiCache di https://console.aws.amazon.com/elasticache/.
2. Untuk melihat daftar dari semua peristiwa yang tersedia, pada panel navigasi, pilih Peristiwa.

   Di layar Peristiwa setiap baris daftar mewakili satu peristiwa dan menampilkan sumber peristiwa, jenis peristiwa (cache-cluster, cache-parameter-grup, cache-security-group, atau cache-subnet-group), waktu GMT peristiwa, serta deskripsi dari peristiwa.

   Dengan menggunakan Filter, Anda dapat menentukan apakah Anda ingin melihat semua peristiwa, atau hanya peristiwa dari jenis tertentu dalam daftar peristiwa.

### Melihat peristiwa ElastiCache (AWS CLI)

Untuk menghasilkan daftar dari peristiwa ElastiCache menggunakan AWS CLI, gunakan perintah `describe-events`. Anda dapat menggunakan parameter opsional untuk mengontrol jenis peristiwa yang tercantum, jangka waktu peristiwa yang dicantumkan, jumlah maksimum peristiwa yang akan dicantumkan, dan lainnya.

Kode berikut mencantumkan hingga 40 peristiwa klaster cache.

```
aws elasticache describe-events --source-type cache-cluster --max-items 40
```

Kode berikut mencantumkan semua peristiwa selama 24 jam terakhir (1.440 menit).

```
aws elasticache describe-events --source-type cache-cluster --duration 1440
```

Output dari perintah `describe-events` terlihat seperti berikut ini.

```
aws elasticache describe-events --source-type cache-cluster --max-items 40
{
    "Events": [
        {
            "SourceIdentifier": "my-mem-cluster",
            "SourceType": "cache-cluster",
            "Message": "Finished modifying number of nodes from 1 to 3",
            "Date": "2020-06-09T02:01:21.772Z"
        },
        {
            "SourceIdentifier": "my-mem-cluster",
            "SourceType": "cache-cluster",
            "Message": "Added cache node 0002 in availability zone us-west-2a",
            "Date": "2020-06-09T02:01:21.716Z"
        },
        {
            "SourceIdentifier": "my-mem-cluster",
            "SourceType": "cache-cluster",
            "Message": "Added cache node 0003 in availability zone us-west-2a",
            "Date": "2020-06-09T02:01:21.706Z"
        },
        {
            "SourceIdentifier": "my-mem-cluster",
            "SourceType": "cache-cluster",
            "Message": "Increasing number of requested nodes",
            "Date": "2020-06-09T01:58:34.178Z"
        },
        {
            "SourceIdentifier": "mycluster-0003-004",
            "SourceType": "cache-cluster",
            "Message": "Added cache node 0001 in availability zone us-west-2c",
            "Date": "2020-06-09T01:51:14.120Z"
        },
        {
            "SourceIdentifier": "mycluster-0003-004",
            "SourceType": "cache-cluster",
            "Message": "This cache cluster does not support persistence (ex:
 'appendonly').  Please use a different instance type to enable persistence.",
            "Date": "2020-06-09T01:51:14.095Z"
```

```
        },
        {
            "SourceIdentifier": "mycluster-0003-004",
            "SourceType": "cache-cluster",
            "Message": "Cache cluster created",
            "Date": "2020-06-09T01:51:14.094Z"
        },
        {
            "SourceIdentifier": "mycluster-0001-005",
            "SourceType": "cache-cluster",
            "Message": "Added cache node 0001 in availability zone us-west-2b",
            "Date": "2020-06-09T01:42:55.603Z"
        },
        {
            "SourceIdentifier": "mycluster-0001-005",
            "SourceType": "cache-cluster",
            "Message": "This cache cluster does not support persistence (ex:
 'appendonly').  Please use a different instance type to enable persistence.",
            "Date": "2020-06-09T01:42:55.576Z"
        },
        {
            "SourceIdentifier": "mycluster-0001-005",
            "SourceType": "cache-cluster",
            "Message": "Cache cluster created",
            "Date": "2020-06-09T01:42:55.574Z"
        },
        {
            "SourceIdentifier": "mycluster-0001-004",
            "SourceType": "cache-cluster",
            "Message": "Added cache node 0001 in availability zone us-west-2b",
            "Date": "2020-06-09T01:28:40.798Z"
        },
        {
            "SourceIdentifier": "mycluster-0001-004",
            "SourceType": "cache-cluster",
            "Message": "This cache cluster does not support persistence (ex:
 'appendonly').  Please use a different instance type to enable persistence.",
            "Date": "2020-06-09T01:28:40.775Z"
        },
        {
            "SourceIdentifier": "mycluster-0001-004",
            "SourceType": "cache-cluster",
            "Message": "Cache cluster created",
            "Date": "2020-06-09T01:28:40.773Z"
```

```
        }
    ]
}
```

Untuk informasi selengkapnya, seperti parameter yang tersedia dan nilai parameter yang diizinkan, lihat `describe-events`.

Melihat peristiwa ElastiCache (API ElastiCache)

Untuk menghasilkan daftar dari peristiwa ElastiCache menggunakan API ElastiCache, gunakan tindakan `DescribeEvents`. Anda dapat menggunakan parameter opsional untuk mengontrol jenis peristiwa yang tercantum, jangka waktu peristiwa yang dicantumkan, jumlah maksimum peristiwa yang akan dicantumkan, dan lainnya.

Kode berikut mencantumkan 40 peristiwa klaster cache terbaru.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeEvents
   &MaxRecords=40
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &SourceType=cache-cluster
   &Timestamp=20150202T192317Z
   &Version=2015-02-02
   &X-Amz-Credential=<credential>
```

Kode berikut mencantumkan peristiwa klaster cache selama 24 jam terakhir (1.440 menit).

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeEvents
   &Duration=1440
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &SourceType=cache-cluster
   &Timestamp=20150202T192317Z
   &Version=2015-02-02
   &X-Amz-Credential=<credential>
```

Perintah di atas akan menghasilkan output yang serupa dengan berikut ini.

```
<DescribeEventsResponse xmlns="http://elasticache.amazonaws.com/doc/2015-02-02/">
```

```
    <DescribeEventsResult>
        <Events>
            <Event>
                <Message>Cache cluster created</Message>
                <SourceType>cache-cluster</SourceType>
                <Date>2015-02-02T18:22:18.202Z</Date>
                <SourceIdentifier>mem01</SourceIdentifier>
            </Event>

 (...output omitted...)

        </Events>
    </DescribeEventsResult>
    <ResponseMetadata>
        <RequestId>e21c81b4-b9cd-11e3-8a16-7978bb24ffdf</RequestId>
    </ResponseMetadata>
</DescribeEventsResponse>
```

Untuk informasi selengkapnya, seperti parameter yang tersedia dan nilai parameter yang diizinkan, lihat `DescribeEvents`.

## Notifikasi Peristiwa dan Amazon SNS

ElastiCache dapat mempublikasikan olahpesan menggunakan Amazon Simple Notification Service (SNS) ketika peristiwa penting terjadi pada klaster cache. Fitur ini dapat digunakan untuk menyegarkan daftar server pada mesin klien yang terhubung ke setiap titik akhir simpul cache pada klaster cache.

> **Note**
>
> Untuk informasi selengkapnya Amazon Simple Notification Service (SNS), termasuk informasi mengenai harga dan tautan ke dokumentasi Amazon SNS, lihat Halaman produk Amazon SNS.

Notifikasi dipublikasikan ke topik Amazon SNS tertentu. Berikut ini adalah persyaratan untuk notifikasi:

- Hanya satu topik yang dapat dikonfigurasi untuk notifikasi ElastiCache.
- Akun AWS yang memiliki topik Amazon SNS harus merupakan akun yang sama yang memiliki klaster cache yang notifikasinya diaktifkan.

- Topik Amazon SNS yang Anda publikasikan tidak dapat dienkripsi.

> **Note**
>
> Dimungkinkan untuk melampirkan topik Amazon SNS yang terenkripsi (diam) ke klaster. Namun, status topik dari konsol ElastiCache akan ditampilkan tidak aktif, yang secara efektif memisahkan topik itu dari klaster ketika ElastiCache mendorong pesan ke topik tersebut.

- Topik Amazon SNS harus berada di Wilayah yang sama dengan klaster ElastiCache.

Peristiwa ElastiCache

Peristiwa ElastiCache berikut memicu notifikasi Amazon SNS. Untuk informasi tentang detail peristiwa, lihat Melihat peristiwa ElastiCache.

| Nama Peristiwa | Pesan | Deskripsi |
|---|---|---|
| ElastiCache:AddCacheNodeComplete | `ElastiCache:AddCacheNodeComplete : cache-cluster` | Simpul cache telah ditambahkan ke klaster cache dan siap untuk digunakan. |
| ElastiCache:AddCacheNodeFailed karena alamat IP yang bebas tidak mencukupi | `ElastiCache:AddCacheNodeFailed : cluster-name` | Simpul cache tidak dapat ditambahkan karena alamat IP yang tersedia tidak cukup. |
| ElastiCache:CacheClusterParametersChanged | `ElastiCache:CacheClusterParametersChanged : cluster-name` | Satu atau beberapa parameter klaster cache telah berubah. |
| ElastiCache:CacheClusterProvisioningComplete | `ElastiCache:CacheClusterProvisioningComplete cluster-name-0001-005` | Penyediaan klaster cache selesai, dan simpul cache dalam klaster cache siap untuk digunakan. |

| Nama Peristiwa | Pesan | Deskripsi |
| --- | --- | --- |
| ElastiCache:CacheClusterProvisioningFailed karena status jaringan yang tidak kompatibel | `ElastiCache:CacheClusterProvisioningFailed : cluster-name` | Percobaan dilakukan untuk meluncurkan klaster cache baru ke cloud privat virtual (VPC) yang tidak ada. |
| ElastiCache:CacheClusterScalingComplete | `CacheClusterScalingComplete : cluster-name` | Penskalaan untuk klaster cache berhasil diselesaikan. |
| ElastiCache:CacheClusterScalingFailed | ElastiCache:CacheClusterScalingFailed : *nama klaster* | Operasi peningkatan skala pada klaster cache gagal. |
| ElastiCache:CacheClusterSecurityGroupModified | `ElastiCache:CacheClusterSecurityGroupModified : cluster-name` | Salah satu peristiwa berikut telah terjadi:<br>• Daftar grup keamanan cache yang diotorisasi untuk klaster cache telah berubah.<br>• Satu atau beberapa grup keamanan EC2 telah diotorisasi pada grup keamanan cache yang terkait dengan klaster cache.<br>• Satu atau beberapa grup keamanan EC2 baru telah dicabut otorisasinya dari salah satu grup keamanan cache yang terkait dengan klaster cache. |

| Nama Peristiwa | Pesan | Deskripsi |
| --- | --- | --- |
| ElastiCache:CacheNodeReplaceStarted | `ElastiCache:CacheNodeReplaceStarted : cluster-name` | ElastiCache telah mendeteksi bahwa host yang menjalankan simpul cache mengalami penurunan performa atau tidak dapat dijangkau dan telah mulai mengganti simpul cache.<br><br>**Note**<br>Entri DNS untuk simpul cache yang diganti tidak berubah.<br><br>Pada kebanyakan kasus, Anda tidak perlu menyegarkan daftar server untuk klien Anda ketika peristiwa ini terjadi. Namun, beberapa pustaka klien cache mungkin berhenti menggunakan simpul cache bahkan setelah ElastiCache mengganti simpul cache; dalam hal ini, aplikasi harus menyegarkan daftar server ketika peristiwa ini terjadi. |

| Nama Peristiwa | Pesan | Deskripsi |
| --- | --- | --- |
| ElastiCache:CacheNodeReplaceComplete | `ElastiCache:CacheNodeReplaceComplete : cluster-name` | ElastiCache mendeteksi bahwa host yang menjalankan simpul cache terdegradasi atau tidak terjangkau dan telah selesai mengganti simpul cache.<br><br>**Note**<br>Entri DNS untuk simpul cache yang diganti tidak berubah.<br><br>Pada kebanyakan kasus, Anda tidak perlu menyegarkan daftar server untuk klien Anda ketika peristiwa ini terjadi. Namun, beberapa pustaka klien cache mungkin berhenti menggunakan simpul cache bahkan setelah ElastiCache mengganti simpul cache; dalam hal ini, aplikasi harus menyegarkan daftar server ketika peristiwa ini terjadi. |
| ElastiCache:CacheNodesRebooted | `ElastiCache:CacheNodesRebooted : cluster-name` | Satu atau beberapa simpul cache telah di-boot ulang.<br><br>Pesan (Memcached): `"Cache node %s shutdown"` Kemudian pesan kedua: `"Cache node %s restarted"` |

| Nama Peristiwa | Pesan | Deskripsi |
| --- | --- | --- |
| ElastiCache:CertificateRenewalComplete (hanya Redis) | `ElastiCache:CertificateRenewalComplete` | Sertifikat Amazon CA berhasil diperbarui. |
| ElastiCache:CreateReplicationGroupComplete | `ElastiCache:CreateReplicationGroupComplete : cluster-name` | Grup replikasi berhasil dibuat. |
| ElastiCache:DeleteCacheClusterComplete | `ElastiCache:DeleteCacheClusterComplete : cluster-name` | Penghapusan klaster cache dan semua simpul cache terkait telah selesai. |
| ElastiCache:FailoverComplete (hanya Redis) | `ElastiCache:FailoverComplete : mycluster` | Failover ke simpul replika telah berhasil. |
| ElastiCache:ReplicationGroupIncreaseReplicaCountFinished | `ElastiCache:ReplicationGroupIncreaseReplicaCountFinished : cluster-name-0001-005` | Jumlah replika dalam klaster telah meningkat. |
| ElastiCache:ReplicationGroupIncreaseReplicaCountStarted | `ElastiCache:ReplicationGroupIncreaseReplicaCountStarted : cluster-name-0003-004` | Proses penambahan replika ke klaster Anda telah dimulai. |
| ElastiCache:NodeReplacementCanceled | `ElastiCache:NodeReplacementCanceled : cluster-name` | Sebuah simpul di klaster Anda yang dijadwalkan akan diganti tidak lagi dijadwalkan akan diganti. |

| Nama Peristiwa | Pesan | Deskripsi |
| --- | --- | --- |
| ElastiCache:NodeReplacementRescheduled | `ElastiCache:NodeReplacementRescheduled : cluster-name` | Sebuah simpul di klaster Anda yang sebelumnya dijadwalkan akan diganti telah dijadwalkan ulang untuk diganti selama periode baru yang dijelaskan dalam notifikasi.<br><br>Untuk informasi tentang tindakan yang dapat Anda ambil, lihat Mengganti simpul. |
| ElastiCache:NodeReplacementScheduled | `ElastiCache:NodeReplacementScheduled : cluster-name` | Sebuah simpul di klaster Anda dijadwalkan akan diganti selama periode yang dijelaskan dalam notifikasi.<br><br>Untuk informasi tentang tindakan yang dapat Anda ambil, lihat Mengganti simpul. |
| ElastiCache:RemoveCacheNodeComplete | `ElastiCache:RemoveCacheNodeComplete : cluster-name` | Sebuah simpul cache telah dihapus dari klaster cache. |
| ElastiCache:ReplicationGroupScalingComplete | `ElastiCache:ReplicationGroupScalingComplete : cluster-name` | Operasi peningkatan skala pada grup replikasi berhasil diselesaikan. |
| ElastiCache:ReplicationGroupScalingFailed | `"Failed applying modification to cache node type to %s."` | Operasi peningkatan skala pada grup replikasi gagal. |
| ElastiCache:ServiceUpdateAvailableForNode | `"Service update is available for cache node %s."` | Pembaruan mandiri tersedia untuk simpul. |

| Nama Peristiwa | Pesan | Deskripsi |
| --- | --- | --- |
| ElastiCache:SnapshotComplete (hanya Redis) | `ElastiCache:SnapshotComplete : cluster-name` | Sebuah snapshot cache telah berhasil diselesaikan. |
| ElastiCache:SnapshotFailed (hanya Redis) | `SnapshotFailed : cluster-name` | Sebuah snapshot cache telah gagal. Lihat peristiwa klaster cache untuk penyebab yang lebih terperinci.<br><br>Jika Anda menjelaskan snapshot, lihat `DescribeSnapshots` , statusnya adalah `failed`. |

Topik terkait

- Melihat peristiwa ElastiCache

# Pencatatan log panggilan API Amazon ElastiCache dengan AWS CloudTrail

Amazon ElastiCache terintegrasi dengan AWS CloudTrail, sebuah layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, atau layanan AWS di Amazon ElastiCache. CloudTrail menangkap semua panggilan API untuk Amazon ElastiCache sebagai peristiwa, termasuk panggilan dari konsol ElastiCache dan dari panggilan kode ke operasi API Amazon ElastiCache. Jika membuat jejak, Anda dapat mengaktifkan pengiriman peristiwa CloudTrail berkelanjutan ke bucket Amazon S3, termasuk peristiwa untuk Amazon ElastiCache. Jika Anda tidak mengonfigurasi jejak, Anda masih dapat melihat peristiwa terbaru dalam konsol CloudTrail di Riwayat peristiwa. Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat untuk Amazon ElastiCache, alamat IP asal permintaan, siapa yang membuat permintaan, waktu pembuatan permintaan, dan detail tambahan.

Untuk mempelajari selengkapnya tentang CloudTrail, lihat Panduan Pengguna AWS CloudTrail.

## Informasi Amazon ElastiCache di CloudTrail

CloudTrail diaktifkan pada akun AWS saat Anda membuat akun tersebut. Saat terjadi aktivitas di Amazon ElastiCache, aktivitas tersebut akan dicatat dalam peristiwa CloudTrail bersama peristiwa layanan AWS lainnya di Riwayat peristiwa. Anda dapat melihat, mencari, dan mengunduh peristiwa terbaru di akun AWS. Untuk informasi selengkapnya, lihat Melihat Peristiwa dengan Riwayat Peristiwa CloudTrail.

Untuk rekaman yang sedang berlangsung suatu peristiwa di akun AWS Anda, termasuk peristiwa untuk Amazon ElastiCache, buatlah jejak. Jejak memungkinkan CloudTrail mengirimkan file log ke bucket Amazon S3. Secara default, ketika Anda membuat jejak di konsol tersebut, jejak diterapkan ke semua Wilayah . Jejak mencatat log peristiwa dari semua Wilayah di partisi AWS dan mengirimkan file log ke bucket Amazon S3 yang Anda tentukan. Selain itu, Anda dapat mengonfigurasi layanan AWS lainnya untuk menganalisis lebih lanjut dan bertindak berdasarkan data peristiwa yang dikumpulkan di log CloudTrail. Untuk informasi selengkapnya, lihat berikut ini:

- Gambaran Umum untuk Membuat Jejak
- Layanan dan Integrasi yang Didukung CloudTrail
- Mengonfigurasi Notifikasi Amazon SNS untuk CloudTrail
- Menerima File Log CloudTrail dari Banyak Wilayah dan Menerima File Log CloudTrail dari Banyak Akun

Semua tindakan Amazon ElastiCache dicatat oleh CloudTrail dan didokumentasikan dalam Referensi API ElastiCache. Misalnya, panggilan ke tindakan `CreateCacheCluster`, `DescribeCacheCluster`, dan `ModifyCacheCluster` menghasilkan entri di file log CloudTrail.

Setiap entri peristiwa atau log berisi informasi tentang siapa yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan berikut ini:

- Apakah permintaan tersebut dibuat dengan kredensial root atau pengguna IAM.
- Apakah permintaan tersebut dibuat dengan kredensial keamanan sementara untuk satu peran atau pengguna gabungan.
- Apakah permintaan tersebut dibuat oleh layanan AWS lain.

Untuk informasi selengkapnya, silakan lihat CloudTrail userIdentity Element.

## Memahami entri file log Amazon ElastiCache

Jejak adalah konfigurasi yang memungkinkan pengiriman peristiwa sebagai file log ke bucket Amazon S3 yang telah Anda tentukan. File log CloudTrail berisi satu atau beberapa entri log. Peristiwa merepresentasikan satu permintaan dari sumber apa pun dan menyertakan informasi tentang tindakan yang diminta, tanggal dan waktu tindakan, parameter permintaan, dan sebagainya. File log CloudTrail bukan merupakan jejak tumpukan panggilan API publik yang berurutan, sehingga file tersebut tidak muncul dalam urutan tertentu.

Contoh berikut menunjukkan entri log CloudTrail yang menunjukkan tindakan `CreateCacheCluster`.

```
{
    "eventVersion":"1.01",
    "userIdentity":{
        "type":"IAMUser",
        "principalId":"EXAMPLEEXAMPLEEXAMPLE",
        "arn":"arn:aws:iam::123456789012:user/elasticache-allow",
        "accountId":"123456789012",
        "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
        "userName":"elasticache-allow"
    },
    "eventTime":"2014-12-01T22:00:35Z",
    "eventSource":"elasticache.amazonaws.com",
    "eventName":"CreateCacheCluster",
    "awsRegion":"us-west-2",
```

```
    "sourceIPAddress":"192.0.2.01",
    "userAgent":"AWS CLI/ElastiCache 1.10 API 2014-12-01",
    "requestParameters":{
        "numCacheNodes":2,
        "cacheClusterId":"test-memcached",
        "engine":"memcached",
        "aZMode":"cross-az",
        "cacheNodeType":"cache.m1.small",

    },
    "responseElements":{
        "engine":"memcached",
        "clientDownloadLandingPage":"https://console.aws.amazon.com/elasticache/
home#client-download:",
        "cacheParameterGroup":{
            "cacheParameterGroupName":"default.memcached1.4",
            "cacheNodeIdsToReboot":{
            },
            "parameterApplyStatus":"in-sync"
        },
        "preferredAvailabilityZone":"Multiple",
        "numCacheNodes":2,
        "cacheNodeType":"cache.m1.small",

        "cacheClusterStatus":"creating",
        "autoMinorVersionUpgrade":true,
        "preferredMaintenanceWindow":"thu:05:00-thu:06:00",
        "cacheClusterId":"test-memcached",
        "engineVersion":"1.4.14",
        "cacheSecurityGroups":[
            {
                "status":"active",
                "cacheSecurityGroupName":"default"
            }
        ],
        "pendingModifiedValues":{
        }
    },
    "requestID":"104f30b3-3548-11e4-b7b8-6d79ffe84edd",
    "eventID":"92762127-7a68-42ce-8787-927d2174cde1"
}
```

Contoh berikut menunjukkan entri log CloudTrail yang menunjukkan tindakan `DescribeCacheCluster`. Perhatikan bahwa untuk semua panggilan Describe Amazon ElastiCache (`Describe*`), bagian `ResponseElements` dihapus dan muncul sebagai `null`.

```
{
    "eventVersion":"1.01",
    "userIdentity":{
        "type":"IAMUser",
        "principalId":"EXAMPLEEXAMPLEEXAMPLE",
        "arn":"arn:aws:iam::123456789012:user/elasticache-allow",
        "accountId":"123456789012",
        "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
        "userName":"elasticache-allow"
    },
    "eventTime":"2014-12-01T22:01:00Z",
    "eventSource":"elasticache.amazonaws.com",
    "eventName":"DescribeCacheClusters",
    "awsRegion":"us-west-2",
    "sourceIPAddress":"192.0.2.01",
    "userAgent":"AWS CLI/ElastiCache 1.10 API 2014-12-01",
    "requestParameters":{
        "showCacheNodeInfo":false,
        "maxRecords":100
    },
    "responseElements":null,
    "requestID":"1f0b5031-3548-11e4-9376-c1d979ba565a",
    "eventID":"a58572a8-e81b-4100-8e00-1797ed19d172"
}
```

Contoh berikut menunjukkan entri log CloudTrail yang mencatat tindakan `ModifyCacheCluster`.

```
{
    "eventVersion":"1.01",
    "userIdentity":{
        "type":"IAMUser",
        "principalId":"EXAMPLEEXAMPLEEXAMPLE",
        "arn":"arn:aws:iam::123456789012:user/elasticache-allow",
        "accountId":"123456789012",
        "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
        "userName":"elasticache-allow"
    },
    "eventTime":"2014-12-01T22:32:21Z",
    "eventSource":"elasticache.amazonaws.com",
```

```
    "eventName":"ModifyCacheCluster",
    "awsRegion":"us-west-2",
    "sourceIPAddress":"192.0.2.01",
    "userAgent":"AWS CLI/ElastiCache 1.10 API 2014-12-01",
    "requestParameters":{
        "applyImmediately":true,
        "numCacheNodes":3,
        "cacheClusterId":"test-memcached"
    },
    "responseElements":{
        "engine":"memcached",
        "clientDownloadLandingPage":"https://console.aws.amazon.com/elasticache/
home#client-download:",
        "cacheParameterGroup":{
            "cacheParameterGroupName":"default.memcached1.4",
            "cacheNodeIdsToReboot":{
            },
            "parameterApplyStatus":"in-sync"
        },
        "cacheClusterCreateTime":"Dec 1, 2014 10:16:06 PM",
        "preferredAvailabilityZone":"Multiple",
        "numCacheNodes":2,
        "cacheNodeType":"cache.m1.small",
        "cacheClusterStatus":"modifying",
        "autoMinorVersionUpgrade":true,
        "preferredMaintenanceWindow":"thu:05:00-thu:06:00",
        "cacheClusterId":"test-memcached",
        "engineVersion":"1.4.14",
        "cacheSecurityGroups":[
            {
                "status":"active",
                "cacheSecurityGroupName":"default"
            }
        ],
        "configurationEndpoint":{
            "address":"test-memcached.example.cfg.use1prod.cache.amazonaws.com",
            "port":11211
        },
        "pendingModifiedValues":{
            "numCacheNodes":3
        }
    },
    "requestID":"807f4bc3-354c-11e4-9376-c1d979ba565a",
    "eventID":"e9163565-376f-4223-96e9-9f50528da645"
```

```
}
```

# Kuota untuk ElastiCache

Akun AWS Anda memiliki kuota default, yang sebelumnya disebut sebagai batas, untuk setiap layanan AWS. Kecuali dinyatakan sebaliknya, setiap kuota unik untuk suatu Wilayah. Anda dapat meminta penambahan untuk beberapa kuota, sementara kuota lainnya tidak dapat ditambah.

Untuk melihat kuota ElastiCache, buka Konsol Kuota Layanan. Di panel navigasi, pilih Layanan AWS dan pilih ElastiCache.

Untuk meminta penambahan kuota, lihat Meminta Penambahan Kuota dalam Panduan Pengguna Kuota Layanan. Jika kuota belum tersedia dalam Kuota Layanan, gunakan formulir penambahan batas.

Akun AWS Anda memiliki kuota berikut yang terkait dengan ElastiCache.

| Sumber daya | Default |
|---|---|
| Cache nirserver per wilayah | 40 |
| Snapshot nirserver per hari per cache | 24 |
| Simpul per Wilayah | 300* |
| Simpul per klaster per jenis instans (Mode klaster Redis diaktifkan) | 90 |
| Simpul per serpihan (mode klaster Redis dinonaktifkan) | 6 |
| Grup parameter per Wilayah | 300* |
| Grup keamanan per Wilayah | 50 |
| Grup subnet per Wilayah | 300* |
| Subnet per grup subnet | 20* |
| Pengguna per grup pengguna | =100. |
| Jumlah maksimum pengguna | 1000* |

| Sumber daya | Default |
| --- | --- |
| Jumlah maksimum grup pengguna | =100. |

# Referensi

Topik pada bagian ini mencakup bekerja dengan API Amazon ElastiCache dan bagian ElastiCache dari AWS CLI. Juga disertakan pada bagian ini adalah pesan kesalahan dan notifikasi layanan yang umum.

- Menggunakan API ElastiCache
- Referensi API ElastiCache
- Bagian ElastiCache dari Referensi AWS CLI
- Pesan kesalahan Amazon ElastiCache
- Notifikasi

# Menggunakan API ElastiCache

Bagian ini menyediakan deskripsi berorientasi tugas tentang cara menggunakan dan menerapkan operasi ElastiCache. Untuk deskripsi lengkap tentang operasi ini, lihat Referensi API Amazon ElastiCache

Topik

- Menggunakan API kueri
- Pustaka yang tersedia
- Memecahkan masalah aplikasi

## Menggunakan API kueri

### Parameter kueri

Permintaan berbasis Kueri HTTP adalah permintaan HTTP yang menggunakan verb GET atau POST dan parameter Kueri bernama `Action`.

Setiap permintaan Kueri harus menyertakan beberapa parameter umum untuk menangani autentikasi dan pemilihan tindakan.

Beberapa operasi mengambil daftar parameter. Daftar ini ditentukan menggunakan notasi `param.`*n*. Nilai *n* adalah integer yang dimulai dari 1.

## Autentikasi permintaan Kueri

Anda hanya dapat mengirim permintaan Kueri melalui HTTPS, dan Anda harus menyertakan tanda tangan di setiap permintaan Kueri. Bagian ini menjelaskan cara membuat tanda tangan. Metode yang dijelaskan dalam prosedur berikut ini dikenal sebagai tanda tangan versi 4.

Berikut ini adalah langkah dasar yang digunakan untuk memberikan autentikasi kepada AWS. Ini mengasumsikan Anda telah terdaftar dengan AWS dan memiliki ID Kunci Aksed dan Kunci Akses Rahasia.

Proses autentikasi Query

1. Pengirim membangun permintaan ke AWS.
2. Pengirim menghitung tanda tangan permintaan, Hashing Berkunci untuk Kode Autentikasi Pesan Berbasis Hash (HMAC) dengan fungsi hash SHA-1, seperti yang didefinisikan dalam bagian berikutnya dari topik ini.
3. Pengirim permintaan mengirimkan data permintaan, tanda tangan, dan ID Kunci Akses (pengidentifikasi kunci dari Kunci Akses Rahasia yang digunakan) ke AWS.
4. AWS menggunakan ID Kunci Akses untuk mencari Kunci Akses Rahasia.
5. AWS menghasilkan tanda tangan dari data permintaan dan Kunci Akses Rahasia menggunakan algoritma yang sama dengan yang digunakan untuk menghitung tanda tangan pada permintaan.
6. Jika tanda tangan cocok, maka permintaan tersebut dianggap autentik. Jika perbandingan gagal, permintaan dibatalkan, dan AWS akan memberikan respons kesalahan.

> **Note**
>
> Jika permintaan berisi parameter `Timestamp`, tanda tangan yang dihitung untuk permintaan akan kedaluwarsa dalam 15 menit setelah nilai yang tertera.
> Jika permintaan berisi parameter `Expires`, tanda tangan berakhir pada waktu yang ditentukan oleh parameter `Expires`.

Untuk menghitung tanda tangan permintaan

1. Buat string kueri kanonikalisasi yang akan Anda butuhkan nanti dalam prosedur ini:

a. Urutkan komponen string kueri UTF-8 berdasarkan nama parameter dengan pengurutan byte alami. Parameter dapat berasal dari GET URI atau dari isi POST (jika Content-Type adalah application/x-www-form-urlencoded).

b. URL mengodekan nama parameter dan nilainya sesuai dengan aturan berikut:

   i. Jangan mengodekan URL karakter tanpa syarat apa pun yang didefinisikan RFC 3986. Karakter tanpa syarat tersebut adalah A-Z, a-z, 0-9, tanda hubung (-), garis bawah (_), titik (.), dan tilde (~).

   ii. Tanda persen mengodekan semua karaktern lain dengan %XY, yang mana X dan Y adalah karakter hex 0-9 dan kapital A-F.

   iii. Tanda persen mengodekan karakter UTF-8 yang diperluas dalam format %XY%ZA....

   iv. Tanda persen mengodekan karakter spasi sebagai %20 (dan bukan + seperti skema pengodean umum).

c. Pisahkan nama parameter yang dikodekan dari nilai yang dikodekan dengan tanda sama dengan (=) (karakter ASCII 61), meskipun jika nilai parameter itu kosong.

d. Pisahkan pasangan nama-nilai dengan tanda ampersan (&) (kode ASCII 38).

2. Buat string untuk menandatangani sesuai dengan pseudo-tata bahasa berikut (tanda “\n” berarti baris baru ASCII).

```
StringToSign = HTTPVerb + "\n" +
ValueOfHostHeaderInLowercase + "\n" +
HTTPRequestURI + "\n" +
CanonicalizedQueryString <from the preceding step>
```

   Komponen HTTPRequestURI adalah komponen jalur absolut HTTP dari URI hingga, tapi tidak termasuk, string kueri. Jika HTTPRequestURI kosong, gunakan garis miring depan ( / ).

3. Hitung HMAC yang sesuai RFC 2104 dengan string yang baru saja Anda buat, Kunci Akses Rahasia Anda sebagai kunci, dan SHA256 atau SHA1 sebagai algoritma hash.

   Untuk informasi lebih lanjut, lihat https://www.ietf.org/rfc/rfc2104.txt.

4. Konversikan nilai yang dihasilkan ke base64.

5. Sertakan nilai sebagai nilai dari parameter `Signature` dalam permintaan.

Misalnya, berikut ini adalah contoh permintaan (jeda baris ditambahkan agar jelas).

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeCacheClusters
    &CacheClusterIdentifier=myCacheCluster
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-12-01
```

Untuk kueri string sebelumnya, Anda akan menghitung tanda tangan HMAC atas string berikut.

```
GET\n
    elasticache.amazonaws.com\n
    Action=DescribeCacheClusters
    &CacheClusterIdentifier=myCacheCluster
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-12-01
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE%2F20140523%2Fus-west-2%2Felasticache
%2Faws4_request
    &X-Amz-Date=20141201T223649Z
    &X-Amz-SignedHeaders=content-type%3Bhost%3Buser-agent%3Bx-amz-content-sha256%3Bx-
amz-date
        content-type:
        host:elasticache.us-west-2.amazonaws.com
        user-agent:CacheServicesAPICommand_Client
    x-amz-content-sha256:
    x-amz-date:
```

Hasilnya adalah permintaan yang ditandatangani berikut.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeCacheClusters
    &CacheClusterIdentifier=myCacheCluster
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-12-01
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20141201/us-west-2/elasticache/aws4_request
    &X-Amz-Date=20141201T223649Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=2877960fced9040b41b4feaca835fd5cfeb9264f768e6a0236c9143f915ffa56
```

Untuk informasi rinci tentang proses penandatanganan dan penghitungan tanda tangan permintaan, lihat topik Proses Penandatanganan Tanda Tangan Versi 4 dan subtopiknya.

## Pustaka yang tersedia

AWS menyediakan kit pengembangan perangkat lunak (SDK) untuk developer perangkat lunak yang lebih suka membangun aplikasi menggunakan API bahasa tertentu bukannya API Kueri. SDK ini menyediakan fungsi dasar (tidak disertakan dalam API), seperti autentikasi permintaan, percobaan ulang permintaan, dan penanganan kesalahan sehingga lebih mudah untuk memulai. SDK dan sumber daya tambahan tersedia dalam bahasa pemrograman berikut:

- Java
- Windows dan .NET
- PHP
- Python
- Ruby

Untuk informasi tentang bahasa lain, lihat Contoh Kode & Pustaka.

## Memecahkan masalah aplikasi

ElastiCache memberikan penjelasan tentang kesalahan spesifik dan deskriptif untuk membantu Anda memecahkan masalah saat berinteraksi dengan API ElastiCache.

### Mengambil kesalahan

Biasanya, Anda ingin aplikasi Anda memeriksa apakah permintaan menimbulkan kesalahan sebelum Anda menghabiskan waktu untuk memproses hasil. Cara termudah untuk mengetahui jika terjadi kesalahan adalah dengan mencari simpul `Error` di dalam respons dari API ElastiCache.

Sintaks XPath menyediakan cara sederhana untuk mencari keberadaan simpul `Error`, serta cara mudah untuk mengambil kode dan pesan kesalahan. Cuplikan kode berikut menggunakan modul Perl dan XML::XPath untuk menentukan jika kesalahan terjadi selama permintaan. Jika terjadi kesalahan, kode akan mencetak pesan dan kode kesalahan pertama dalam tanggapannya.

```
use XML::XPath;
```

```
my $xp = XML::XPath->new(xml =>$response);
if ( $xp->find("//Error") )
{print "There was an error processing your request:\n", " Error code: ",
$xp->findvalue("//Error[1]/Code"), "\n", " ",
$xp->findvalue("//Error[1]/Message"), "\n\n"; }
```

### Tip pemecahan masalah

Dianjurkan untuk melakukan proses berikut untuk mendiagnosis dan menyelesaikan masalah dengan API ElastiCache.

- Pastikan bahwa ElastiCache berjalan dengan benar.

  Untuk melakukannya, cukup buka jendela browser dan kirimkan permintaan kueri ke layanan ElastiCache (seperti https://elasticache.amazonaws.com). MissingAuthenticationTokenException atau Kesalahan Server Internal 500 mengonfirmasi bahwa layanan tersedia dan menanggapi permintaan.

- Periksa struktur permintaan Anda.

  Setiap operasi ElastiCache memiliki halaman referensi di Referensi API ElastiCache. Periksa ulang bahwa Anda menggunakan parameter dengan benar. Untuk mengetahui kemungkinan kesalahan, lihat contoh permintaan atau skenario pengguna untuk melihat apakah contoh tersebut melakukan operasi serupa.

- Periksa forum.

  ElastiCache memiliki forum diskusi tempat Anda dapat mencari solusi untuk masalah yang dialami orang lain selama ini. Untuk melihat forum, kunjungi

  https://forums.aws.amazon.com/ .

## Menyiapkan antarmuka baris perintah ElastiCache

Bagian ini menjelaskan prasyarat untuk menjalankan alat baris perintah, tempat mendapatkannya, cara mengatur lingkungannya, dan contoh umum penggunaan alat.

Ikuti petunjuk dalam topik ini hanya jika Anda akan menggunakan AWS CLI untuk ElastiCache.

> **Important**
>
> Antarmuka Baris Perintah (CLI) Amazon ElastiCache tidak support perbaikan ElastiCache setelah API versi 2014-09-30. Untuk menggunakan fungsionalitas ElastiCache yang lebih baru dari baris perintah, gunakan Antarmuka Baris Perintah AWS.

Topik



# Prasyarat

Dokumen ini mengasumsikan bahwa Anda dapat bekerja di lingkungan Linux/UNIX atau Windows. Alat baris perintah Amazon ElastiCache juga bekerja pada Mac OS X, yang merupakan lingkungan berbasis UNIX; namun, tidak ada petunjuk Mac OS X tertentu yang disertakan dalam panduan ini.

Sebagai konvensi, semua teks baris perintah diawali dengan prompt baris perintah **`PROMPT>`** generik. Prompt baris perintah yang sebenarnya pada mesin Anda mungkin berbeda. Kami juga menggunakan **`$`** untuk menunjukkan perintah khusus Linux/UNIX dan **`C:\>`** untuk perintah khusus Windows. Contoh output yang dihasilkan dari perintah ditampilkan segera tanpa prefiks apapun.

## Lingkungan runtime Java

Alat baris perintah yang digunakan dalam panduan ini memerlukan Java versi 5 atau yang lebih baru. Dapat menggunakan instalasi JRE atau JDK. Untuk melihat dan mengunduh JRE untuk berbagai platform, termasuk Linux/UNIX dan Windows, lihat Java SE Downloads.

Mengatur variabel home Java

Alat baris perintah bergantung pada variabel lingkungan (`JAVA_HOME`) untuk menemukan Java Runtime. Variabel lingkungan ini harus ditetapkan ke jalur lengkap dari direktori yang berisi subdirektori bernama `bin` yang berisi `java` executable (di Linux dan UNIX) atau `java.exe` (pada Windows) executable.

Untuk mengatur variabel Java Home

1. Mengatur variabel Java Home.

   - Di Linux dan UNIX, masukkan perintah berikut:

     ```
     $ export JAVA_HOME=<PATH>
     ```

   - Di Windows, masukkan perintah berikut:

     ```
     C:\> set JAVA_HOME=<PATH>
     ```

2. Konfirmasikan pengaturan jalur dengan menjalankan **$JAVA_HOME/bin/java -version** dan memeriksa outputnya.

   - Di Linux/UNIX, Anda akan melihat output yang mirip dengan berikut ini:

     ```
     $ $JAVA_HOME/bin/java -version
     java version "1.6.0_23"
     Java(TM) SE Runtime Environment (build 1.6.0_23-b05)
     Java HotSpot(TM) Client VM (build 19.0-b09, mixed mode, sharing)
     ```

   - Di Windows, Anda akan melihat output yang mirip dengan berikut ini:

     ```
     C:\> %JAVA_HOME%\bin\java -version
     java version "1.6.0_23"
     Java(TM) SE Runtime Environment (build 1.6.0_23-b05)
     Java HotSpot(TM) Client VM (build 19.0-b09, mixed mode, sharing)
     ```

## Mendapatkan alat baris perintah

Alat baris perintah tersedia sebagai file ZIP di situs web Alat Developer ElastiCache. Alat ini ditulis dalam Jawa, dan meliputi skrip shell untuk Windows 2000/XP/Vista/Windows 7, Linux/UNIX, dan Mac OSX. File ZIP bersifat mandiri dan tidak memerlukan penginstalan; cukup unduh file zip dan lakukan unzip ke direktori di mesin lokal Anda.

# Menyiapkan alat

Alat baris perintah bergantung pada variabel lingkungan (AWS_ELASTICACHE_HOME) untuk menemukan pustaka pendukung. Anda perlu mengatur variabel lingkungan ini sebelum dapat menggunakan alat. Tetapkan ke jalur direktori tempat Anda melakukan unzip dengan alat baris perintah. Direktori ini bernama ElastiCachecli-A.B.nnn (A, B dan n adalah nomor versi/rilis), dan berisi subdirektori bernama bin dan lib.

Untuk menetapkan variabel lingkungan AWS_ELASTICACHE_HOME

- Buka jendela baris perintah dan masukkan salah satu perintah berikut untuk menetapkan variabel lingkungan AWS_ElastiCache_HOME.
  - Di Linux dan UNIX, masukkan perintah berikut:

```
$ export &AWS;_ELASTICACHE_HOME=<path-to-tools>
```

  - Di Windows, masukkan perintah berikut:

```
C:\> set &AWS;_ELASTICACHE_HOME=<path-to-tools>
```

Agar alat lebih mudah digunakan, sebaiknya tambahkan direktori BIN dari alat di PATH sistem Anda. Sisa dari panduan ini mengasumsikan bahwa direktori BIN sudah ada di jalur sistem Anda.

Untuk menambahkan direktori BIN dari alat ke jalur sistem Anda

- Masukkan perintah berikut untuk menambahkan direktori BIN dari alat ke PATH sistem Anda.
  - Di Linux dan UNIX, masukkan perintah berikut:

```
$ export PATH=$PATH:$&AWS;_ELASTICACHE_HOME/bin
```

  - Di Windows, masukkan perintah berikut:

```
C:\> set PATH=%PATH%;%&AWS;_ELASTICACHE_HOME%\bin
```

> **Note**
>
> Variabel lingkungan Windows di-reset ketika Anda menutup jendela perintah. Anda mungkin ingin menetapkannya secara permanen. Lihat dokumentasi untuk versi Windows Anda untuk informasi selengkapnya.

> **Note**
>
> Jalur yang berisi spasi harus dibungkus dengan tanda kutip ganda, misalnya: "C:\Program Files\Java"

## Memberikan kredensial untuk alat

Alat baris perintah membutuhkan Kunci Akses dan Kunchi Akses Rahasi AWS yang disediakan dengan akun AWS Anda. Anda bisa mendapatkannya menggunakan baris perintah atau dari file kredensial yang terletak di sistem lokal Anda.

Deployment termasuk file templat ${AWS_ELASTICACHE_HOME}/credential-file-path.template yang Anda butuhkan untuk mengedit dengan informasi Anda. Berikut adalah isi dari file templat:

```
AWSAccessKeyId=<Write your AWS access ID>
AWSSecretKey=<Write your AWS secret key>
```

> **Important**
>
> Di UNIX, batasi izin untuk pemilik file kredensial:
>
> ```
> $ chmod 600 <the file created above>
> ```

Dengan pengaturan file kredensial, Anda harus menetapkan variabel lingkungan _CREDENTIAL_FILE AWS sehingga alat ElastiCache dapat menemukan informasi Anda.

Untuk menetapkan variabel lingjungan AWS_CREDENTIAL_FILE

1. Atur variabel lingkungan :

   - Di Linux dan UNIX, perbarui variabel menggunakan perintah berikut:

     ```
     $ export &AWS;_CREDENTIAL_FILE=<the file created above>
     ```

   - Di Windows, tetapkan variabel menggunakan perintah berikut:

     ```
     C:\> set &AWS;_CREDENTIAL_FILE=<the file created above>
     ```

2. Periksa bahwa pengaturan Anda bekerja dengan baik, jalankan perintah berikut:

   ```
   elasticache --help
   ```

   Anda harus melihat halaman penggunaan untuk semua perintah ElastiCache.

## Variabel lingkungan

Variabel lingkungan dapat berguna untuk membuat skrip, membuat konfigurasi default atau menimpa variabel untuk sementara.

Selain variabel lingkungan AWS_CREDENTIAL_FILE, sebagian besar alat API disertakan dengan Antarmuka Baris Perintah ElastiCache yang mendukung variabel berikut:

- EC2_REGION — Wilayah AWS untuk digunakan.
- AWS_ELASTICACHE_URL — URL yang digunakan untuk panggilan layanan. Tidak perlu menentukan titik akhir wilayah yang berbeda jika EC2_REGION ditentukan atau parameter — region dilewatkan.

Contoh berikut menunjukkan cara menetapkan variabel lingkungan EC2_REGION untuk membuat konfigurasi wilayah yang digunakan oleh alat API:

Linux, OS X, atau Unix

```
$ export EC2_REGION=us-west-1
```

Windows

```
$ set EC2_REGION=us-west-1
```

# Pesan kesalahan Amazon ElastiCache

Pesan kesalahan berikut dikembalikan oleh Amazon ElastiCache. Anda mungkin menerima pesan kesalahan lain yang dikembalikan oleh ElastiCache, layanan AWS lain, atau oleh Redis. Untuk deskripsi pesan kesalahan dari sumber selain ElastiCache, lihat dokumentasi dari sumber yang menghasilkan pesan kesalahan itu.

- Cluster node quota exceeded
- Customer's node quota exceeded
- Manual snapshot quota exceeded
- Insufficient cache cluster capacity

Pesan Kesalahan: Kuota simpul klaster terlampaui. Setiap klaster dapat memiliki paling banyak %n simpul di wilayah ini.

Penyebab: Anda mencoba untuk membuat atau memodifikasi sebuah klaster yang mengakibatkan klaster akan memiliki lebih dari %n simpul.

Solusi: Ubah permintaan Anda sehingga klaster tidak memiliki lebih dari %n simpul. Atau, jika Anda membutuhkan lebih dari %n simpul, buat permintaan Anda menggunakan Formulir permintaan simpul Amazon ElastiCache.

Untuk informasi selengkapnya, lihat Batas Amazon ElastiCache di Referensi Umum Amazon Web Services.

Pesan Kesalahan: Kuota simpul pelanggan terlampaui. Anda dapat memiliki maksimal %n simpul di wilayah ini Atau,Anda telah mencapai kuota Anda %s simpul di wilayah ini.

Penyebab: Anda mencoba untuk membuat atau mengubah sebuah klaster yang mengakibatkan akun Anda memiliki lebih dari %n simpul di seluruh klaster di wilayah ini.

Solusi: Ubah permintaan Anda sehingga jumlah simpul di wilayah di semua klaster untuk akun ini tidak melebihi %n. Atau, jika Anda membutuhkan lebih dari %n simpul, buat permintaan Anda menggunakan Formulir permintaan simpul Amazon ElastiCache.

Untuk informasi selengkapnya, lihat Batas Amazon ElastiCache di Referensi Umum Amazon Web Services.

Pesan Kesalahan: Jumlah maksimum snapshot manual untuk klaster ini yang diambil dalam waktu 24 jam telah tercapai atau Jumlah maksimum snapshot manual untuk simpul ini yang diambil dalam waktu 24 jam telah mencapai kuota %n

Penyebab: Anda mencoba untuk mengambil snapshot manual dari sebuah klaster ketika Anda telah mengambil jumlah maksimum snapshot manual diperbolehkan dalam periode 24 jam.

Solusi: Tunggu 24 jam untuk mencoba kembali snapshot manual dari klaster tersebut. Atau, jika Anda perlu untuk mengambil snapshot manual sekarang, ambil snapshot dari simpul lain yang memiliki data yang sama, seperti simpul yang berbeda dalam sebuah klaster.

Pesan Kesalahan:InsufficientCacheClusterCapacity

Penyebab: AWS saat ini tidak memiliki kapasitas item Sesuai Permintaan yang tersedia untuk melayani permintaan Anda.

Solusi:

- Tunggu beberapa menit, lalu kirim permintaan Anda lagi; kapasitas sering kali dapat berubah.
- Kirim permintaan baru dengan pengurangan jumlah simpul atau serpihan (grup simpul). Misalnya, jika Anda membuat satu permintaan untuk meluncurkan 15 instans, cobalah membuat 3 permintaan untuk 5 instans, alih-alih 15 permintaan untuk 1 simpul.
- Jika Anda meluncurkan klaster, kirimkan permintaan baru tanpa menentukan Zona Ketersediaan.
- Jika Anda meluncurkan sebuah klaster, kirimkan permintaan baru menggunakan jenis simpul yang berbeda (skalanya dapat dinaikkan di tahap berikutnya). Untuk informasi selengkapnya, lihat Penskalaan ElastiCache untuk Redis .

# Notifikasi

Topik berikut mencakup notifikasi ElastiCache yang mungkin Anda minati. Notifikasi adalah situasi atau peristiwa yang, dalam kebanyakan kasus, bersifat sementara, hanya berlangsung sampai solusi

ditemukan dan diimplementasikan. Notifikasi umumnya memiliki tanggal mulai dan tanggal resolusi, setelah itu notifikasi menjadi tidak relevan lagi. Setiap notifikasi mungkin atau mungkin tidak relevan bagi Anda. Kami menganjurkan pedoman implementasi yang, jika diikuti, akan meningkatkan kinerja klaster Anda.

Notifikasi tidak mengumumkan fitur atau fungsionalitas ElastiCache yang baru atau diperbaiki.

## Notifikasi ElastiCache Umum

Saat ini tidak ada notifikasi ElastiCache yang menggantung yang tidak spesifik pada mesin.

## Notifikasi khusus ElastiCache for Redis

Saat ini tidak ada notifikasi ElastiCache for Redis yang belum ditangani.

# ElastiCache untuk sejarah Dokumentasi Redis

- Versi API: 2015-02-02
- Pembaruan dokumentasi terbaru: 27 November 2023

Tabel berikut menjelaskan perubahan penting dalam setiap rilis Panduan Pengguna Redis setelah Maret 2018. Untuk notifikasi tentang pembaruan-pembaruan dokumentasi ini, Anda dapat berlangganan ke sebuah umpan RSS.

Terbaru ElastiCache untuk Redis Updates

| Perubahan | Deskripsi | Tanggal |
| --- | --- | --- |
| ElastiCache untuk Redis menambahkan dukungan untuk ukuran node C7gn tambahan | ElastiCache untuk Redis menambahkan dukungan untuk ukuran node C7gn tambahan. | 10 Januari 2024 |
| ElastiCache untuk Redis sekarang mendukung pembuatan cache tanpa server | Anda sekarang dapat membuat cache nirserver , yang menyederhanakan manajemen cache dan langsung melakukan penskalaan untuk mendukung aplikasi dengan tuntutan tinggi. Untuk informasi selengkapnya, lihat Memilih di antara opsi deploymen t. Sebagai bagian dari fitur ini, izin baru ditambahkan ke `ElastiCacheService RolePolicy` dan `AmazonElastiCacheF ullAccess` untuk memungkinkan asosiasi cache nirserver dengan titik | 27 November 2023 |

| | | |
|---|---|---|
| | akhir VPC terkelola. Selain itu, izin ditambahkan untuk mendukung pengalaman konsol yang direvisi menggunakan kebijakan `AmazonElastiCacheFullAccess` . | |
| ElastiCache untuk Redis sekarang mendukung memodifikasi mode cluster | Anda sekarang dapat memigrasikan klaster dari Mode Klaster Dinonaktifkan (CMD) ke Mode Klaster Diaktifkan (CME). Untuk informasi selengkapnya, lihat Mengubah mode klaster. | 11 Mei 2023 |
| ElastiCache untuk Redis sekarang mendukung modifikasi pengaturan enkripsi dalam transit | Anda sekarang dapat mengubah konfigurasi TLS klaster Redis Anda tanpa perlu membangun kembali atau menyediakan kembali klaster atau memengaruhi ketersediaan aplikasi. Untuk informasi selengkapnya, lihat Mengaktifkan enkripsi bergerak pada klaster yang telah ada. | 28 Desember 2022 |

| | | |
|---|---|---|
| ElastiCache untuk Redis sekarang mendukung otentikasi pengguna menggunakan IAM | Otentikasi IAM memungkinkan Anda untuk mengautentikasi koneksi ke Redis menggunakan ElastiCache identitas IAM. AWS Hal ini memungkinkan Anda memperkuat model keamanan Anda dan menyederhanakan banyak tugas keamanan administratif. Untuk informasi selengkapnya, silakan lihat Autentikasi dengan IAM. | 16 November 2022 |
| ElastiCache untuk Redis sekarang mendukung Redis 7 | Rilis ini membawa beberapa fitur baru ke Amazon ElastiCache untuk Redis: fungsi Redis, peningkatan ACL dan Sharded Pub/Sub. Untuk informasi selengkapnya, lihat ElastiCache untuk Redis versi 7.0. | 8 November 2022 |

| | | |
|---|---|---|
| ElastiCache untuk Redis sekarang mendukung IPV6 | ElastiCache mendukung Internet Protocol versi 4 dan 6 (IPv4 dan IPv6), memungkinkan Anda untuk mengkonfigurasi cluster Anda untuk hanya menerima koneksi IPv4, hanya koneksi IPv6 atau koneksi IPv4 dan IPv6 (dual-stack). IPv6 didukung untuk beban kerja yang menggunakan mesin Redis versi 6.2 dan seterusnya pada semua instans yang dibangun di sistem Nitro. Tidak ada biaya tambahan untuk mengakses ElastiCache lebih dari IPv6. Untuk informasi selengkapnya, lihat Memilih jenis jaringan. | 7 November 2022 |

| | | |
|---|---|---|
| ElastiCache untuk Redis sekarang mendukung format Notasi JavaScript Objek (JSON) asli | Format asli JavaScript Object Notation (JSON) adalah cara sederhana dan tanpa skema untuk menyandikan kumpulan data kompleks di dalam kluster Redis. Anda dapat menyimpan dan mengakses data secara native menggunakan format JavaScript Object Notation (JSON) di dalam kluster Redis dan memperbarui data JSON yang disimpan dalam cluster tersebut, tanpa perlu mengelola kode khusus untuk membuat serial dan deseriali sasinya. Untuk informasi selengkapnya, lihat Memulai dengan JSON. | 25 Mei 2022 |
| ElastiCache sekarang mendukung PrivateLink | AWS PrivateLink memungkin kan Anda mengakses operasi ElastiCache API secara pribadi tanpa gateway internet, perangkat NAT, koneksi VPN, atau koneksi Direct AWS Connect. Untuk informasi selengkapnya, lihat Amazon ElastiCache API dan antarmuka VPC endpoint (AWS PrivateLink) untuk Redis atau ElastiCache Amazon API dan antarmuka VPC endpoint () untuk Memcached. AWS PrivateLink | 24 Januari 2022 |

| | | |
|---|---|---|
| ElastiCache untuk Redis sekarang mendukung Redis 6.2 dan Data Tiering | Amazon ElastiCache for Redis memperkenalkan versi berikutnya dari mesin Redis yang didukung oleh Amazon. ElastiCache ElastiCache untuk Redis 6.2 mencakup peningkatan kinerja untuk cluster berkemampuan TLS menggunakan tipe node x86 dengan 8 vCPU atau lebih atau tipe node Graviton2 dengan 4 vCPU atau lebih. ElastiCache untuk Redis juga memperkenalkan data tiering. Anda dapat menggunak an tingkatan data sebagai cara berbiaya lebih rendah untuk menskalakan klaster Anda hingga ratusan terabyte kapasitas. Untuk informasi selengkapnya, lihat ElastiCac he untuk Redis versi 6.2 (ditingkatkan) dan Tingkat data. | 23 November 2021 |

| | | |
|---|---|---|
| Dukungan untuk Penskalaan Otomatis | ElastiCache untuk Redis sekarang mendukung Auto Scaling. ElastiCache untuk Redis auto scaling adalah kemampuan untuk menambah atau mengurangi pecahan atau replika yang diinginkan dalam layanan ElastiCache untuk Redis Anda secara otomatis. ElastiCache memanfaatkan layanan Application Auto Scaling untuk menyediakan fungsionalitas ini. Untuk informasi selengkapnya, lihat Auto Scaling ElastiCache untuk kluster Redis. | 19 Agustus 2021 |
| Dukungan untuk pengiriman Log Lambat Redis | ElastiCache sekarang memungkinkan Anda melakukan streaming Redis SLOWLOG ke salah satu dari dua tujuan: Amazon Data Firehose atau Amazon Logs. CloudWatch Untuk informasi selengkapnya, lihat Pengiriman log. | 22 April 2021 |

| | | |
|---|---|---|
| Dukungan untuk penandaan sumber daya dan kunci kondisi | ElastiCache sekarang mendukung penandaan untuk membantu Anda mengelola cluster dan sumber daya lainnya ElastiCache . Untuk informasi selengkapnya, lihat Menandai ElastiCache sumber daya Anda. ElastiCache juga memperkenalkan dukungan untuk kunci kondisi. Anda dapat menentukan kondisi yang menentukan penerapan kebijakan IAM. Untuk informasi selengkapnya, lihat Menggunakan kunci kondisi. | 7 April 2021 |
| ElastiCache sekarang tersedia di AWS Outposts | AWS Outposts membawa AWS layanan asli, infrastruktur, dan model operasi ke hampir semua pusat data, ruang co-lokasi, atau fasilitas lokal. Anda dapat menerapkan ElastiCache di Outposts untuk menyiapkan, mengoperasikan, dan menggunakan cache lokal, seperti yang Anda lakukan di cloud. Untuk informasi selengkapnya, lihat Menggunakan Outposts untuk Redis atau Menggunakan Outposts untuk Memcached. | 8 Oktober 2020 |

| | | |
|---|---|---|
| ElastiCache sekarang mendukung Redis 6 | Amazon ElastiCache for Redis memperkenalkan versi berikutnya dari mesin Redis yang didukung oleh Amazon. ElastiCache Versi ini mencakup autentikasi pengguna dengan kontrol akses berbasis peran, dukungan tanpa versi, caching sisi klien, dan peningkatan operasional yang signifikan. Untuk informasi selengkapnya, lihat ElastiCache untuk Redis Versi 6.0 (Ditingkatkan). | 7 Oktober 2020 |
| ElastiCache sekarang mendukung Local Zones | Zona Lokal adalah perpanjangan dari AWS Wilayah yang secara geografis dekat dengan pengguna Anda. Anda dapat memperluas virtual private cloud (VPC) dari AWS Region induk ke Local Zones dengan membuat subnet baru dan menetapkannya ke Local Zone. Untuk informasi selengkapnya, lihat Menggunakan Zona Lokal. | 25 September 2020 |

| | | |
|---|---|---|
| ElastiCache untuk Redis sekarang mendukung penskalaan lingkungan Redis Cluster Anda hingga 500 node atau 500 pecahan | Mode Klaster Redis membuat konfigurasi yang memungkinkan Anda menggunakannya untuk melakukan partisi data Anda di beberapa serpihan dan menawarkan skalabilitas, kinerja, dan ketersediaan yang lebih baik. Fitur ini tersedia di Amazon ElastiCache untuk Redis versi 5.0.6 dan seterusnya di semua AWS Wilayah dan untuk semua yang ada dan yang baru ElastiCache untuk lingkungan Redis Cluster. Untuk informasi selengkapnya, lihat Simpul dan Serpihan Redis. | 13 Agustus 2020 |
| ElastiCache sekarang mendukung izin tingkat sumber daya | Sekarang Anda dapat membatasi cakupan izin pengguna dengan menentukan ElastiCache sumber daya dalam kebijakan AWS Identity and Access Management (IAM). Untuk informasi selengkapnya, lihat Izin di level sumber daya. | 12 Agustus 2020 |
| ElastiCache untuk Redis menambahkan metrik Amazon CloudWatch tambahan | ElastiCache untuk Redis sekarang mendukung CloudWatch metrik baru, termasuk `PubSubCmds` dan. `HyperLogLogBasedCmds` Untuk daftar yang lengkap, lihat Metrik untuk Redis. | 10 Juni 2020 |

| | | |
|---|---|---|
| ElastiCache sekarang mendukung pembaruan otomatis cluster ElastiCache | Amazon ElastiCache sekarang mendukung pembaruan otomatis ElastiCache cluster setelah “direkomendasikan berlaku berdasarkan tanggal” pembaruan layanan telah berlalu. ElastiCache akan menggunakan jendela pemeliharaan Anda untuk menjadwalkan pembaruan otomatis cluster yang berlaku. Untuk informasi selengkap nya, lihat Pembaruan secara mandiri. | 13 Mei, 2020 |

| | | |
|---|---|---|
| ElastiCache untuk Redis sekarang mendukung Global Datastore untuk Redis | Fitur Global Datastore for Redis menawarkan replikasi yang dikelola sepenuhnya, cepat, andal, dan aman di seluruh Wilayah. AWS Dengan menggunakan fitur ini, Anda dapat membuat klaster replika baca lintas wilayah ElastiCache untuk Redis guna mengaktifkan pembacaan latensi rendah dan pemulihan bencana di seluruh Wilayah. AWS Anda dapat membuat, mengubah, dan menjelaskan penyimpanan data global. Anda juga dapat menambah atau menghapus AWS Regions dari datastore global Anda dan mempromosikan AWS Region sebagai yang utama dalam datastore global. Untuk informasi selengkapnya, lihat Replikasi Lintas AWS Wilayah Menggunakan Datastore Global. | 16 Maret 2020 |
| ElastiCache untuk Redis sekarang mendukung Redis versi 5.0.6 | Untuk informasi selengkapnya, lihat ElastiCache untuk Redis Versi 5.0.6 (Ditingkatkan). | 18 Desember 2019 |

| | | |
|---|---|---|
| Amazon ElastiCache sekarang mendukung node cache T3-Standar | Anda sekarang dapat meluncurkan node cache T3-Standard burstable serba guna generasi berikutnya di Amazon. ElastiCache Instans Standar T3 dari Amazon EC2 memberikan kinerja CPU di tingkat garis dasar dengan kemampuan untuk menangani lonjakan penggunaan CPU kapan pun hingga kredit terakumulasi habis. Untuk informasi selengkapnya, lihat Jenis Simpul yang Didukung. | 12 November 2019 |

| | | |
|---|---|---|
| Amazon ElastiCache sekarang mendukung modifikasi token AUTH pada yang sudah ada ElastiCache untuk server Redis | ElastiCache untuk Redis 5.0.6 sekarang memungkinkan Anda untuk memodifikasi token otentikasi dengan mengatur dan memutar token baru. Anda sekarang dapat mengubah token aktif saat sedang digunakan. Anda juga dapat menambahkan token baru ke klaster yang telah ada yang mengaktifkan enkripsi bergerak yang sebelumnya dipersiapkan tanpa token autentikasi. Ini adalah proses dua langkah di mana Anda dapat menetapkan dan merotasikan token tanpa mengganggu permintaan klien. Fitur ini saat ini tidak didukung pada AWS CloudFormation. Untuk informasi selengkapnya, lihat Mengautentikasi Pengguna dengan Perintah AUTH Redis. | 30 Oktober 2019 |
| Amazon ElastiCache sekarang mendukung migrasi data online dari Redis di Amazon EC2 | Anda sekarang dapat menggunakan Migrasi Online untuk memigrasikan data Anda dari Redis yang dihosting sendiri di Amazon EC2 ke Amazon. ElastiCache Untuk informasi selengkapnya, lihat Migrasi Online ke ElastiCache. | 28 Oktober 2019 |

| | | |
|---|---|---|
| ElastiCache untuk Redis memperkenalkan penskalaan vertikal online untuk mode Redis Cluster. | Anda sekarang dapat meningkatkan atau menurunkan skala Redis Cluster sharded Anda sesuai permintaan. ElastiCache untuk Redis mengubah ukuran cluster Anda dengan mengubah jenis node, sementara cluster terus online dan melayani permintaan yang masuk. Untuk informasi selengkapnya, lihat Penskalaan Vertikal Online dengan Mengubah Jenis Simpul. | 20 Agustus 2019 |
| ElastiCache untuk Redis sekarang memungkinkan pengguna untuk menggunakan titik akhir pembaca tunggal untuk Amazon ElastiCache Anda untuk kluster Redis. | Fitur ini memungkinkan Anda mengarahkan semua lalu lintas baca ke klaster Redis Anda ElastiCache melalui satu titik akhir tingkat cluster untuk memanfaatkan penyeimbangan beban dan ketersediaan yang lebih tinggi. Untuk informasi selengkapnya, silakan lihat Menemukan Titik Akhir Koneksi. | 13 Juni 2019 |

| | | |
|---|---|---|
| ElastiCache untuk Redis sekarang memungkinkan pengguna untuk menerapkan pembaruan layanan pada jadwal mereka sendiri | Dengan fitur ini, Anda dapat memilih untuk menerapkan pembaruan layanan yang tersedia pada saat yang Anda pilih dan tidak hanya selama periode pemeliharaan. Ini akan meminimalkan gangguan layanan, terutama selama arus bisnis puncak, dan membantu memastikan Anda tetap patuh jika klaster Anda berada dalam program kepatuhan yang ElastiCache didukung. Untuk informasi selengkapnya, lihat Pembaruan Layanan Mandiri di Amazon ElastiCache dan validasi Kepatuhan untuk Amazon. ElastiCache | 4 Juni 2019 |
| ElastiCache Penawaran Instans Cadangan Standar: Sebagian di Muka, Semua di Muka dan Tidak Ada di Muka. | Instans Cadangan memberi Anda fleksibilitas untuk memesan ElastiCache instans Amazon untuk jangka waktu satu atau tiga tahun berdasarkan jenis dan AWS Wilayah instans. Untuk informasi selengkapnya, lihat Mengelola Biaya dengan Simpul Terpesan. | 18 Januari 2019 |
| ElastiCache untuk dukungan Redis hingga 250 node per cluster Redis | Batas node atau shard dapat ditingkatkan hingga maksimum 250 per ElastiCache untuk cluster Redis. Untuk informasi selengkapnya, lihat Serpihan. | 19 November 2018 |

| | | |
|---|---|---|
| ElastiCache untuk dukungan Redis untuk autofailover dan backup dan restore pada semua node T2 | ElastiCache untuk Redis memperkenalkan dukungan untuk autofailover, membuat snapshot, dan backup dan restore pada semua node T2. Untuk informasi selengkap nya, lihat ElastiCache untuk Redis Backup and Restore dan Snapshot. | 19 November 2018 |
| ElastiCache untuk dukungan Redis untuk node M5 dan R5 | ElastiCache untuk Redis sekarang mendukung node M5 dan R5, tipe instans tujuan umum dan dioptimalkan memori berdasarkan Sistem Nitro. AWS Untuk informasi selengkapnya, lihat Jenis Simpul yang Didukung. | 23 Oktober 2018 |
| Dukungan untuk perubahan jumlah replika baca secara dinamis | ElastiCache untuk Redis telah menambahkan dukungan untuk menambahkan dan menghapus replika baca dari cluster mana pun tanpa waktu henti cluster. Untuk informasi selengkapnya tentang perubahan ini dan perubahan lainnya dalam rilis ini, lihat Mengubah Jumlah Replika di ElastiCache Panduan Pengguna Redis. Lihat juga DecreaseReplicaCountdan IncreaseReplicaCountdi Referensi ElastiCache API. | 17 September 2018 |

| | | |
|---|---|---|
| Sertifikasi kepatuhan FedRAMP | ElastiCache untuk Redis sekarang disertifikasi untuk kepatuhan FedRAMP. Untuk informasi selengkapnya, lihat Validasi kepatuhan untuk Amazon ElastiCache. | 30 Agustus 2018 |
| Pembaruan mesin Redis (mode klaster diaktifkan) | Amazon ElastiCache untuk Redis telah menambahkan dukungan untuk meningkatkan versi mesin Redis (mode cluster diaktifkan). Untuk informasi selengkapnya, lihat Meningkatkan Versi Mesin. | 20 Agustus 2018 |
| Sertifikasi kepatuhan PCI DSS | ElastiCache untuk Redis sekarang disertifikasi untuk kepatuhan PCI DSS. Untuk informasi selengkapnya, lihat Validasi kepatuhan untuk Amazon ElastiCache. | 5 Juli 2018 |
| Support ElastiCache untuk Redis 4.0.10 | ElastiCache untuk Redis sekarang mendukung Redis 4.0.10, termasuk enkripsi dan pengubahan ukuran cluster online dalam satu versi. Untuk informasi selengkapnya, lihat ElastiCache untuk Redis Versi 4.0.10 (Ditingkatkan). | 14 Juni 2018 |

| | | |
|---|---|---|
| Perubahan struktur Panduan Pengguna | Panduan ElastiCache Pengguna tunggal sekarang direstrukturisasi sehingga ada panduan pengguna terpisah untuk Redis (untuk Panduan Pengguna Redis) dan ElastiCache untuk Memcached (ElastiCache untuk Panduan Pengguna Memcached). Struktur dokumentasi di bagian AWS CLI Command Reference : elasticache dan Referensi Amazon ElastiCache API tetap tidak berubah. | 20 April 2018 |
| Dukungan untuk metrik EngineCPUUtilization | ElastiCache untuk Redis menambahkan metrik baru`EngineCPUUtilizati on` ,, yang melaporkan persentase kapasitas CPU Anda yang saat ini sedang digunakan. Untuk informasi selengkapnya, lihat Metrik untuk Redis. | 9 April 2018 |

Tabel berikut menjelaskan perubahan penting pada Panduan Pengguna Redis sebelum Maret 2018.

| Perubahan | Deskripsi | Tanggal Diubah |
|---|---|---|
| Dukungan untuk Wilayah Asia Pasifik (Osaka-lokal). | ElastiCache Menambahkan dukungan untuk Wilayah Asia Pasifik (Osaka-lokal). Wilayah Asia Pasifik (Osaka) saat ini mendukung Zona Ketersediaan tunggal dan hanya berdasarkan undangan saja. Untuk informasi selengkapnya, lihat hal berikut: | 12 Februari 2018 |

| Perubahan | Deskripsi | Tanggal Diubah |
| --- | --- | --- |
| | • Wilayah yang didukung<br>• Jenis simpul cache yang didukung | |
| Dukungan untuk EU (Paris). | ElastiCache menambahkan dukungan untuk Wilayah UE (Paris). Untuk informasi selengkapnya, lihat hal berikut:<br>• Wilayah yang didukung<br>• Jenis simpul cache yang didukung | 18 Desember 2017 |
| Dukungan untuk Wilayah Tiongkok (Ningxia) | Amazon ElastiCache menambahkan dukungan untuk Wilayah China (Ningxia). Untuk informasi selengkapnya, lihat hal berikut:<br>• Wilayah yang didukung<br>• Jenis simpul cache yang didukung | 11 Desember 2017 |
| Dukungan untuk Peran Tertaut Layanan | Rilis dukungan ElastiCache tambahan untuk Service Linked Roles (SLR) ini. Untuk informasi selengkapnya, lihat hal berikut:<br>• Menggunakan Peran Tertaut Layanan untuk Amazon ElastiCache<br>• Siapkan izin Anda (hanya ElastiCache pengguna baru) | 7 Desember 2017 |

| Perubahan | Deskripsi | Tanggal Diubah |
| --- | --- | --- |
| Dukungan untuk jenis simpul R4 | Rilis ini ElastiCache menambahkan jenis node R4 dukungan di semua AWS Wilayah yang didukung oleh ElastiCache. Anda dapat membeli jenis simpul R4 sebagai item Sesuai Permintaan atau sebagai Simpul Cache Terpesan. Untuk informasi selengkapnya, lihat hal berikut:<br>• Jenis simpul cache yang didukung<br>• Parameter khusus jenis simpul Redis | 20 November 2017 |
| ElastiCache untuk Redis 3.2.10 dan dukungan untuk resharding online | Amazon ElastiCache untuk Redis menambahk an dukungan ElastiCache untuk Redis 3.2.10. ElastiCache untuk Redis juga memperken alkan pengubahan ukuran cluster online untuk menambah atau menghapus pecahan dari cluster sambil terus melayani permintaan I/O yang masuk. Untuk informasi selengkapnya, lihat hal berikut:<br>• Perubahan ukuran klaster online<br>• Resharding dan penyeimbangan ulang serpihan secara online untuk Redis (mode klaster diaktifkan) | 9 November 2017 |
| Kelayakan HIPAA | ElastiCache untuk Redis versi 3.2.6 sekarang disertifikasi untuk kelayakan HIPAA saat enkripsi diaktifkan di cluster Anda. Untuk informasi selengkapnya, lihat hal berikut:<br>• Validasi kepatuhan untuk Amazon ElastiCache<br>• Keamanan data di Amazon ElastiCache | 2 November 2017 |

| Perubahan | Deskripsi | Tanggal Diubah |
| --- | --- | --- |
| ElastiCache untuk Redis 3.2.6 dan dukungan untuk enkripsi | ElastiCache menambahkan dukungan ElastiCache untuk Redis 3.2.6, yang mencakup dua fitur enkripsi:<br>• Enkripsi bergerak mengenkripsi data Anda setiap kali data bergerak dari satu tempat ke tempat lain, misalnya antara beberapa simpul di klaster atau antara klaster dan aplikasi Anda.<br>• Enkripsi diam mengenkripsi data pada disk Anda selama operasi sinkronisasi dan pencadangan.<br><br>Untuk informasi selengkapnya, lihat hal berikut:<br>• Keamanan data di Amazon ElastiCache<br>• Versi ElastiCache for Redis yang Didukung | 25 Oktober 2017 |
| Topik pola koneksi | ElastiCache dokumentasi menambahkan topik yang mencakup berbagai pola untuk mengakses ElastiCache cluster di VPC Amazon.<br><br>Untuk informasi lain, lihat Pola Akses untuk Mengakses Cache ElastiCache di Amazon VPC dalam Panduan Pengguna ElastiCache . | 24 April 2017 |

| Perubahan | Deskripsi | Tanggal Diubah |
| --- | --- | --- |
| Dukungan untuk menguji Failover Otomatis | ElastiCache menambahkan dukungan untuk menguji Failover Otomatis pada kluster Redis yang mendukung replikasi. Untuk informasi selengkap nya, lihat hal berikut:<br>• Menguji failover otomatis di Panduan Pengguna ElastiCache .<br>• TestFailover di Referensi API ElastiCache .<br>• test-failover di Referensi AWS CLI . | 4 April 2017 |
| Pemulihan Redis yang Ditingkat kan | ElastiCache menambahkan cadangan dan pemulihan Redis yang disempurnakan dengan mengubah ukuran cluster. Fitur ini mendukung pemulihan cadangan ke klaster dengan jumlah serpihan yang berbeda daripada klaster yang digunakan untuk membuat cadangan. (Untuk API dan CLI, fitur ini dapat memulihkan jumlah grup simpul yang berbeda, bukannya jumlah serpihan yang berbeda.) Pembaruan ini juga mendukung konfigurasi slot Redis yang berbeda. Untuk informasi selengkapnya, lihat Melakukan pemulihan dari cadangan ke dalam cache baru. | 15 Maret 2017 |
| Parameter baru manajemen memori Redis | ElastiCache menambahkan parameter Redis baru,`reserved-memory-percent` , yang membuat pengelolaan memori cadangan Anda lebih mudah. Parameter ini tersedia di semua versi ElastiCache untuk Redis. Untuk informasi selengkapnya, lihat hal berikut:<br>• Mengelola Memori Terpesan<br>• Parameter baru untuk Redis 3.2.4 | 15 Maret 2017 |

| Perubahan | Deskripsi | Tanggal Diubah |
| --- | --- | --- |
| Dukungan untuk Wilayah EU West (London) | ElastiCache menambahkan dukungan untuk Wilayah UE (London). Hanya jenis simpul T2 dan M4 yang didukung saat ini. Untuk informasi selengkapnya, lihat hal berikut:<br>• Wilayah yang didukung<br>• Jenis simpul cache yang didukung | 13 Desember 2016 |
| Dukungan untuk Wilayah Kanada (Montreal) | ElastiCache menambahkan dukungan untuk Wilayah Kanada (Montreal). Hanya tipe simpul M4 dan T2 yang saat ini didukung di Wilayah ini AWS . Untuk informasi selengkapnya, lihat hal berikut:<br>• Wilayah yang didukung<br>• Jenis simpul cache yang didukung | 8 Desember 2016 |
| Dukungan untuk jenis simpul M4 dan R3 | ElastiCache menambahkan dukungan untuk tipe node R3 dan M4 di Wilayah Amerika Selatan (São Paulo) dan tipe node M4 di Wilayah China (Beijing) . Untuk informasi selengkapnya, lihat hal berikut:<br>• Wilayah yang didukung<br>• Jenis simpul cache yang didukung | 1 November 2016 |
| Dukungan Wilayah AS Timur 2 (Ohio) | ElastiCache menambahkan dukungan untuk Wilayah Timur AS (Ohio) (us-timur-2) dengan tipe node M4, T2, dan R3. Untuk informasi selengkap nya, lihat hal berikut:<br>• Wilayah yang didukung<br>• Jenis simpul cache yang didukung | 17 Oktober 2016 |

| Perubahan | Deskripsi | Tanggal Diubah |
| --- | --- | --- |
| Dukungan untuk Klaster Redis | ElastiCache menambahkan dukungan untuk Redis Cluster (ditingkatkan). Pelanggan yang menggunakan Klaster Redis, dapat mempartisi datanya hingga 15 serpihan (grup simpul). Setiap serpihan mendukung replikasi maksimum hingga 5 replika baca per serpihan. Waktu failover otomatis Klaster Redis sekitar seperempat dari versi sebelumnya.<br><br>Rilis ini mencakup konsol manajemen yang dirancang ulang yang menggunakan terminologi yang mengikuti perkembangan penggunaan industri.<br><br>Untuk informasi selengkapnya, lihat hal berikut:<br><br>• Membandingkan Memcached dan Redis<br>• ElastiCache untuk komponen dan fitur Redis — perhatikan bagian pada Simpul, Serpihan, Klaster, dan Replikasi.<br>• Terminologi ElastiCache for Redis | 12 Oktober 2016 |
| Dukungan jenis simpul M4 | ElastiCache menambahkan dukungan untuk keluarga tipe node M4 di sebagian besar AWS Wilayah yang didukung olehElastiCache. Anda dapat membeli jenis simpul M4 sebagai item Sesuai Permintaan atau sebagai Simpul Cache Terpesan. Untuk informasi selengkapnya, lihat hal berikut:<br><br>• Jenis simpul cache yang didukung<br>• Parameter khusus jenis simpul Redis | 3 Agustus 2016 |

| Perubahan | Deskripsi | Tanggal Diubah |
| --- | --- | --- |
| Dukungan Wilayah Mumbai | ElastiCache Menambahkan dukungan untuk Wilayah Asia Pasifik (Mumbai). Untuk informasi selengkapnya, lihat hal berikut:<br>• Jenis simpul cache yang didukung<br>• Parameter khusus jenis simpul Redis | 27 Juni 2016 |
| Ekspor snapshot | ElastiCache menambahkan kemampuan untuk mengekspor snapshot Redis sehingga Anda dapat mengaksesnya dari luar. ElastiCache Untuk informasi selengkapnya, lihat hal berikut:<br>• Mengekspor cadangandi Panduan ElastiCache Pengguna Amazon<br>• CopySnapshotdi Referensi ElastiCache API Amazon | 26 Mei 2016 |
| Menaikkan skala jenis simpul | ElastiCache menambahkan kemampuan untuk meningkatkan jenis node Redis Anda. Untuk informasi selengkapnya, lihat Penskalaan ElastiCache untuk Redis . | 24 Maret 2016 |
| Peningkatan mesin yang mudah | ElastiCache menambahkan kemampuan untuk dengan mudah meningkatkan mesin cache Redis Anda. Untuk informasi selengkapnya, lihat Versi mesin dan pemutakhiran . | 22 Maret 2016 |
| Dukungan untuk jenis simpul R3 | ElastiCache menambahkan dukungan untuk tipe node R3 di Wilayah China (Beijing) dan Wilayah Amerika Selatan (São Paulo). Untuk informasi selengkapnya, lihat Jenis simpul cache yang didukung. | 16 Maret 2016 |

| Perubahan | Deskripsi | Tanggal Diubah |
| --- | --- | --- |
| Mengakses ElastiCache menggunakan fungsi Lambda | Menambahkan tutorial tentang mengonfigurasi fungsi Lambda untuk ElastiCache mengakses di VPC Amazon. Untuk informasi selengkapnya, lihat Tutorial dan video ElastiCache. | 12 Februari 2016 |
| Dukungan untuk Redis 2.8.24 | ElastiCache menambahkan dukungan untuk Redis versi 2.8.24 dengan peningkatan ditambahkan sejak Redis 2.8.23. Perbaikan mencakup perbaikan bug dan dukungan untuk membuat log untuk alamat akses memori yang buruk. Untuk informasi selengkapnya, lihat hal berikut:<br>• ElastiCache for Redis versi 2.8.24 (Ditingkatkan)<br>• Catatan Rilis Redis 2.8 | 20 Januari 2016 |
| Mendukung Wilayah Asia Pasifik (Seoul) | ElastiCache menambahkan dukungan untuk Wilayah Asia Pasifik (Seoul) (ap-timur laut-2) dengan tipe node t2, m3, dan r3. | 6 Januari 2016 |
| Perubahan ElastiCache konsol Amazon. | Karena versi Redis yang lebih baru memberikan pengalaman pengguna yang lebih baik dan lebih stabil, Redis versi 2.6.13, 2.8.6, dan 2.8.19 tidak lagi terdaftar di Konsol Manajemen. ElastiCache Untuk opsi lainnya dan informasi selengkapnya, lihat Versi ElastiCache for Redis yang Didukung. | 15 Desember 2015 |
| Dukungan untuk Redis 2.8.23. | ElastiCache menambahkan dukungan untuk Redis versi 2.8.23 dengan peningkatan ditambahkan sejak Redis 2.8.22. Perbaikan ini mencakup perbaikan bug dan dukungan untuk parameter baru `close-on-slave-write` yang, jika diaktifkan, memutuskan klien yang mencoba untuk menulis ke replika baca-saja. Untuk informasi selengkapnya, lihat ElastiCache for Redis versi 2.8.23 (Ditingkatkan). | 13 November 2015 |

| Perubahan | Deskripsi | Tanggal Diubah |
| --- | --- | --- |
| Dukungan untuk Redis 2.8.22. | ElastiCache menambahkan dukungan untuk Redis versi 2.8.22 dengan peningkatan dan peningkatan ElastiCache tambahan sejak versi 2.8.21. Peningkatan meliputi:<br>• Pelaksanaan proses menyimpan forkless yang memungkinkan penyimpanan yang berhasil saat memori tersedia rendah yang dapat menyebabkan penyimpanan dengan forked gagal.<br>• CloudWatch Metrik tambahan — SaveInProgressdan ReplicationBytes.<br>• Untuk mengaktifkan sinkronisasi parsial, parameter Redis `repl-backlog-size` sekarang berlaku untuk semua klaster.<br><br>Untuk daftar lengkap perubahan dan informasi selengkapnya, lihat ElastiCache for Redis versi 2.8.22 (Ditingkatkan).<br><br>Rilis dokumentasi ini mencakup reorganisasi dokumentasi dan penghapusan dokumentasi antarmuka baris ElastiCache perintah (CLI). Untuk penggunaan baris perintah, lihat AWS Baris Perintah untuk ElastiCache. | 28 September 2015 |
| Dukungan untuk Redis 2.8.21 | ElastiCache menambahkan dukungan untuk Redis versi 2.8.21 dan perbaikan Redis sejak versi 2.8.19. Rilis Redis ini mencakup beberapa perbaikan bug. Untuk informasi selengkapnya, lihat Catatan rilis Redis 2.8. | 29 Juli 2015 |

| Perubahan | Deskripsi | Tanggal Diubah |
| --- | --- | --- |
| Topik baru: Mengakses ElastiCache dari luar AWS | Menambahkan topik baru tentang cara mengakses ElastiCache sumber daya dari luar AWS. Untuk informasi selengkapnya, lihat Mengakses ElastiCache dari luar AWS. | 9 Juli 2015 |
| Pesan penggantian simpul ditambahkan | ElastiCache menambahkan tiga pesan yang berkaitan dengan penggantian node terjadwal ,ElastiCache:NodeReplacementScheduled,ElastiCache:NodeReplacementRescheduled, danElastiCache:NodeReplacementCanceled.<br><br>Untuk informasi dan tindakan selengkapnya yang dapat Anda lakukan saat node dijadwalkan untuk diganti, ElastiCache lihatNotifikasi Peristiwa dan Amazon SNS. | 11 Juni 2015 |

| Perubahan | Deskripsi | Tanggal Diubah |
| --- | --- | --- |
| Dukungan untuk Redis 2.8.19. | ElastiCache menambahkan dukungan untuk Redis versi 2.8.19 dan perbaikan Redis sejak versi 2.8.6. Dukungan ini mencakup dukungan untuk:<br>• Struktur HyperLogLog data, dengan perintah Redis PFADD, PFCOUNT, dan PFMERGE.<br>• Kueri rentang leksikografis dengan perintah baru ZRANGEBYLEX, ZLEXCOUNT, dan ZREMRANGEBYLEX.<br>• Memperkenalkan sejumlah perbaikan bug, yaitu mencegah simpul primer mengirim data yang sudah usang ke simpul replika dengan menggagalkan SYNC primer ketika proses anak untuk simpan di latar belakang (bgsave) berhenti tiba-tiba.<br><br>Untuk informasi lebih lanjut tentang HyperLogLog, lihat Redis struktur data baru: the. HyperLogLog<br><br>Untuk informasi lebih lanjut tentang PFADD, PFCOUNT, dan PFMERGE, lihat Dokumentasi Redis dan klik. HyperLogLog | 11 Maret 2015 |
| Dukungan untuk tanda alokasi biaya | ElastiCache menambahkan dukungan untuk tag alokasi biaya. Untuk informasi selengkapnya, lihat Memantau biaya dengan tanda alokasi biaya. | 9 Februari 2015 |
| Support untuk AWS GovCloud Wilayah (AS-Barat) | ElastiCache menambahkan dukungan untuk Wilayah AWS GovCloud (AS-Barat) (us-gov-west-1). | 29 Januari 2015 |

| Perubahan | Deskripsi | Tanggal Diubah |
| --- | --- | --- |
| Dukungan untuk Wilayah Eropa (Frankfurt) | ElastiCache menambahkan dukungan untuk Wilayah Eropa (Frankfurt) (eu-central-1). | 19 Januari 2015 |
| Dukungan Multi-AZ untuk grup replikasi Redis | ElastiCache menambahkan dukungan untuk Multi-AZ dari node utama ke replika baca dalam grup replikasi Redis. ElastiCache memantau kesehatan kelompok replikasi. Jika primer gagal, ElastiCache secara otomatis mempromosikan replika ke primer, kemudian menggantikan replika. Untuk informasi selengkapnya, lihat Meminimalkan waktu henti di ElastiCache for Redis dengan Multi-AZ. | 24 Oktober 2014 |
| AWS CloudTrail pencatatan panggilan API didukung | ElastiCache menambahkan dukungan untuk menggunakan AWS CloudTrail untuk mencatat semua panggilan ElastiCache API. Untuk informasi selengkapnya, lihat Pencatatan log panggilan API Amazon ElastiCache dengan AWS CloudTrail. | 15 September 2014 |
| Ukuran instans baru didukung | ElastiCache menambahkan dukungan untuk instance Tujuan Umum (T2) tambahan. Untuk informasi selengkapnya, lihat Mengonfigurasi parameter mesin menggunakan grup parameter. | 11 September 2014 |
| Ukuran instans baru didukung | ElastiCache menambahkan dukungan untuk instans Tujuan Umum (M3) tambahan dan instans Memory Optimized (R3). Untuk informasi selengkapnya, lihat Mengonfigurasi parameter mesin menggunakan grup parameter. | 1 Juli 2014 |

| Perubahan | Deskripsi | Tanggal Diubah |
| --- | --- | --- |
| Pencadangan dan Pemulihan untuk klaster Redis | Dalam rilis ini, ElastiCache memungkinkan pelanggan untuk membuat snapshot dari cluster Redis mereka, dan membuat cluster baru menggunakan snapshot ini. Cadangan adalah salinan klaster pada saat tertentu di suatu waktu, dan terdiri dari metadata klaster dan semua data di dalam cache Redis. Cadangan disimpan di Amazon S3, dan pelanggan dapat memulihkan data dari snapshot ke klaster baru setiap saat. Untuk informasi selengkapnya, lihat Melakukan snapshot dan pemulihan. | 24 April 2014 |
| Redis 2.8.6 | ElastiCache mendukung Redis 2.8.6, selain Redis 2.6.13. Dengan Redis 2.8.6, pelanggan dapat meningkatkan ketahanan dan toleransi kesalahan replika baca, dengan dukungan untuk resinkronisasi parsial, dan jumlah replika baca minimum yang ditetapkan pengguna yang harus tersedia setiap saat. Redis 2.8.6 juga menawarkan dukungan penuh untuk publish-and-subscribe, di mana klien dapat diberitahu tentang peristiwa yang terjadi di server. | 13 Maret 2014 |

| Perubahan | Deskripsi | Tanggal Diubah |
| --- | --- | --- |
| Mesin cache Redis | ElastiCache menawarkan perangkat lunak mesin cache Redis, selain Memcached. Pelanggan yang saat ini menggunakan Redis dapat “menyemai ” cluster cache ElastiCache Redis baru dengan data mereka yang ada dari file snapshot Redis, memudahkan migrasi ke lingkungan terkelola. ElastiCache<br><br>Untuk mendukung kemampuan replikasi Redis, ElastiCache API sekarang mendukung grup replikasi. Pelanggan dapat membuat grup replikasi dengan simpul cache Redis primer, dan menambahkan satu atau lebih simpul replika baca yang secara otomatis tetap disinkronkan dengan data cache di simpul primer. Aplikasi yang intensif membaca dapat dilepaskan ke replika baca, mengurangi beban pada simpul primer. Replika baca juga dapat menjaga terhadap kehilangan data dalam hal peristiwa kegagalan simpul cache primer. | 3 September 2013 |
| Dukungan Cloud Privat Virtual (VPC) Amazon default | Dalam rilis ElastiCache ini, terintegrasi penuh dengan Amazon Virtual Private Cloud (VPC). Untuk pelanggan baru, klaster cache dibuat dalam Amazon VPC secara default. Untuk informasi selengkapnya, lihat Amazon VPC dan keamanan ElastiCache. | 8 Januari 2013 |

| Perubahan | Deskripsi | Tanggal Diubah |
| --- | --- | --- |
| Dukungan Cloud Privat Virtual (VPC) Amazon | Dalam rilis ini, ElastiCache cluster dapat diluncurkan di Amazon Virtual Private Cloud (VPC). Secara default, klaster cache pelanggan baru dibuat di Amazon VPC secara otomatis; pelanggan yang sudah ada dapat bermigrasi ke Amazon VPC sesuai kebutuhannya sendiri. Untuk informasi selengkapnya, lihat Amazon VPC dan keamanan ElastiCache. | 20 Desember 2012 |
| Jenis baru simpul cache | Rilis ini menyediakan empat jenis simpul cache tambahan. | 13 November 2012 |
| Simpul cache terpesan | Rilis ini menambahkan dukungan untuk simpul cache terpesan. | 5 April 2012 |
| Panduan baru | Ini adalah rilis pertama Panduan ElastiCache Pengguna Amazon. | 22 Agustus 2011 |

# AWS Glosarium

Untuk AWS terminologi terbaru, lihat AWS glosarium di Referensi.Glosarium AWS

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.