aws

Panduan Developer

Amazon MemoryDB for Redis

Amazon MemoryDB for Redis: Panduan Developer

Table of Contents

Apa itu MemoryDB untuk Redis?

MemoryDB for Redis adalah layanan database dalam memori yang tahan lama yang memberikan kinerja ultra-cepat. Ini dibuat khusus untuk aplikasi modern dengan arsitektur layanan mikro.

MemoryDB kompatibel dengan Redis, penyimpanan data open source yang populer, memungkinkan Anda untuk dengan cepat membangun aplikasi menggunakan struktur data Redis yang fleksibel dan ramah yang sama, API, dan perintah yang sudah mereka gunakan saat ini. Dengan MemoryDB, semua data Anda disimpan dalam memori, yang memungkinkan Anda mencapai baca mikrodetik dan latensi penulisan milidetik satu digit dan throughput tinggi. MemoryDB juga menyimpan data secara tahan lama di beberapa Availability Zones (AZ) menggunakan log transaksional multi-AZ untuk mengaktifkan failover cepat, pemulihan database, dan restart node.

Memberikan kinerja dalam memori dan daya tahan multi-AZ, MemoryDB dapat digunakan sebagai database utama berkinerja tinggi untuk aplikasi layanan mikro Anda, menghilangkan kebutuhan untuk mengelola cache dan database yang tahan lama secara terpisah.

Topik

Fitur MemoryDB

MemoryDB for Redis adalah layanan database dalam memori yang tahan lama yang memberikan kinerja ultra-cepat. Fitur MemoryDB meliputi:

- Konsistensi yang kuat untuk node primer dan konsistensi akhirnya dijamin untuk node replika. Untuk informasi selengkapnya, lihat Konsistensi.
- Latensi baca mikrodetik dan latensi tulis milidetik satu digit dengan hingga 160 juta TPS per cluster.
- Struktur dan API data Redis yang fleksibel dan ramah. Mudah membangun aplikasi baru atau memigrasikan aplikasi Redis yang ada tanpa modifikasi.

- Daya tahan data menggunakan log transaksional multi-AZ yang menyediakan pemulihan database yang cepat dan restart.
- Ketersediaan multi-AZ dengan failover otomatis, dan deteksi dan pemulihan dari kegagalan node.
- Skala secara horizontal dengan mudah dengan menambahkan dan menghapus node atau secara vertikal dengan pindah ke tipe node yang lebih besar atau lebih kecil. Anda dapat menskalakan throughput tulis dengan menambahkan pecahan dan skala throughput baca dengan menambahkan replika.
- ead-after-write Konsistensi R untuk node primer dan konsistensi akhirnya dijamin untuk node replika.
- MemoryDB mendukung enkripsi dalam perjalanan, enkripsi saat istirahat dan otentikasi pengguna melalui. Mengautentikasi pengguna dengan Daftar Kontrol Akses (ACL)
- Snapshot otomatis di Amazon S3 dengan retensi hingga 35 hari.
- Support hingga 500 node dan penyimpanan lebih dari 100 TB per cluster (dengan 1 replika per pecahan).
- Enkripsi dalam perjalanan dengan TLS dan enkripsi saat istirahat dengan kunci. AWS KMS
- Otentikasi dan otorisasi pengguna dengan Redis. Mengautentikasi pengguna dengan Daftar Kontrol Akses (ACL)
- Support untuk jenis AWS instans Graviton2.
- Integrasi dengan AWS layanan lain seperti CloudWatch, Amazon VPC, CloudTrail, dan Amazon SNS untuk pemantauan, keamanan, dan pemberitahuan.
- Penambalan dan peningkatan perangkat lunak yang dikelola sepenuhnya.
- AWSIntegrasi Identity and Access Management (IAM) dan kontrol akses berbasis tag untuk API manajemen.

Komponen inti MemoryDB

Berikut ini, Anda dapat menemukan ikhtisar komponen utama dari penyebaran MemoryDB.

Topik

- Klaster
- Simpul
- Serpihan
- Grup parameter

- Grup Subjaringan
- Daftar Kontrol Akses
- Pengguna

Klaster

Sebuah klaster adalah kumpulan satu simpul atau lebih yang melayani satu set data tunggal. Sebuah dataset MemoryDB dipartisi menjadi pecahan, dan setiap pecahan memiliki node primer dan hingga 5 node replika opsional. Node primer melayani permintaan baca dan tulis, sementara replika hanya melayani permintaan baca. Sebuah node primer dapat failover ke node replika, mempromosikan replika itu ke node primer baru untuk pecahan itu. MemoryDB menjalankan Redis sebagai mesin database, dan ketika Anda membuat cluster, Anda menentukan versi Redis untuk klaster Anda. Anda dapat membuat dan memodifikasi klaster menggunakanAWS CLI, API MemoryDB, atauAWS Management Console.

Setiap klaster MemoryDB menjalankan versi mesin Redis. Setiap versi mesin Redis memiliki fitur yang didukungnya sendiri. Selain itu, setiap mesin Redis memiliki sekumpulan parameter dalam grup parameter yang mengontrol perilaku klaster yang dikelolanya.

Komputasi dan kapasitas memori dari sebuah klaster ditentukan oleh tipe simpul -nya. Anda dapat memilih jenis instans yang paling sesuai untuk kebutuhan Anda. Jika kebutuhan Anda berubah seiring waktu, Anda dapat mengubah jenis simpul. Untuk informasi, lihat Tipe simpul yang didukung.

> **Note**
>
> Untuk informasi harga pada jenis node MemoryDB, lihatHarga MemoryDB.

Anda menjalankan klaster pada Virtual Private Cloud (VPC) menggunakan layanan Amazon Virtual Private Cloud (Amazon VPC). Saat Anda menggunakan VPC, Anda dapat mengontrol lingkungan jaringan virtual Anda. Anda dapat memilih rentang alamat IP Anda sendiri, membuat subnet, dan mengonfigurasi perutean dan access control list. MemoryDB mengelola snapshot, patching perangkat lunak, deteksi kegagalan otomatis, dan pemulihan. Tidak ada biaya tambahan untuk menjalankan klaster Anda di dalam VPC. Untuk informasi selengkapnya tentang penggunaan Amazon VPC dengan MemoryDB, lihatMemoryDB dan Amazon VPC.

Banyak operasi MemoryDB ditargetkan pada cluster:

- Membuat klaster
- Mengubah klaster
- Mengambil snapshot dari sebuah cluster
- Menghapus klaster
- Melihat elemen di klaster
- Menambahkan atau menghapus tag alokasi biaya ke dan dari klaster

Untuk informasi lebih detail, lihat topik terkait berikut:

- Mengelola Klaster dan Mengelola simpul

  Informasi tentang klaster, simpul, dan operasi yang terkait.
- Ketahanan dalam MemoryDB untuk Redis

  Informasi tentang peningkatan toleransi kesalahan klaster Anda.

Simpul

SEBUAHsimpuladalah blok bangunan terkecil dari penerapan MemoryDB dan berjalan menggunakan instans Amazon EC2. Setiap node menjalankan versi Redis yang dipilih saat Anda membuat klaster Anda. Sebuah node milik pecahan yang milik cluster.

Setiap node menjalankan instance mesin pada versi yang dipilih saat Anda membuat klaster Anda. Jika perlu, Anda dapat skala node dalam cluster atas atau ke bawah ke jenis yang berbeda. Untuk informasi selengkapnya, lihat Penskalaan .

Setiap simpul dalam sebuah klaster adalah tipe simpul yang sama. Beberapa jenis node didukung, masing-masing dengan jumlah memori yang bervariasi. Untuk mengetahui daftar jenis data yang didukung, lihat Tipe simpul yang didukung.

Untuk informasi lain tentang simpul, lihat Mengelola simpul.

Serpihan

Shard adalah pengelompokan satu sampai 6 node, dengan satu melayani sebagai node tulis utama dan 5 lainnya berfungsi sebagai replika baca. Cluster MemoryDB selalu memiliki setidaknya satu beling.

Cluster MemoryDB dapat memiliki pecahan hingga 500, dengan data Anda dipartisi di seluruh pecahan. 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.

SEBUAHbeberapa simpul belingmengimplementasikan replikasi dengan memiliki satu membaca/menulis node primer dan 1-5 node replika. Untuk informasi selengkapnya, lihat Memahami replikasi MemoryDB.

Untuk informasi lebih lanjut tentang serpihan, lihat Bekerja dengan Serpihan.

Grup parameter

Grup parameter adalah cara mudah untuk mengelola pengaturan runtime untuk Redis di klaster Anda. Parameter digunakan untuk mengontrol penggunaan memori, ukuran item, dan banyak lagi. Sebuah kelompok parameter MemoryDB adalah kumpulan bernama parameter khusus mesin yang dapat Anda terapkan ke cluster, dan semua node dalam cluster yang dikonfigurasi dengan cara yang persis sama.

Untuk informasi selengkapnya tentang grup parameter MemoryDB, lihatMengonfigurasi parameter mesin menggunakan grup parameter.

Grup Subjaringan

SEBUAHGrup subnetadalah kumpulan subnet (biasanya private) yang dapat Anda tetapkan untuk klaster Anda yang berjalan di lingkungan Amazon Virtual Private Cloud (VPC).

Saat membuat klaster di Amazon VPC, Anda dapat menentukan grup subnet atau menggunakan grup default yang disediakan. MemoryDB menggunakan grup subnet dan IP dalam subnet tersebut untuk mengasosiasikan dengan simpul Anda.

Untuk informasi lebih rinci tentang grup subnet MemoryDB, lihatSubnet dan grup subnet.

Daftar Kontrol Akses

Daftar kontrol Akses adalah kumpulan satu pengguna atau lebih. Akses string mengikuti RedisAturan ACLuntuk mengotorisasi akses pengguna ke perintah Redis dan data.

Untuk informasi lebih rinci tentang MemoryDB Access Control Lists, lihatMengautentikasi pengguna dengan Daftar Kontrol Akses (ACL).

Pengguna

Pengguna memiliki nama pengguna dan kata sandi, dan digunakan untuk mengakses data dan mengeluarkan perintah pada klaster MemoryDB Anda. Pengguna adalah anggota dari Access Control List (ACL), yang dapat Anda gunakan untuk menentukan izin bagi pengguna tersebut di klaster MemoryDB. Untuk informasi selengkapnya, lihatMengautentikasi pengguna dengan Daftar Kontrol Akses (ACL)

Layanan-layanan terkait

ElastiCache untuk Redis

Saat memutuskan apakah akan menggunakan MemoryDB untuk Redis atau ElastiCache untuk Redis pertimbangkan perbandingan berikut:

- MemoryDB for Redis adalah database dalam memori yang tahan lama untuk beban kerja yang membutuhkan database primer yang sangat cepat. Anda harus mempertimbangkan untuk menggunakan MemoryDB jika beban kerja Anda memerlukan database tahan lama yang memberikan kinerja sangat cepat (baca mikrodetik dan latensi penulisan milidetik satu digit). MemoryDB mungkin juga cocok untuk kasus penggunaan Anda jika Anda ingin membangun aplikasi menggunakan struktur data Redis dan API dengan database primer yang tahan lama. Terakhir, Anda harus mempertimbangkan untuk menggunakan MemoryDB untuk menyederhanakan arsitektur aplikasi Anda dan menurunkan biaya dengan mengganti penggunaan database dengan cache untuk daya tahan dan kinerja.
- ElastiCache untuk Redis adalah layanan yang biasa digunakan untuk cache data dari database lain dan penyimpanan data menggunakan Redis. Anda harus mempertimbangkan Redis ElastiCache untuk beban kerja caching di mana Anda ingin mempercepat akses data dengan database utama atau penyimpanan data yang ada (kinerja baca dan tulis mikrodetik). Anda juga harus mempertimbangkan ElastiCache Redis untuk kasus penggunaan di mana Anda ingin menggunakan struktur data Redis dan API untuk mengakses data yang disimpan dalam database utama atau penyimpanan data.

Memilih Wilayah dan Availability Zone

Sumber daya komputasi AWS Cloud disimpan di fasilitas pusat data dengan ketersediaan tinggi. Untuk memberikan skalabilitas dan keandalan tambahan, fasilitas pusat data ini ditempatkan di beberapa lokasi fisik yang berbeda. Lokasi ini dikategorikan berdasarkan wilayah dan Availability Zones.

Wilayah AWS besar dan tersebar meluas ke dalam beberapa lokasi geografis terpisah. Availability Zone adalah beberapa lokasi berbeda di dalam sebuah Wilayah AWS yang direkayasa sedemikian rupa agar terisolasi dari kegagalan di Availability Zone yang lain. Availability Zone menyediakan konektivitas jaringan murah berlatensi rendah untuk Availability Zone lain di Wilayah AWS yang sama.

> **Important**
>
> Setiap wilayah adalah independen sepenuhnya. Aktivitas MemoryDB apa pun yang Anda mulai (misalnya, membuat cluster) hanya berjalan di wilayah default Anda saat ini.

Untuk membuat atau bekerja dengan klaster di wilayah tertentu, gunakan titik akhir layanan regional yang terkait. Untuk titik akhir layanan, lihat Wilayah & titik akhir yang didukung.

Menempatkan simpul Anda

Setiap cluster yang memiliki setidaknya satu replika harus tersebar di seluruh AZ. Satu-satunya cara Anda dapat menemukan semuanya dalam satu AZ adalah dengan cluster yang terdiri dari pecahan simpul tunggal.

Dengan menemukan node di AZ yang berbeda, MemoryDB menghilangkan kemungkinan kegagalan, seperti pemadaman listrik, dalam satu AZ akan menyebabkan hilangnya ketersediaan.

- Membuat cluster MemoryDB
- Memodifikasi cluster MemoryDB

Wilayah & titik akhir yang didukung

MemoryDB untuk Redis tersedia di beberapa Wilayah. AWS Ini berarti Anda dapat meluncurkan cluster MemoryDB di lokasi yang memenuhi kebutuhan Anda. Misalnya, Anda dapat meluncurkan di Wilayah AWS yang terdekat dengan pelanggan Anda atau meluncurkan di Wilayah AWS tertentu untuk memenuhi persyaratan legal tertentu. Selain itu, karena MemoryDB memperluas ketersediaan ke AWS Wilayah baru, MemoryDB mendukung dua `MAJOR.MINOR` versi terbaru pada waktu itu untuk Wilayah baru. Untuk informasi lebih lanjut tentang versi MemoryDB, lihat. Versi mesin Redis

Secara default, AWS SDK, MemoryDB APIAWS CLI, dan konsol MemoryDB mengacu pada Wilayah AS-Timur (Virginia N.). Saat MemoryDB memperluas ketersediaan ke wilayah baru, titik akhir baru untuk wilayah ini juga tersedia untuk digunakan dalam permintaan HTTP, AWS SDK, AWS CLI dan konsol Anda.

Setiap Wilayah dirancang untuk terisolasi sepenuhnya dari Wilayah lain. Di dalam setiap wilayah terdapat beberapa Availability Zone (AZ). Dengan meluncurkan node Anda di AZ yang berbeda, Anda mencapai toleransi kesalahan sebesar mungkin. Untuk informasi selengkapnya tentang wilayah dan Availability Zone, Memilih Wilayah dan Availability Zone lihat di awal topik ini.

Wilayah di mana MemoryDB didukung

Nama Wilayah/Wilayah	Titik akhir	Protokol	
Wilayah AS Timur (Ohio) `us-east-2`	`memory-db.us-east-2.amazonaws.com`	HTTPS	
Wilayah AS Timur (Virginia Utara) `us-east-1`	`memory-db.us-east-1.amazonaws.com`	HTTPS	
Wilayah AS Barat (California Utara) `us-west-1`	`memory-db.us-west-1.amazonaws.com`	HTTPS	

Nama Wilayah/Wilayah	Titik akhir	Protokol	
Wilayah AS Barat (Oregon) `us-west-2`	`memory-db.us-west-2.amazonaws.com`	HTTPS	
Wilayah Kanada (Pusat) `ca-central-1`	`memory-db.ca-central-1.amazonaws.com`	HTTPS	
Wilayah Asia Pasifik (Hong Kong) `ap-east-1`	`memory-db.ap-eastl-1.amazonaws.com`	HTTPS	
Wilayah Asia Pasifik (Mumbai) `ap-south-1`	`memory-db.ap-south-1.amazonaws.com`	HTTPS	
Wilayah Asia Pasifik (Tokyo) `ap-northeast-1`	`memory-db.ap-northeast-1.amazonaws.com`	HTTPS	
Wilayah Asia Pasifik (Seoul) `ap-northeast-2`	`memory-db.ap-northeast-2.amazonaws.com`	HTTPS	
Wilayah Asia Pasifik (Singapura) `ap-southeast-1`	`memory-db.ap-southeast-1.amazonaws.com`	HTTPS	

Nama Wilayah/Wilayah	Titik akhir	Protokol	
Wilayah Asia Pasifik (Sydney) `ap-southeast-2`	`memory-db.ap-southeast-2.amazonaws.com`	HTTPS	
Wilayah Eropa (Frankfurt) `eu-central-1`	`memory-db.eu-central-1.amazonaws.com`	HTTPS	
Wilayah Eropa (Irlandia) `eu-west-1`	`memory-db.eu-west-1.amazonaws.com`	HTTPS	
Wilayah Eropa (London) `eu-west-2`	`memory-db.eu-west-2.amazonaws.com`	HTTPS	
Wilayah EU (Paris) `eu-west-3`	`memory-db.eu-west-3.amazonaws.com`	HTTPS	
Wilayah Eropa (Stockholm) `eu-north-1`	`memory-db.eu-north-1.amazonaws.com`	HTTPS	
Wilayah Eropa (Milan) `eu-south-1`	`memory-db.eu-south-1.amazonaws.com`	HTTPS	
Wilayah Amerika Selatan (Sao Paulo) `sa-east-1`	`memory-db.sa-east-1.amazonaws.com`	HTTPS	

Nama Wilayah/Wilayah	Titik akhir	Protokol	
Wilayah Tiongkok (Beijing) `cn-north-1`	`memory-db.cn-north-1.amazonaws.com.cn`	HTTPS	
Wilayah Tiongkok (Ningxia) `cn-northwest-1`	`memory-db.cn-northwest-1.amazonaws.com.cn`	HTTPS	

Untuk tabel AWS produk dan layanan menurut wilayah, lihat Produk dan layanan menurut Wilayah.

Untuk tabel Availability Zone yang didukung dalam Wilayah, lihatSubnet dan grup subnet.

Mengakses MemoryDB

Setiap titik akhir cluster MemoryDB berisi alamat dan port. Endpoint cluster ini mendukung protokol Redis Cluster untuk memungkinkan klien menemukan peran, alamat ip, dan slot tertentu untuk setiap node di cluster. Ketika node primer gagal dan replika dipromosikan sebagai gantinya, Anda dapat terhubung ke titik akhir cluster untuk menemukan primer baru menggunakan protokol Redis Cluster.

Anda perlu terhubung ke titik akhir cluster untuk menemukan titik akhir node menggunakan cluster nodes atau cluster slots perintah. Setelah menemukan node yang tepat untuk kunci, Anda dapat terhubung langsung ke node untuk permintaan baca/tulis. Klien Redis dapat menggunakan titik akhir cluster untuk secara otomatis terhubung ke node yang benar.

Untuk memecahkan masalah node tertentu dalam klaster, Anda juga dapat menggunakan titik akhir khusus node, tetapi ini tidak diperlukan untuk penggunaan normal.

Untuk menemukan titik akhir cluster, lihat berikut ini:

- Menemukan Endpoint untuk Cluster MemoryDB (AWSCLI)
- Menemukan Endpoint untuk Cluster MemoryDB (MemoryDB API)

Untuk menghubungkan ke node atau cluster, lihatMenghubungkan ke node MemoryDB menggunakan redis-cli.

Keamanan MemoryDB

Keamanan untuk MemoryDB dikelola pada tiga tingkatan:

- Untuk mengontrol siapa yang dapat melakukan tindakan manajemen pada cluster dan node MemoryDB, Anda menggunakan AWS Identity and Access Management (IAM). Saat Anda terhubung AWS menggunakan kredensyal IAM, AWS akun Anda harus memiliki kebijakan IAM yang memberikan izin yang diperlukan untuk melakukan operasi. Untuk informasi selengkapnya, silakan lihat Manajemen identitas dan akses di MemoryDB untuk Redis
- Untuk mengontrol tingkat akses ke cluster, Anda membuat pengguna dengan izin tertentu dan menetapkannya ke Daftar Kontrol Akses (ACL). ACL, pada gilirannya, kemudian dikaitkan dengan satu atau lebih cluster. Untuk informasi selengkapnya, lihat Mengautentikasi pengguna dengan Daftar Kontrol Akses (ACL).
- Cluster MemoryDB harus dibuat di cloud pribadi virtual (VPC) berdasarkan layanan Amazon VPC. Untuk mengontrol perangkat dan instans Amazon EC2 mana yang dapat membuka koneksi ke titik akhir dan port node untuk cluster MemoryDB di VPC, Anda menggunakan grup keamanan VPC. Anda dapat membuat endpoint dan koneksi port ini menggunakan Transport Layer Security (TLS)/Secure Sockets Layer (SSL). Selain itu, aturan firewall di perusahaan Anda dapat mengontrol apakah perangkat yang berjalan di perusahaan Anda dapat membuka koneksi ke cluster MemoryDB. Untuk informasi selengkapnya tentang VPC, lihat MemoryDB dan Amazon VPC.

Untuk informasi tentang konfigurasi keamanan, lihat Keamanan di MemoryDB untuk Redis.

Memulai dengan MemoryDB

Latihan ini mengarahkan Anda melalui langkah-langkah untuk membuat, memberikan akses ke, terhubung ke, dan akhirnya menghapus cluster MemoryDB menggunakan MemoryDB Management Console.

> **Note**
>
> Untuk tujuan latihan ini, kami sarankan Anda menggunakan opsi Easy create saat membuat cluster dan kembali ke dua opsi lainnya setelah Anda menjelajahi lebih lanjut fitur MemoryDB.

Topik

Mengatur

Berikut ini, Anda dapat menemukan topik yang menggambarkan tindakan satu kali yang harus Anda ambil untuk mulai menggunakan MemoryDB.

Topik

Buat akun AWS Anda

Daftar Akun AWS

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

Untuk mendaftar 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 Akun AWS, Pengguna root akun AWS dibuat. 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 akan mengirimkan email konfirmasi kepada Anda 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 mendaftarAkun AWS, amankanPengguna root akun AWS, aktifkanAWS IAM Identity Center, dan buat pengguna administratif sehingga Anda tidak menggunakan pengguna root untuk tugas sehari-hari.

Mengamankan Pengguna root akun AWS Anda

1. Masuk ke AWS Management Console sebagai pemilik akun dengan memilih Pengguna root dan memasukkan alamat email Akun AWS Anda. Pada halaman berikutnya, masukkan kata sandi Anda.

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

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

Membuat pengguna administratif

1. Aktifkan Pusat Identitas IAM.

   Untuk petunjuk, lihat Mengaktifkan AWS IAM Identity Center di Panduan AWS IAM Identity Center Pengguna.

2. Di Pusat Identitas IAM, berikan akses administratif ke 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 akses AWS dalam Panduan Pengguna AWS Sign-In.

Berikan akses terprogram

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

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

Pengguna mana yang membutuhkan akses terprogram?	Ke	Oleh
Identitas tenaga kerja (Pengguna yang dikelola di Pusat Identitas IAM)	Gunakan kredensial sementara untuk menandatangani permintaan terprogram ke AWS CLI, SDK AWS, atau API AWS.	Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan. • Untuk AWS CLI, lihat Mengonfigurasi AWS CLI untuk menggunakan AWS IAM Identity Center di Panduan Pengguna AWS Command Line Interface. • Untuk SDK AWS, alat, dan API AWS, lihat Autentikasi Pusat Identitas IAM di Panduan Referensi SDK dan Alat AWS.
IAM	Gunakan kredensial sementara untuk menandatangani permintaan terprogram ke AWS CLI, SDK AWS, atau API AWS.	Ikuti petunjuk dalam Menggunakan kredensial sementara dengan sumber daya AWS di Panduan Pengguna IAM.
IAM	(Tidak disarankan) Gunakan kredensial jangka panjang untuk menandatangani permintaan terprogram ke AWS CLI, AWS SDK, atau API AWS.	Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan. • Untuk AWS CLI, lihat Mengautentikasi menggunakan kredensial pengguna IAM di Panduan Pengguna AWS Command Line Interface.

Pengguna mana yang membutuhkan akses terprogram?	Ke	Oleh
		• Untuk SDK dan alat AWS, lihat Mengautentikasi menggunakan kredensial jangka panjang di Panduan Referensi SDK dan Alat AWS. • Untuk API AWS, lihat Mengelola kunci akses untuk pengguna IAM di Panduan Pengguna IAM.

Topik-topik terkait:

- Apa itu IAM? dalam Panduan Pengguna IAM.
- Kredensial keamanan AWS dalam Referensi Umum AWS.

Siapkan izin Anda (hanya pengguna MemoryDB baru)

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

- Pengguna dan grup di AWS IAM Identity Center:

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

  Buat peran untuk federasi identitas. Ikuti instruksi dalam Membuat peran untuk penyedia identitas pihak ketiga (federasi) di Panduan Pengguna IAM.
- Pengguna IAM:
  - Buat peran yang dapat diambil pengguna Anda. Ikuti instruksi dalam Membuat peran untuk pengguna IAM dalam Panduan Pengguna IAM.

- (Tidak disarankan) Lampirkan kebijakan langsung ke pengguna atau tambahkan pengguna ke grup pengguna. Ikuti instruksi dalam Menambahkan izin ke pengguna (konsol) dalam Panduan Pengguna IAM.

MemoryDB for Redis membuat dan menggunakan peran terkait layanan untuk menyediakan sumber daya dan mengakses sumber daya dan layanan lain atas AWS nama Anda. Agar MemoryDB membuat peran terkait layanan untuk Anda, gunakan kebijakan -managed bernama. AWS `AmazonMemoryDBFullAccess` 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 menelepon `iam:createServiceLinkedRole` atau bahwa Anda telah membuat peran terkait layanan MemoryDB.

Untuk informasi lain, lihat yang berikut ini:

- Membuat Kebijakan Baru (IAM)
- AWSKebijakan (yang telah ditentukan sebelumnya) untuk MemoryDB untuk Redis
- Menggunakan Peran Tertaut Layanan untuk Amazon MemoryDB untuk Redis

Mengunduh dan Membuat Konfigurasi CLI AWS

AWS CLI tersedia di http://aws.amazon.com/cli. CLI berjalan di Windows, MacOS, atau Linux. Setelah Anda mengunduh AWS CLI, ikuti langkah ini untuk menginstal dan membuat konfigurasinya:

1. Lihat ke Panduan Pengguna Antar Muka Baris Perintah AWS.
2. Ikuti petunjuk untuk Menginstal CLI AWS dan Membuat Konfigurasi CLI AWS.

Langkah 1: Buat cluster

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 dapat menerima nilai konfigurasi default yang diterapkan.

Klaster yang Anda buat akan berjalan langsung, dan tidak berjalan di sandbox. Anda akan dikenakan biaya penggunaan MemoryDB standar untuk instance 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 tingkat penggunaan MemoryDB, lihat MemoryDB.

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

Membuat cluster MemoryDB

Contoh berikut menunjukkan cara membuat cluster menggunakan AWS Management Console, AWS CLI dan MemoryDB API.

Membuat cluster (Konsol)

Untuk membuat cluster menggunakan konsol MemoryDB

1. Masuk ke AWS Management Console dan buka konsol MemoryDB for Redis di https://console.aws.amazon.com/memorydb/.
2. Pilih Cluster Di panel navigasi kiri dan kemudian pilih Buat.

Easy create

1. Lengkapi bagian Konfigurasi. Ini mengonfigurasi tipe node dan konfigurasi default cluster Anda. Pilih ukuran memori dan kinerja jaringan yang sesuai yang Anda butuhkan dari opsi berikut:
   - Produksi
   - Dev/Uji
   - Demo
2. Lengkapi bagian Info Cluster.

a. Pada Nama, masukkan nama untuk klaster Anda.

   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.

b. Pada kotak Deskripsi, masukkan deskripsi untuk klaster ini.

3. Lengkapi bagian grup Subnet:

   - Untuk grup Subnet, buat grup subnet baru atau pilih yang sudah ada dari daftar yang tersedia yang ingin Anda terapkan ke cluster ini. Jika Anda membuat yang baru:

     - Masukkan Nama
     - Masukkan Deskripsi
     - Jika Anda mengaktifkan Multi-AZ, grup subnet harus berisi setidaknya dua subnet yang berada dalam availability zone yang berbeda. Untuk informasi selengkapnya, lihat Subnet dan grup subnet.
     - Jika Anda membuat grup subnet baru dan tidak memiliki VPC yang ada, Anda akan diminta untuk membuat VPC. Untuk informasi lebih lanjut, lihat Apa itu Amazon VPC? di Panduan Pengguna Amazon VPC.

4. Untuk pencarian Vektor, Anda dapat Aktifkan kemampuan pencarian Vektor untuk menyimpan embeddings vektor dan melakukan pencarian vektor. Perhatikan bahwa ini akan memperbaiki nilai untuk kompatibilitas versi Redis, grup Parameter dan Pecahan. Untuk informasi selengkapnya, lihat Pencarian vektor.

5. Lihat pengaturan default:

   Saat menggunakan Easy create, pengaturan cluster yang tersisa diatur secara default. Perhatikan bahwa beberapa pengaturan ini dapat diubah setelah pembuatan, seperti yang ditunjukkan oleh Editable setelah pembuatan.

6. Untuk Tag, Anda dapat menerapkan tag secara opsional untuk mencari dan memfilter cluster Anda atau melacak biaya Anda AWS .

7. Tinjau semua entri dan pilihan Anda, lalu lakukan koreksi yang diperlukan. Saat Anda siap, pilih Buat untuk meluncurkan klaster Anda, atau Batal untuk membatalkan operasi.

Setelah status klaster Anda menjadi tersedia, Anda dapat memberikan akses ke EC2, menyambungkannya, dan mulai menggunakannya. Lihat informasi yang lebih lengkap di Langkah 2: Otorisasi akses ke cluster

> **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 Langkah 4: Menghapus cluster.

Create new cluster

1. Lengkapi bagian Info Cluster.

   a. Pada Nama, masukkan nama untuk klaster Anda.

      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.

   b. Pada kotak Deskripsi, masukkan deskripsi untuk klaster ini.

2. Lengkapi bagian grup Subnet:

   - Untuk grup Subnet, buat grup subnet baru atau pilih yang sudah ada dari daftar yang tersedia yang ingin Anda terapkan ke cluster ini. Jika Anda membuat yang baru:

     - Masukkan Nama
     - Masukkan Deskripsi
     - Jika Anda mengaktifkan Multi-AZ, grup subnet harus berisi setidaknya dua subnet yang berada dalam availability zone yang berbeda. Untuk informasi selengkapnya, lihat Subnet dan grup subnet.
     - Jika Anda membuat grup subnet baru dan tidak memiliki VPC yang ada, Anda akan diminta untuk membuat VPC. Untuk informasi lebih lanjut, lihat Apa itu Amazon VPC? di Panduan Pengguna Amazon VPC.

3. Lengkapi bagian Pengaturan Cluster:

   a. Untuk mengaktifkan kemampuan pencarian Vektor, Anda dapat mengaktifkan ini untuk menyimpan embeddings vektor dan melakukan pencarian vektor. Perhatikan bahwa ini akan memperbaiki nilai untuk kompatibilitas versi Redis, grup Parameter dan Pecahan. Untuk informasi selengkapnya, lihat Pencarian vektor.

   b. Untuk kompatibilitas versi Redis, terima default`6.2`.

   c. Untuk Port, terima port Redis default 6379 atau, jika Anda memiliki alasan untuk menggunakan port yang berbeda, masukkan nomor port..

   d. Untuk grup Parameter, jika Anda telah mengaktifkan pencarian vektor, gunakan`default.memorydb-redis7.search.preview`. Jika tidak, terima grup `default.memorydb-redis7` parameter.

      Grup parameter mengontrol parameter runtime klaster Anda. Untuk mengetahui informasi selengkapnya tentang grup parameter, lihat Parameter spesifik Redis.

   e. Untuk tipe Node, pilih nilai untuk tipe node (bersama dengan ukuran memori terkait) yang Anda inginkan.

      Jika Anda memilih tipe node dari keluarga r6gd, Anda akan secara otomatis mengaktifkan data-tiering, yang membagi penyimpanan data antara memori dan SSD. Untuk informasi selengkapnya, lihat Tingkatan data.

   f. Untuk Jumlah pecahan, pilih jumlah pecahan yang Anda inginkan untuk cluster ini. Untuk ketersediaan cluster Anda yang lebih tinggi, kami sarankan Anda menambahkan setidaknya 2 pecahan.

      Anda dapat mengubah jumlah pecahan di cluster Anda secara dinamis. Untuk informasi selengkapnya, lihat Menskalakan klaster MemoryDB.

   g. Untuk Replika per serpihan, pilih jumlah simpul replika baca yang Anda inginkan dalam setiap serpihan.

      Pembatasan berikut ada:

      - Jika Multi-AZ diaktifkan, pastikan bahwa Anda memiliki setidaknya satu replika per serpihan.
      - Replika berjumlah sama untuk setiap serpihan saat membuat klaster menggunakan konsol.

h. Pilih Selanjutnya

i. Lengkapi bagian Pengaturan lanjutan:

i. 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.

ii. 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 selain default dengan memilih kunci KMS AWS milik Pelanggan yang Dikelola dan memilih kunci.

- Enkripsi Data Bergerak – Mengaktifkan enkripsi pada data selama pengiriman. Jika Anda memilih tidak ada enkripsi, maka daftar kontrol Akses terbuka yang disebut “akses terbuka” akan dibuat dengan pengguna default. Untuk informasi selengkapnya, lihat Mengautentikasi pengguna dengan Daftar Kontrol Akses (ACL).

iii. Untuk Snapshot, secara opsional tentukan periode retensi snapshot dan jendela snapshot. Secara default, Aktifkan snapshot otomatis telah dipilih sebelumnya.

iv. Untuk jendela Pemeliharaan opsional menentukan jendela pemeliharaan. Jendela pemeliharaan adalah waktu, umumnya satu jam panjangnya, setiap minggu ketika MemoryDB menjadwalkan pemeliharaan sistem untuk cluster Anda. Anda dapat mengizinkan MemoryDB 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.

v. Untuk Pemberitahuan, pilih topik Amazon Simple Notification Service (Amazon SNS) yang ada, atau pilih Input ARN Manual dan masukkan Nama Sumber Daya Amazon (ARN) topik. Amazon SNS memungkinkan Anda untuk mendorong pemberitahuan ke perangkat pintar yang terhubung ke Internet. Defaultnya adalah menonaktifkan notifikasi. Untuk informasi lebih lanjut, lihat https://aws.amazon.com/sns/.

vi. Untuk Tag, Anda dapat menerapkan tag secara opsional untuk mencari dan memfilter cluster Anda atau melacak biaya Anda AWS .

j. Tinjau semua entri dan pilihan Anda, lalu lakukan koreksi yang diperlukan. Saat Anda siap, pilih Buat untuk meluncurkan klaster Anda, atau Batal untuk membatalkan operasi.

Setelah status klaster Anda menjadi tersedia, Anda dapat memberikan akses ke EC2, menyambungkannya, dan mulai menggunakannya. Lihat informasi yang lebih lengkap di Langkah 2: Otorisasi akses ke cluster

> **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 Langkah 4: Menghapus cluster.

Restore from snapshots

Di bawah Sumber snapshot, pilih snapshot sumber untuk memigrasikan data. Untuk informasi selengkapnya, lihat Snapshot dan pulihkan .

> **Note**
>
> Jika Anda ingin cluster baru Anda mengaktifkan pencarian vektor, snapshot sumber juga harus mengaktifkan pencarian vektor.

Cluster target default ke pengaturan cluster sumber. Secara opsional, Anda dapat mengubah pengaturan berikut pada cluster target:

1. Info klaster

   a. Pada Nama, masukkan nama untuk klaster Anda.

      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.

   b. Pada kotak Deskripsi, masukkan deskripsi untuk klaster ini.

2. Grup subnet

   - Untuk grup Subnet, buat grup subnet baru atau pilih yang sudah ada dari daftar yang tersedia yang ingin Anda terapkan ke cluster ini. Jika Anda membuat yang baru:

     - Masukkan Nama
     - Masukkan Deskripsi
     - Jika Anda mengaktifkan Multi-AZ, grup subnet harus berisi setidaknya dua subnet yang berada dalam availability zone yang berbeda. Untuk informasi selengkapnya, lihat Subnet dan grup subnet.
     - Jika Anda membuat grup subnet baru dan tidak memiliki VPC yang ada, Anda akan diminta untuk membuat VPC. Untuk informasi lebih lanjut, lihat Apa itu Amazon VPC? di Panduan Pengguna Amazon VPC.

3. Pengaturan cluster

   a. Untuk mengaktifkan kemampuan pencarian Vektor, Anda dapat mengaktifkan ini untuk menyimpan embeddings vektor dan melakukan pencarian vektor. Perhatikan bahwa ini akan memperbaiki nilai untuk kompatibilitas versi Redis, grup Parameter dan Pecahan. Untuk informasi selengkapnya, lihat Pencarian vektor.

   b. Untuk kompatibilitas versi Redis, terima default`6.2`.

   c. Untuk Port, terima port Redis default 6379 atau, jika Anda memiliki alasan untuk menggunakan port yang berbeda, masukkan nomor port..

   d. Untuk grup Parameter, jika Anda telah mengaktifkan pencarian vektor, gunakan`default.memorydb-redis7.search.preview`. Jika tidak, terima grup `default.memorydb-redis7` parameter.

Grup parameter mengontrol parameter runtime klaster Anda. Untuk mengetahui informasi selengkapnya tentang grup parameter, lihat Parameter spesifik Redis.

e. Untuk tipe Node, pilih nilai untuk tipe node (bersama dengan ukuran memori terkait) yang Anda inginkan.

   Jika Anda memilih tipe node dari keluarga r6gd, Anda akan secara otomatis mengaktifkan data-tiering, yang membagi penyimpanan data antara memori dan SSD. Untuk informasi selengkapnya, lihat Tingkatan data.

f. Untuk Jumlah pecahan, pilih jumlah pecahan yang Anda inginkan untuk cluster ini. Untuk ketersediaan cluster Anda yang lebih tinggi, kami sarankan Anda menambahkan setidaknya 2 pecahan.

   Anda dapat mengubah jumlah pecahan di cluster Anda secara dinamis. Untuk informasi selengkapnya, lihat Menskalakan klaster MemoryDB.

g. Untuk Replika per serpihan, pilih jumlah simpul replika baca yang Anda inginkan dalam setiap serpihan.

   Pembatasan berikut ada:

   - Jika Multi-AZ diaktifkan, pastikan bahwa Anda memiliki setidaknya satu replika per serpihan.
   - Replika berjumlah sama untuk setiap serpihan saat membuat klaster menggunakan konsol.

h. Pilih Selanjutnya

i. Pengaturan lanjutan

   i. 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.

   ii. 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 selain default dengan memilih kunci KMS AWS milik Pelanggan yang Dikelola dan memilih kunci.

- Enkripsi Data Bergerak – Mengaktifkan enkripsi pada data selama pengiriman. Jika Anda memilih tidak ada enkripsi, maka daftar kontrol Akses terbuka yang disebut “akses terbuka” akan dibuat dengan pengguna default. Untuk informasi selengkapnya, lihat Mengautentikasi pengguna dengan Daftar Kontrol Akses (ACL).

iii. Untuk Snapshot, secara opsional tentukan periode retensi snapshot dan jendela snapshot. Secara default, Aktifkan snapshot otomatis telah dipilih sebelumnya.

iv. Untuk jendela Pemeliharaan opsional menentukan jendela pemeliharaan. Jendela pemeliharaan adalah waktu, umumnya satu jam panjangnya, setiap minggu ketika MemoryDB menjadwalkan pemeliharaan sistem untuk cluster Anda. Anda dapat mengizinkan MemoryDB 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.

v. Untuk Pemberitahuan, pilih topik Amazon Simple Notification Service (Amazon SNS) yang ada, atau pilih Input ARN Manual dan masukkan Nama Sumber Daya Amazon (ARN) topik. Amazon SNS memungkinkan Anda untuk mendorong pemberitahuan ke perangkat pintar yang terhubung ke Internet. Defaultnya adalah menonaktifkan notifikasi. Untuk informasi lebih lanjut, lihat https://aws.amazon.com/sns/.

vi. Untuk Tag, Anda dapat menerapkan tag secara opsional untuk mencari dan memfilter cluster Anda atau melacak biaya Anda AWS .

j. Tinjau semua entri dan pilihan Anda, lalu lakukan koreksi yang diperlukan. Saat Anda siap, pilih Buat untuk meluncurkan klaster Anda, atau Batal untuk membatalkan operasi.

Setelah status klaster Anda menjadi tersedia, Anda dapat memberikan akses ke EC2, menyambungkannya, dan mulai menggunakannya. Lihat informasi yang lebih lengkap di Langkah 2: Otorisasi akses ke cluster

> **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 Langkah 4: Menghapus cluster.

Membuat klaster AWS (CLI)

Untuk membuat cluster menggunakan AWS CLI, lihat `create-cluster`. Berikut ini adalah contohnya:

Untuk Linux, macOS, atau Unix:

```
aws memorydb create-cluster \
    --cluster-name my-cluster \
    --node-type db.r6g.large \
    --acl-name my-acl \
    --subnet-group my-sg
```

Untuk Windows:

```
aws memorydb create-cluster ^
   --cluster-name my-cluster ^
   --node-type db.r6g.large ^
   --acl-name my-acl ^
   --subnet-group my-sg
```

Anda harus mendapatkan respons JSON berikut:

```
{
    "Cluster": {
        "Name": "my-cluster",
        "Status": "creating",
        "NumberOfShards": 1,
        "AvailabilityMode": "MultiAZ",
        "ClusterEndpoint": {
            "Port": 6379
        },
        "NodeType": "db.r6g.large",
        "EngineVersion": "6.2",
        "EnginePatchVersion": "6.2.6",
        "ParameterGroupName": "default.memorydb-redis6",
        "ParameterGroupStatus": "in-sync",
        "SubnetGroupName": "my-sg",
        "TLSEnabled": true,
        "ARN": "arn:aws:memorydb:us-east-1:xxxxxxxxxxxxxx:cluster/my-cluster",
        "SnapshotRetentionLimit": 0,
        "MaintenanceWindow": "wed:03:00-wed:04:00",
```

```
        "SnapshotWindow": "04:30-05:30",
        "ACLName": "my-acl",
        "DataTiering": "false",
        "AutoMinorVersionUpgrade": true
    }
}
```

Anda dapat mulai menggunakan cluster setelah statusnya berubah menjadi`available`.

> **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 Langkah 4: Menghapus cluster.

Membuat cluster (MemoryDB API)

Untuk membuat cluster menggunakan API MemoryDB, gunakan tindakan. CreateCluster

> **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 Langkah 4: Menghapus cluster.

Menyiapkan otentikasi

Untuk informasi tentang menyiapkan otentikasi untuk klaster Anda, lihat Mengtentikasi IAM danMengautentikasi pengguna dengan Daftar Kontrol Akses (ACL).

Langkah 2: Otorisasi akses ke cluster

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

Cluster MemoryDB dirancang untuk diakses dari instans Amazon EC2. Mereka juga dapat diakses oleh aplikasi kontainer atau tanpa server yang berjalan di Amazon Elastic Container Service atau. AWS Lambda Skenario yang paling umum adalah mengakses cluster MemoryDB dari instans Amazon EC2 di Amazon Virtual Private Cloud (Amazon VPC) yang sama, yang akan menjadi kasus untuk latihan ini.

Sebelum Anda dapat menyambung ke klaster dari instans EC2, Anda harus memberikan otorisasi pada instans EC2 untuk mengakses klaster tersebut.

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 cluster. 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 cluster MemoryDB 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 khusus 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.

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

1. Masuk ke Konsol AWS Manajemen dan buka konsol VPC Amazon di https://console.aws.amazon.com/vpc.

2. Di panel navigasi kiri, pilih Grup Keamanan.
3. Pilih atau buat grup keamanan yang akan Anda gunakan untuk cluster 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 cluster MemoryDB 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 terhubung ke cluster, seperti yang dibahas di bagian berikutnya.

Untuk informasi tentang mengakses klaster MemoryDB Anda dari VPC Amazon yang berbeda, AWS Wilayah yang berbeda, atau bahkan jaringan perusahaan Anda, lihat berikut ini:

- Pola Akses untuk Mengakses Cluster MemoryDB di Amazon VPC
- Mengakses sumber daya MemoryDB dari luarAWS

Langkah 3: Connect ke cluster

Sebelum melanjutkan, selesaikan Langkah 2: Otorisasi akses ke cluster.

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 klaster hanya jika Anda telah mengotorisasi untuk melakukannya.

Temukan titik akhir klaster 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.

Untuk mengeksplorasi lebih lanjut cara menemukan titik akhir Anda, lihat yang berikut ini:

- Menemukan Endpoint untuk Cluster MemoryDB (AWS Management Console)
- Menemukan Endpoint untuk Cluster MemoryDB (AWSCLI)
- Menemukan Endpoint untuk Cluster MemoryDB (MemoryDB API)

Connect ke cluster MemoryDB (Linux)

Sekarang Anda memiliki titik akhir yang Anda butuhkan, Anda dapat masuk ke instans EC2 dan terhubung ke cluster. Dalam contoh berikut, Anda menggunakan utilitas cli untuk terhubung ke cluster menggunakan Ubuntu 22. Versi terbaru cli juga mendukung SSL/TLS untuk menghubungkan kluster yang diaktifkan enkripsi/otentikasi.

Menghubungkan ke node MemoryDB menggunakan redis-cli

Untuk mengakses data dari node MemoryDB, 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.

Untuk menggunakan redis-cli untuk terhubung ke cluster MemoryDB di Amazon Linux 2 atau Amazon Linux

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
> ```

Langkah 4: Menghapus cluster

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 cluster MemoryDB, snapshot manual Anda dipertahankan. Anda juga dapat membuat snapshot akhir sebelum klaster dihapus. Snapshot otomatis tidak dipertahankan. Untuk informasi selengkapnya, lihat Snapshot dan pulihkan .

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 MemoryDB for Redis di https://console.aws.amazon.com/memorydb/.
2. Untuk memilih cluster yang akan dihapus, pilih tombol radio di sebelah nama cluster dari daftar cluster. Dalam kasus ini, nama klaster yang Anda buat di Langkah 1: Buat cluster.
3. Untuk Tindakan, pilih Hapus.
4. Pertama pilih apakah akan membuat snapshot cluster sebelum menghapusnya dan kemudian masukkan `delete` di kotak konfirmasi dan Hapus untuk menghapus cluster, atau pilih Batal untuk menjaga cluster.

   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 `my-cluster`. Dalam kasus ini, ganti `my-cluster` dengan nama klaster yang Anda buat di Langkah 1: Buat cluster.

```
aws memorydb delete-cluster --cluster-name my-cluster
```

Operasi `delete-cluster` CLI hanya menghapus satu cluster. Untuk menghapus beberapa cluster, panggil `delete-cluster` setiap cluster yang ingin Anda hapus. Anda tidak perlu menunggu satu cluster selesai menghapus sebelum menghapus yang lain.

Untuk Linux, macOS, atau Unix:

```
aws memorydb delete-cluster \
    --cluster-name my-cluster \
    --region us-east-1
```

Untuk Windows:

```
aws memorydb delete-cluster ^
    --cluster-name my-cluster ^
    --region us-east-1
```

Untuk informasi selengkapnya, lihat `delete-cluster`.

Menggunakan MemoryDB API

Kode berikut menghapus klaster `my-cluster`. Dalam kasus ini, ganti `my-cluster` dengan nama klaster yang Anda buat di Langkah 1: Buat cluster.

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=DeleteCluster
    &ClusterName=my-cluster
    &Region=us-east-1
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210802T220302Z
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Date=20210802T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20210802T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

Operasi `DeleteCluster` API hanya menghapus satu cluster. Untuk menghapus beberapa cluster, panggil `DeleteCluster` setiap cluster yang ingin Anda hapus. Anda tidak perlu menunggu satu cluster selesai menghapus sebelum menghapus yang lain.

Untuk informasi lebih lanjut, lihat DeleteCluster.

Apa yang saya lakukan selanjutnya?

Sekarang setelah Anda mencoba latihan Memulai, Anda dapat menjelajahi bagian berikut untuk mempelajari lebih lanjut tentang MemoryDB dan alat yang tersedia:

- Memulai dengan AWS
- Alat untuk Amazon Web Services

- AWS Antarmuka Baris Perintah
- MemoryDB untuk Referensi API Redis.

Mengelola simpul

Node adalah blok bangunan terkecil dari MemoryDB untuk penyebaran Redis. Sebuah simpul milik pecahan yang termasuk dalam sebuah cluster. Setiap node menjalankan versi mesin yang dipilih saat cluster dibuat atau terakhir dimodifikasi. Setiap simpul mempunyai nama dan port Layanan Nama Domain (DNS) sendiri. Beberapa jenis node MemoryDB didukung, masing-masing dengan berbagai jumlah memori terkait dan daya komputasi.

Topik

Beberapa operasi penting yang melibatkan simpul adalah sebagai berikut:

- Menambahkan/Menghapus node dari cluster
- Penskalaan
- Menemukan titik akhir sambungan

Node dan pecahan MemoryDB

Shard adalah susunan hierarkis node, masing-masing dibungkus dalam sebuah cluster. 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. MemoryDB mendukung beberapa pecahan dalam sebuah cluster. Dukungan ini memungkinkan partisi data Anda dalam cluster MemoryDB.

MemoryDB mendukung replikasi melalui pecahan. Operasi API DescribeClustersmencantumkan pecahan dengan node anggota, nama node, titik akhir dan juga informasi lainnya.

Setelah cluster MemoryDB dibuat, itu dapat diubah (diskalakan masuk atau keluar). Untuk informasi lain, lihat Penskalaan 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. Melakukan hal ini dapat membantu jika Anda perlu

mengubah jenis node, versi engine, atau bermigrasi dari Amazon ElastiCache untuk Redis. Lihat informasi yang lebih lengkap di Membuat snapshot manual dan Memulihkan dari snapshot.

Tipe simpul yang didukung

MemoryDB mendukung jenis node berikut.

Memori yang dioptimalkan

Jenis instans	Bandwidth acuan (Gbps)	Bandwidth burst (Gbps)	Peningkatan I/O Multiplexing (Redis 7.0.4+)	Versi mesin minimum
db.r7g.large	0,937	12,5	Tidak	6.2
db.r7g.xlarge	1.876	12,5	Tidak	6.2
db.r7g.2xlarge	3,75	15	Ya	6.2
db.r7g.4xlarge	7,5	15	Ya	6.2
db.r7g.8xlarge	15	N/A	Ya	6.2
db.r7g.12xlarge	22.5	N/A	Ya	6.2
db.r7g.16xlarge	30	N/A	Ya	6.2
db.r6g.large	0,75	10.0	Tidak	6.2
db.r6g.xlarge	1,25	10.0	Tidak	6.2
db.r6g.2xlarge	2.5	10.0	Ya	6.2
db.r6g.4xlarge	5.0	10.0	Ya	6.2
db.r6g.8xlarge	12	N/A	Ya	6.2
db.r6g.12xlarge	20	N/A	Ya	6.2
db.r6g.16xlarge	25	N/A	Ya	6.2

Memori yang dioptimalkan dengan tingkatan data

Jenis instans	Bandwidth acuan (Gbps)	Bandwidth burst (Gbps)	Peningkatan I/O Multiplexing (Redis 7.0.4+)	Versi mesin minimum
db.r6gd.xlarge	1,25	10	Tidak	6.2
db.r6gd.2xlarge	2.5	10	Tidak	6.2
db.r6gd.4xlarge	5.0	10	Tidak	6.2
db.r6gd.8xlarge	12	N/A	Tidak	6.2

Node tujuan umum

Jenis instans	Bandwidth acuan (Gbps)	Bandwidth burst (Gbps)	Peningkatan I/O Multiplexing (Redis 7.0.4+)	Versi mesin minimum
db.t4g.small	0,18	5.0	Tidak	6.2
db.t4g.medium	0,256	5.0	Tidak	6.2

Untuk ketersediaan AWS Wilayah, lihat MemoryDB untuk Harga Redis

Semua jenis node dibuat dalam virtual private cloud (VPC).

Node cadangan MemoryDB

Node cadangan memberi Anda diskon yang signifikan dibandingkan dengan harga node sesuai permintaan. Node yang dicadangkan bukanlah node fisik, melainkan diskon penagihan yang diterapkan pada penggunaan node sesuai permintaan di akun Anda. Diskon untuk node yang dipesan terkait dengan tipe node dan AWS Wilayah.

Proses umum untuk bekerja dengan node yang dicadangkan adalah sebagai berikut:

- Tinjau informasi tentang penawaran node cadangan yang tersedia
- Beli penawaran node cadangan menggunakan AWS Management Console, AWS Command Line Interface atau SDK
- Tinjau informasi tentang node cadangan Anda yang ada

Topik

Ikhtisar node yang dicadangkan

Ketika Anda membeli node cadangan MemoryDB, Anda membeli komitmen untuk mendapatkan tarif diskon, pada jenis node tertentu, selama durasi node yang dicadangkan. Untuk menggunakan node cadangan MemoryDB, Anda membuat node baru seperti yang Anda lakukan untuk node sesuai permintaan. Node baru yang Anda buat harus sesuai dengan spesifikasi node yang dicadangkan. Jika spesifikasi node baru cocok dengan node cadangan yang ada untuk akun Anda, Anda akan ditagih dengan tarif diskon yang ditawarkan untuk node yang dicadangkan. Jika tidak, node ditagih pada tingkat permintaan. Anda dapat menggunakan API AWS Management Console, AWS CLI, atau MemoryDB untuk membuat daftar dan membeli penawaran node cadangan yang tersedia.

MemoryDB menawarkan node cadangan untuk memori yang dioptimalkan R7g, R6g, dan R6gd (dengan data tiering) node. Untuk informasi harga, lihat MemoryDB untuk Harga Redis.

Jenis penawaran

Node cadangan tersedia dalam tiga varietas — No Upfront, Partial Upfront, dan All Upfront — yang memungkinkan Anda mengoptimalkan MemoryDB Anda untuk biaya Redis berdasarkan penggunaan yang Anda harapkan.

No Upfront — Opsi ini menyediakan akses ke node yang dipesan tanpa memerlukan pembayaran di muka. Node cadangan No Upfront Anda menagih tarif per jam diskon untuk setiap jam dalam jangka waktu tersebut, terlepas dari penggunaannya, dan tidak diperlukan pembayaran di muka.

Partial Upfront — Opsi ini membutuhkan bagian dari node yang dicadangkan untuk dibayar di muka. Sisa jam dalam jangka waktu pemesanan akan ditagih dengan tarif per jam yang didiskon, terlepas dari penggunaannya.

Semua di muka — Pembayaran penuh dilakukan pada awal jangka waktu, tanpa biaya lain yang dikeluarkan untuk sisa jangka waktu terlepas dari jumlah jam yang digunakan.

Ketiga jenis penawaran tersedia dalam jangka waktu satu tahun dan tiga tahun.

Ukuran node cadangan fleksibel

Ketika Anda membeli node reserved, satu hal yang Anda tentukan adalah tipe node, misalnya db.r6g.xlarge. Untuk informasi selengkapnya, tentang jenis node, lihat MemoryDB for Redis Pricing.

Jika Anda memiliki node, dan Anda perlu menskalakannya ke kapasitas yang lebih besar, node cadangan Anda secara otomatis diterapkan ke node skala Anda. Artinya, node cadangan Anda secara otomatis diterapkan untuk penggunaan ukuran apa pun dalam keluarga simpul yang sama. Node cadangan fleksibel ukuran tersedia untuk node dengan Wilayah yang sama. AWS Node cadangan fleksibel ukuran hanya dapat menskalakan dalam keluarga simpulnya. Misalnya, node cadangan untuk db.r6g.xlarge dapat diterapkan ke db.r6g.2xlarge, tetapi tidak ke db.r6gd.large, karena db.r6g dan db.r6gd adalah keluarga simpul yang berbeda.

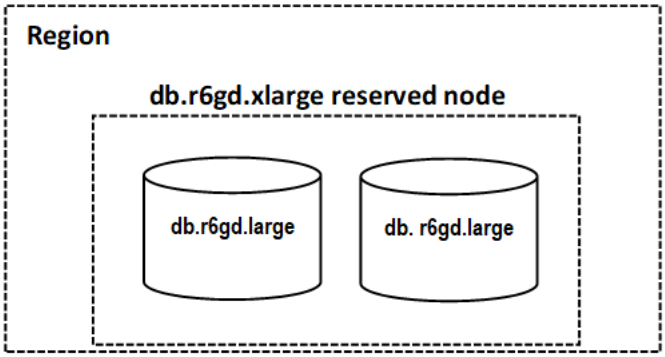
Fleksibilitas ukuran berarti Anda dapat bergerak bebas di antara konfigurasi dalam keluarga node yang sama. Misalnya, Anda dapat berpindah dari node cadangan r6g.xlarge (8 unit dinormalisasi) ke dua r6g.large node cadangan (8 unit dinormalisasi) (2*4 = 8 unit dinormalisasi) di Wilayah yang sama tanpa biaya tambahan. AWS

Anda dapat membandingkan penggunaan untuk ukuran node cadangan yang berbeda dengan menggunakan unit yang dinormalisasi. Misalnya, satu unit penggunaan pada dua node db.r6g.4xlarge setara dengan 16 unit penggunaan yang dinormalisasi pada satu db.r6g.large. Tabel berikut menunjukkan jumlah unit dinormalisasi untuk setiap ukuran node:

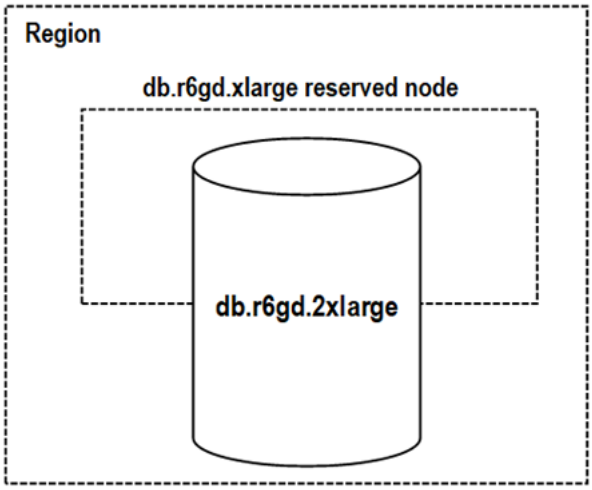
Ukuran simpul	Unit yang dinormalisasi
small	1

Ukuran simpul	Unit yang dinormalisasi
medium	2
large	4
xlarge	8
2xlarge	16
4xlarge	32
6xlarge	48
8xlarge	64
10xlarge	80
12xlarge	96
16xlarge	128

Misalnya, Anda membeli node cadangan db.r6gd.xlarge, dan Anda memiliki dua node cadangan db.r6gd.large yang sedang berjalan di akun Anda di Wilayah yang sama. AWS Dalam hal ini, manfaat penagihan diterapkan secara penuh ke kedua node.



Atau, jika Anda memiliki satu instans db.r6gd.2xlarge yang berjalan di akun Anda di AWS Wilayah yang sama, manfaat penagihan diterapkan ke 50 persen dari penggunaan node yang dicadangkan.



Menghapus node yang dicadangkan

Persyaratan untuk node cadangan melibatkan komitmen satu tahun atau tiga tahun. Anda tidak dapat membatalkan node yang dicadangkan. Namun, Anda dapat menghapus node yang dicakup oleh discount node reserved. Proses untuk menghapus node yang dicakup oleh discount node reserved sama dengan node lainnya.

Jika Anda menghapus node yang dicakup oleh discount node reserved, Anda dapat meluncurkan node lain dengan spesifikasi yang kompatibel. Dalam hal ini, Anda tetap mendapatkan tarif diskon selama jangka waktu pemesanan (satu atau tiga tahun).

Bekerja dengan node yang dicadangkan

Anda dapat menggunakan API AWS Management Console, the AWS Command Line Interface, dan MemoryDB untuk bekerja dengan node yang dicadangkan.

Konsol

Untuk mendapatkan harga dan informasi tentang penawaran node cadangan yang tersedia

1. Masuk ke AWS Management Console dan buka konsol MemoryDB for Redis di https://console.aws.amazon.com/memorydb/.
2. Di panel navigasi, pilih Node cadangan.
3. Pilih Beli node yang dicadangkan.
4. Untuk tipe Node, pilih jenis node yang ingin Anda gunakan.
5. Untuk Kuantitas, pilih jumlah node yang ingin Anda gunakan.

6. Untuk Term, pilih lamanya waktu yang Anda inginkan node database dicadangkan.
7. Untuk Jenis penawaran, pilih jenis penawaran.

Setelah Anda membuat pilihan ini, Anda dapat melihat informasi harga di bawah Ringkasan reservasi.

> **Important**
>
> Pilih Batal untuk menghindari pembelian node yang dipesan ini dan menimbulkan biaya apa pun.

Setelah Anda memiliki informasi tentang penawaran node cadangan yang tersedia, Anda dapat menggunakan informasi tersebut untuk membeli penawaran seperti yang ditunjukkan dalam prosedur berikut:

Untuk membeli node yang dicadangkan

1. Masuk ke AWS Management Console dan buka konsol MemoryDB for Redis di https://console.aws.amazon.com/memorydb/.
2. Di panel navigasi, pilih Node cadangan.
3. Pilih Beli node yang dicadangkan.
4. Untuk tipe Node, pilih jenis node yang ingin Anda gunakan.
5. Untuk Kuantitas, pilih jumlah node yang ingin Anda gunakan.
6. Untuk Term, pilih lamanya waktu yang Anda inginkan node database dicadangkan.
7. Untuk Jenis penawaran, pilih jenis penawaran.
8. (Opsional) Anda dapat menetapkan pengenal Anda sendiri ke node cadangan yang Anda beli untuk membantu Anda melacaknya. Untuk ID Reservasi, ketik pengenal untuk node reservasi Anda.

   Setelah Anda membuat pilihan ini, Anda dapat melihat informasi harga di bawah Ringkasan reservasi.
9. Pilih Beli node yang dicadangkan.
10. Node cadangan Anda dibeli, lalu ditampilkan dalam daftar node Cadangan.

Untuk mendapatkan informasi tentang node yang dicadangkan untuk AWS akun Anda

1. Masuk ke AWS Management Console dan buka konsol MemoryDB for Redis di https://console.aws.amazon.com/memorydb/.
2. Di panel navigasi, pilih Node cadangan.
3. Node yang dicadangkan untuk akun Anda muncul. Untuk melihat informasi rinci tentang node reserved tertentu, pilih node tersebut dalam daftar. Anda kemudian dapat melihat informasi rinci tentang node itu secara detail.

AWS Command Line Interface

`describe-reserved-nodes-offerings`Contoh berikut mengembalikan rincian penawaran reserved-node.

```
aws memorydb describe-reserved-nodes-offerings
```

Ini menghasilkan output yang mirip dengan yang berikut:

```
{
    "ReservedNodesOfferings": [
        {
            "ReservedNodesOfferingId": "0193cc9d-7037-4d49-b332-xxxxxxxxxxxx",
            "NodeType": "db.xxx.large",
            "Duration": 94608000,
            "FixedPrice": $xxx.xx,
            "OfferingType": "Partial Upfront",
            "RecurringCharges": [
                {
                    "RecurringChargeAmount": $xx.xx,
                    "RecurringChargeFrequency": "Hourly"
                }
            ]
        }
    ]
}
```

Anda juga dapat meneruskan parameter berikut untuk membatasi ruang lingkup dari apa yang dikembalikan:

- `--reserved-nodes-offering-id` – ID penawaran yang ingin Anda beli.
- `--node-type`— Nilai filter tipe node. Gunakan parameter ini untuk menampilkan hanya reservasi yang cocok dengan tipe node yang ditentukan.
- `--duration`— Nilai filter durasi, ditentukan dalam tahun atau detik. Gunakan parameter ini untuk hanya menampilkan reservasi selama durasi ini.
- `--offering-type`— Gunakan parameter ini untuk hanya menampilkan penawaran yang tersedia yang cocok dengan jenis penawaran yang ditentukan.

Setelah Anda memiliki informasi tentang penawaran node cadangan yang tersedia, Anda dapat menggunakan informasi tersebut untuk membeli penawaran.

`purchase-reserved-nodes-offering`Contoh berikut membeli node cadangan baru

Untuk Linux, macOS, atau Unix:

```
aws memorydb purchase-reserved-nodes-offering \

    --reserved-nodes-offering-id 0193cc9d-7037-4d49-b332-d5e984f1d8ca \
    --reservation-id reservation \
    --node-count 2
```

Untuk Windows:

```
aws memorydb purchase-reserved-nodes-offering ^
    --reserved-nodes-offering-id 0193cc9d-7037-4d49-b332-d5e984f1d8ca ^
    --reservation-id MyReservation
```

- `--reserved-nodes-offering-id`mewakili nama node cadangan yang menawarkan untuk membeli.
- `--reservation-id`adalah pengenal yang ditentukan pelanggan untuk melacak reservasi ini.

> **Note**
>
> ID Reservasi adalah pengenal unik yang ditentukan pelanggan untuk melacak reservasi ini. Jika parameter ini tidak ditentukan, MemoryDB secara otomatis menghasilkan pengenal untuk reservasi.

- `--node-count`adalah jumlah node yang akan dicadangkan. Defaultnya ke 1.

Ini menghasilkan output yang mirip dengan yang berikut:

```
{
    "ReservedNode": {
        "ReservationId": "reservation",
        "ReservedNodesOfferingId": "0193cc9d-7037-4d49-b332-xxxxxxxxxxxx",
        "NodeType": "db.xxx.large",
        "StartTime": 1671173133.982,
        "Duration": 94608000,
        "FixedPrice": $xxx.xx,
        "NodeCount": 2,
        "OfferingType": "Partial Upfront",
        "State": "payment-pending",
        "RecurringCharges": [
            {
                "RecurringChargeAmount": $xx.xx,
                "RecurringChargeFrequency": "Hourly"
            }
        ],
        "ARN": "arn:aws:memorydb:us-east-1:xxxxxxxx:reservednode/reservation"
    }
}
```

Setelah Anda membeli node yang dicadangkan, Anda bisa mendapatkan informasi tentang node cadangan Anda.

`describe-reserved-nodes`Contoh berikut mengembalikan informasi tentang node reserved untuk akun ini.

```
aws memorydb describe-reserved-nodes
```

Ini menghasilkan output yang mirip dengan yang berikut:

```
{
    "ReservedNodes": [
        {
            "ReservationId": "ri-2022-12-16-00-28-40-600",
            "ReservedNodesOfferingId": "0193cc9d-7037-4d49-b332-xxxxxxxxxxxx",
            "NodeType": "db.xxx.large",
            "StartTime": 1671150737.969,
```

```
            "Duration": 94608000,
            "FixedPrice": $xxx.xx,
            "NodeCount": 1,
            "OfferingType": "Partial Upfront",
            "State": "active",
            "RecurringCharges": [
                {
                    "RecurringChargeAmount": $xx.xx,
                    "RecurringChargeFrequency": "Hourly"
                }
            ],
            "ARN": "arn:aws:memorydb:us-east-1:xxxxxxxx:reservednode/
ri-2022-12-16-00-28-40-600"
        }
    ]
}
```

Anda juga dapat meneruskan parameter berikut untuk membatasi ruang lingkup dari apa yang dikembalikan:

- `--reservation-id`— Anda dapat menetapkan pengenal Anda sendiri ke node cadangan yang Anda beli untuk membantu melacaknya.
- `--reserved-nodes-offering-id`— Nilai filter pengenal penawaran. Gunakan parameter ini untuk menampilkan hanya reservasi yang dibeli yang cocok dengan pengenal penawaran yang ditentukan.
- `--node-type`— Nilai filter tipe node. Gunakan parameter ini untuk menampilkan hanya reservasi yang cocok dengan tipe node yang ditentukan.
- `--duration`— Nilai filter durasi, ditentukan dalam tahun atau detik. Gunakan parameter ini untuk hanya menampilkan reservasi selama durasi ini.
- `--offering-type`— Gunakan parameter ini untuk hanya menampilkan penawaran yang tersedia yang cocok dengan jenis penawaran yang ditentukan.

MemoryDB API

Contoh berikut menunjukkan cara menggunakan MemoryDB Query API untuk node yang dicadangkan:

DescribeReservedNodesOfferings

Mengembalikan rincian penawaran reserved-node.

```
https://memorydb.us-west-2.amazonaws.com/
    ?Action=DescribeReservedNodesOfferings
    &ReservedNodesOfferingId=649fd0c8-xxxx-xxxx-xxxx-06xxxx75e95f
 &"Duration": 94608000,
    &NodeType="db.r6g.large"
    &OfferingType="Partial Upfront"
    &Version=2021-01-01
    &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>
```

Parameter berikut membatasi ruang lingkup dari apa yang dikembalikan:

- `ReservedNodesOfferingId`mewakili nama node cadangan yang menawarkan untuk membeli.
- `Duration`— Nilai filter durasi, ditentukan dalam tahun atau detik. Gunakan parameter ini untuk hanya menampilkan reservasi selama durasi ini.
- `NodeType`— Nilai filter tipe node. Gunakan parameter ini untuk menampilkan hanya penawaran yang cocok dengan tipe node yang ditentukan.
- `OfferingType`— Gunakan parameter ini untuk hanya menampilkan penawaran yang tersedia yang cocok dengan jenis penawaran yang ditentukan.

Setelah Anda memiliki informasi tentang penawaran node cadangan yang tersedia, Anda dapat menggunakan informasi tersebut untuk membeli penawaran.

PurchaseReservedNodesOffering

Memungkinkan Anda membeli penawaran node yang dicadangkan.

```
https://memorydb.us-west-2.amazonaws.com/
    ?Action=PurchaseReservedCacheNodesOffering
    &ReservedNodesOfferingId=649fd0c8-xxxx-xxxx-xxxx-06xxxx75e95f
    &ReservationID=myreservationID
    &NodeCount=1
    &Version=2021-01-01
```

```
    &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>
```

- `ReservedNodesOfferingId`mewakili nama node cadangan yang menawarkan untuk membeli.
- `ReservationID`adalah pengenal yang ditentukan pelanggan untuk melacak reservasi ini.

> **Note**
>
> ID Reservasi adalah pengenal unik yang ditentukan pelanggan untuk melacak reservasi ini. Jika parameter ini tidak ditentukan, MemoryDB secara otomatis menghasilkan pengenal untuk reservasi.

- `NodeCount`adalah jumlah node yang akan dicadangkan. Defaultnya ke 1.

Setelah Anda membeli node yang dicadangkan, Anda bisa mendapatkan informasi tentang node cadangan Anda.

DescribeReservedNodes

Mengembalikan informasi tentang node yang dicadangkan untuk akun ini.

```
https://memorydb.us-west-2.amazonaws.com/
 ?Action=DescribeReservedNodes
 &ReservedNodesOfferingId=649fd0c8-xxxx-xxxx-xxxx-06xxxx75e95f
 &ReservationID=myreservationID
 &NodeType="db.r6g.large"
 &Duration=94608000
 &OfferingType="Partial Upfront"
 &Version=2021-01-01
 &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>
```

Parameter berikut membatasi ruang lingkup dari apa yang dikembalikan:

- `ReservedNodesOfferingId`mewakili nama node reserved.
- `ReservationID`— Anda dapat menetapkan pengenal Anda sendiri ke node cadangan yang Anda beli untuk membantu melacaknya.
- `NodeType`— Nilai filter tipe node. Gunakan parameter ini untuk menampilkan hanya reservasi yang cocok dengan tipe node yang ditentukan.
- `Duration`— Nilai filter durasi, ditentukan dalam tahun atau detik. Gunakan parameter ini untuk hanya menampilkan reservasi selama durasi ini.
- `OfferingType`— Gunakan parameter ini untuk hanya menampilkan penawaran yang tersedia yang cocok dengan jenis penawaran yang ditentukan.

Melihat tagihan untuk node cadangan Anda

Anda dapat melihat penagihan untuk node yang dipesan di Dasbor Penagihan di. AWS Management Console

Untuk melihat penagihan node yang dipesan

1. Masuk ke AWS Management Console dan buka konsol MemoryDB for Redis di https://console.aws.amazon.com/memorydb/.
2. Dari tombol Cari di bagian atas konsol, pilih Penagihan.
3. Pilih Tagihan dari sisi kiri dasbor.
4. Di bawah Biaya AWS Layanan, perluas MemoryDB.
5. Perluas AWS Wilayah tempat node cadangan Anda berada, misalnya US East (Virginia N.).

Node cadangan Anda dan biaya per jamnya untuk bulan berjalan ditampilkan di bawah Instans Cadangan Amazon MemoryDB CreateCluster .

Amazon MemoryDB CreateCluster Reserved Instances		
AmazonMemoryDB, db.r6g.large reserved instance applied	81.000 Hrs	
AmazonMemoryDB, db.r6g.4xlarge reserved instance applied	324.000 Hrs	
AmazonMemoryDB, db.r6g.4xlarge reserved instance applied	162.000 Hrs	
USD (hourly fee per AmazonMemoryDB, db.r6g.large instance	1,488.000 Hrs	
USD (hourly fee per AmazonMemoryDB, db.r6gd.2xlarge instance	744.000 Hrs	
USD 0 hourly fee per AmazonMemoryDB, db.r6g.4xlarge instance	744.000 Hrs	
USD (hourly fee per AmazonMemoryDB, db.r6gd.xlarge instance	744.000 Hrs	
USD 1 hourly fee per AmazonMemoryDB, db.r6gd.4xlarge instance	2,976.000 Hrs	

Mengganti simpul

MemoryDB sering meningkatkan armadanya dengan tambalan dan peningkatan, biasanya mulus. Namun, dari waktu ke waktu kami perlu meluncurkan kembali node MemoryDB Anda untuk menerapkan pembaruan OS wajib ke host yang mendasarinya. Penggantian ini diperlukan untuk menerapkan upgrade yang memperkuat keamanan, keandalan, dan performa operasional.

Anda memiliki opsi 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 akan terus menerima peringatan yang menunjukkan bahwa penggantian simpul harus dilakukan. Jika Anda telah mengurangi kebutuhan pemeliharaan secara manual, Anda dapat mengabaikan peringatan ini.

> **Note**
>
> Node pengganti yang secara otomatis dihasilkan oleh MemoryDB untuk Redis mungkin memiliki alamat IP yang berbeda. Anda bertanggung jawab untuk meninjau konfigurasi aplikasi Anda untuk memastikan bahwa node Anda terkait dengan alamat IP yang sesuai.

Daftar berikut mengidentifikasi tindakan yang dapat Anda lakukan saat MemoryDB menjadwalkan salah satu node Anda untuk diganti:

Opsi penggantian node MemoryDB

- Tidak melakukan apa-apa - Jika Anda tidak melakukan apa-apa, MemoryDB menggantikan node sesuai jadwal.

  Jika node adalah anggota cluster multi-AZ, MemoryDB menyediakan ketersediaan yang lebih baik selama penambalan, pembaruan, dan penggantian node terkait pemeliharaan lainnya.

  Penggantian selesai saat cluster melayani permintaan tulis masuk.
- Ubah jendela pemeliharaan Anda - Untuk acara pemeliharaan terjadwal, Anda menerima email atau acara pemberitahuan dari MemoryDB. Dalam hal ini, jika Anda mengubah periode pemeliharaan Anda sebelum waktu penggantian terjadwal, simpul Anda sekarang akan diganti pada waktu yang baru. Untuk informasi selengkapnya, lihat Memodifikasi cluster MemoryDB.

> **Note**
>
> Kemampuan untuk mengubah jendela pengganti Anda dengan memindahkan jendela pemeliharaan Anda hanya tersedia ketika pemberitahuan MemoryDB menyertakan jendela pemeliharaan. Jika notifikasi tersebut tidak menyertakan periode pemeliharaan, Anda tidak dapat mengubah periode penggantian.

Misalnya, anggaplah pemeliharaan dilakukan pada Kamis, 9 November, pukul 15.00 dan periode pemeliharaan berikutnya adalah Jumat, 10 November, pukul 17.00. Berikut adalah tiga skenario beserta hasilnya:

- Anda mengubah periode pemeliharaan Anda ke Jumat pukul 16.00, setelah tanggal dan waktu saat ini dan sebelum periode pemeliharaan terjadwal berikutnya. Simpul akan diganti 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 periode pemeliharaan terjadwal berikutnya. Simpul akan diganti pada hari Sabtu, 11 November, pukul 16.00.
- Anda mengubah jendela pemeliharaan Anda menjadi Rabu pukul 16:00, lebih awal dalam seminggu dari tanggal dan waktu saat ini. Simpul akan diganti pada Rabu depan, 15 November, pukul 16.00.

Untuk petunjuk, lihat Mengelola pemeliharaan.

Mengelola Klaster

Sebagian besar operasi MemoryDB 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 untuk jenis simpul yang sama dan 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 MemoryDB API dan perintah. AWS CLI Pengidentifikasi klaster harus unik untuk pelanggan itu di Wilayah AWS.

Cluster MemoryDB dirancang untuk diakses menggunakan instans Amazon EC2. Anda hanya dapat meluncurkan cluster MemoryDB Anda di cloud pribadi virtual (VPC) berdasarkan layanan Amazon VPC, tetapi Anda dapat mengaksesnya dari luar. AWS Untuk informasi selengkapnya, lihat Mengakses sumber daya MemoryDB dari luarAWS.

Tingkatan data

Cluster yang menggunakan tipe node dari keluarga r6gd memiliki datanya berjenjang antara memori dan penyimpanan SSD lokal (solid state drive). Data tiering menyediakan opsi harga-kinerja baru untuk beban kerja Redis dengan memanfaatkan solid state drive (SSD) berbiaya lebih rendah di setiap node cluster selain menyimpan data dalam memori. Mirip dengan tipe node lainnya, data yang ditulis ke node r6gd disimpan secara tahan lama dalam log transaksi Multi-AZ. Tiering data sangat ideal untuk beban kerja yang mengakses hingga 20 persen dari keseluruhan dataset mereka secara teratur, dan untuk aplikasi yang dapat mentolerir latensi tambahan saat mengakses data pada SSD.

Pada cluster dengan data tiering, MemoryDB memonitor waktu akses terakhir dari setiap item toko. Ketika memori yang tersedia (DRAM) sepenuhnya dikonsumsi, MemoryDB menggunakan algoritma yang paling jarang digunakan (LRU) untuk secara otomatis memindahkan item yang jarang diakses dari memori ke SSD. Ketika data pada SSD kemudian diakses, MemoryDB secara otomatis dan asinkron bergerak kembali ke memori sebelum memproses permintaan. Jika Anda memiliki beban kerja yang hanya mengakses sebagian datanya secara teratur, penataan data adalah cara optimal untuk menskalakan kapasitas Anda dengan hemat biaya.

Perhatikan bahwa ketika menggunakan data tiering, kunci sendiri selalu tetap dalam memori, sedangkan LRU mengatur penempatan nilai pada memori vs disk. Secara umum, kami menyarankan agar ukuran kunci Anda lebih kecil dari ukuran nilai Anda saat menggunakan tiering data.

Tingkatan data dirancang untuk memiliki dampak kinerja yang minimal pada beban kerja aplikasi. Misalnya, dengan asumsi nilai String 500-byte, Anda biasanya dapat mengharapkan tambahan 450 mikrodetik latensi untuk permintaan baca ke data yang disimpan di SSD dibandingkan dengan permintaan baca ke data dalam memori.

Dengan ukuran node tiering data terbesar (db.r6gd.8xlarge), Anda dapat menyimpan hingga ~ 500 TB dalam satu klaster 500 node (250 TB saat menggunakan 1 replika baca). Untuk data tiering, MemoryDB cadangan 19% dari (DRAM) memori per node untuk penggunaan non-data. Data tiering kompatibel dengan semua perintah Redis dan struktur data yang didukung di MemoryDB. Anda tidak perlu perubahan sisi klien untuk menggunakan fitur ini.

Topik

- Memantau
- Menggunakan data tiering
- Memulihkan data dari snapshot ke dalam klaster dengan tiering data diaktifkan

Praktik terbaik

Kami merekomendasikan praktik terbaik berikut:

- Tiering data sangat ideal untuk beban kerja yang mengakses hingga 20 persen dari keseluruhan dataset mereka secara teratur, dan untuk aplikasi yang dapat mentolerir latensi tambahan saat mengakses data pada SSD.
- Saat menggunakan kapasitas SSD yang tersedia pada node berjenjang data, sebaiknya ukuran nilai lebih besar dari ukuran kunci. Ukuran nilai tidak boleh lebih besar dari 128MB; lain itu tidak akan dipindahkan ke disk. Ketika item dipindahkan antara DRAM dan SSD, kunci akan selalu tetap dalam memori dan hanya nilai yang dipindahkan ke tingkat SSD.

Keterbatasan:

Tingkatan data memiliki batasan berikut:

- Jenis node yang Anda gunakan harus dari keluarga r6gd, yang tersedia di wilayah berikut:`us-east-2`,,,,`us-east-1`,`us-west-2`,`us-west-1`,`eu-west-1`,`eu-west-3`,`eu-central-1`,`ap-northeast-1`,`ap-southeast-1`,`ap-southeast-2`,`ap-south-1`,`ca-central-1` dan`sa-east-1`.
- Anda tidak dapat mengembalikan snapshot dari cluster r6gd ke klaster lain kecuali jika ia juga menggunakan r6gd.
- Anda tidak dapat mengekspor snapshot ke Amazon S3 untuk klaster tingkat data.
- Penyimpanan tanpa forkless tidak didukung.
- Scaling tidak didukung dari klaster data tiering (misalnya, cluster menggunakan tipe node r6gd) ke cluster yang tidak menggunakan data tiering (misalnya, cluster menggunakan tipe node r6g).
- Data tiering hanya mendukung`volatile-lru`,`allkeys-lru` dan kebijakan`noeviction` maxmemory.
- Item yang lebih besar dari 128 MiB tidak dipindahkan ke SSD.

Harga tingkatan data

Node R6gD memiliki kapasitas total 5x lebih banyak (memori+SSD) dan dapat membantu Anda mencapai penghematan biaya penyimpanan lebih dari 60 persen saat berjalan pada pemanfaatan maksimum dibandingkan dengan node R6g (hanya memori). Untuk informasi lebih lanjut, lihat Harga MemoryDB.

Memantau

MemoryDB menawarkan metrik yang dirancang khusus untuk memantau klaster kinerja yang menggunakan tiering data. Untuk memantau rasio item dalam DRAM dibandingkan dengan SSD, Anda dapat menggunakan`CurrItems` metrik diMetrik untuk MemoryDB.. Anda dapat menghitung persentase sebagai:`(CurrItems with Dimension: Tier = Memory * 100) / (CurrItems with no dimension filter)`. Ketika persentase item dalam memori menurun di bawah 5 persen, kami sarankan Anda mempertimbangkannyaMenskalakan klaster MemoryDB.

Untuk informasi selengkapnya, lihat Metrik untuk klaster MemoryDB yang menggunakan tingkat data diMetrik untuk MemoryDB.

Menggunakan data tiering

Menggunakan data tiering menggunakanAWS Management Console

Saat membuat cluster, Anda menggunakan data tiering dengan memilih jenis node dari keluarga r6gd, seperti db.r6gd.xlarge. Memilih jenis node secara otomatis memungkinkan tiering data.

Untuk informasi lebih lanjut tentang membuat cluster, lihatLangkah 1: Buat cluster.

Mengaktifkan tiering data menggunakanAWS CLI

Saat membuat cluster menggunakanAWS CLI, Anda menggunakan data tiering dengan memilih jenis node dari keluarga r6gd, seperti db.r6gd.xlarge dan pengaturan`--data-tiering` parameter.

Anda tidak dapat memilih keluar dari tingkat data saat memilih jenis node dari keluarga r6gd. Jika Anda mengatur`--no-data-tiering` parameter, operasi akan gagal.

Untuk Linux, macOS, atau Unix:

```
aws memorydb create-cluster \
```

```
   --cluster-name my-cluster \
   --node-type db.r6gd.xlarge \
   --acl-name my-acl \
   --subnet-group my-sg \
   --data-tiering
```

Untuk Windows:

```
aws memorydb create-cluster ^
   --cluster-name my-cluster ^
   --node-type db.r6gd.xlarge ^
   --acl-name my-acl ^
   --subnet-group my-sg
   --data-tiering
```

Setelah menjalankan operasi ini, Anda akan melihat respons yang mirip dengan berikut ini:

```
{
    "Cluster": {
        "Name": "my-cluster",
        "Status": "creating",
        "NumberOfShards": 1,
        "AvailabilityMode": "MultiAZ",
        "ClusterEndpoint": {
            "Port": 6379
        },
        "NodeType": "db.r6gd.xlarge",
        "EngineVersion": "6.2",
        "EnginePatchVersion": "6.2.6",
        "ParameterGroupName": "default.memorydb-redis6",
        "ParameterGroupStatus": "in-sync",
        "SubnetGroupName": "my-sg",
        "TLSEnabled": true,
        "ARN": "arn:aws:memorydb:us-east-1:xxxxxxxxxxxxx:cluster/my-cluster",
        "SnapshotRetentionLimit": 0,
        "MaintenanceWindow": "wed:03:00-wed:04:00",
        "SnapshotWindow": "04:30-05:30",
        "ACLName": "my-acl",
        "DataTiering":"true",
        "AutoMinorVersionUpgrade": true
    }
}
```

Memulihkan data dari snapshot ke dalam klaster dengan tiering data diaktifkan

Anda dapat mengembalikan snapshot ke klaster baru dengan data tiering diaktifkan menggunakan (Console), (AWSCLI) atau (MemoryDB API). Ketika Anda membuat klaster menggunakan jenis node dalam keluarga r6gd, tingkat data diaktifkan.

Memulihkan data dari snapshot ke dalam klaster dengan tingkat data diaktifkan (konsol)

Untuk mengembalikan snapshot ke klaster baru dengan data tiering diaktifkan (konsol), ikuti langkah-langkah diMemulihkan dari snapshot (Konsol)

Perhatikan bahwa untuk mengaktifkan data-tiering, Anda perlu memilih jenis node dari keluarga r6gd.

Memulihkan data dari snapshot ke dalam klaster dengan data tiering diaktifkan (AWSCLI)

Saat membuat cluster menggunakanAWS CLI, data tiering secara default digunakan dengan memilih tipe node dari keluarga r6gd, seperti db.r6gd.xlarge dan pengaturan`--data-tiering` parameter.

Anda tidak dapat memilih keluar dari tingkat data saat memilih jenis node dari keluarga r6gd. Jika Anda mengatur`--no-data-tiering` parameter, operasi akan gagal.

Untuk Linux, macOS, atau Unix:

```
aws memorydb create-cluster \
   --cluster-name my-cluster \
   --node-type db.r6gd.xlarge \
   --acl-name my-acl \
   --subnet-group my-sg \
   --data-tiering \
   --snapshot-name my-snapshot
```

Untuk Linux, macOS, atau Unix:

```
aws memorydb create-cluster ^
   --cluster-name my-cluster ^
   --node-type db.r6gd.xlarge ^
   --acl-name my-acl ^
```

```
    --subnet-group my-sg ^
    --data-tiering ^
    --snapshot-name my-snapshot
```

Setelah menjalankan operasi ini, Anda akan melihat respons yang mirip dengan berikut ini:

```
{
    "Cluster": {
        "Name": "my-cluster",
        "Status": "creating",
        "NumberOfShards": 1,
        "AvailabilityMode": "MultiAZ",
        "ClusterEndpoint": {
            "Port": 6379
        },
        "NodeType": "db.r6gd.xlarge",
        "EngineVersion": "6.2",
        "EnginePatchVersion": "6.2.6",
        "ParameterGroupName": "default.memorydb-redis6",
        "ParameterGroupStatus": "in-sync",
        "SubnetGroupName": "my-sg",
        "TLSEnabled": true,
        "ARN": "arn:aws:memorydb:us-east-1:xxxxxxxxxxxxxx:cluster/my-cluster",
        "SnapshotRetentionLimit": 0,
        "MaintenanceWindow": "wed:03:00-wed:04:00",
        "SnapshotWindow": "04:30-05:30",
        "ACLName": "my-acl",
        "DataTiering": "true"
}
```

Menyiapkan klaster

Berikut ini, Anda dapat menemukan petunjuk tentang membuat cluster menggunakan konsol MemoryDB,AWS CLI, atau MemoryDB API.

Setiap kali Anda membuat cluster, itu adalah ide yang baik untuk melakukan beberapa pekerjaan persiapan sehingga Anda tidak perlu meng-upgrade atau membuat perubahan segera.

Topik

- Menentukan kebutuhan Anda

Menentukan kebutuhan Anda

Persiapan

Mengetahui jawaban atas pertanyaan berikut membantu proses membuat klaster Anda berjalan lebih mulus:

- Pastikan untuk membuat grup subnet di VPC yang sama sebelum Anda mulai membuat cluster. Atau, Anda dapat menggunakan grup subnet default yang disediakan. Untuk informasi selengkapnya, lihat Subnet dan grup subnet.

  MemoryDB dirancang untuk diakses dari dalam AWS menggunakan Amazon EC2. Namun, jika Anda meluncurkan dalam VPC berbasis Amazon VPC, Anda dapat memberikan akses dari luar. AWS Untuk informasi selengkapnya, lihat Mengakses sumber daya MemoryDB dari luarAWS.
- Perlukah menyesuaikan nilai parameter tertentu?

  Jika perlu, buat grup parameter khusus. Untuk informasi selengkapnya, lihat Membuat grup parameter.
- Apakah Anda perlu membuat grup keamanan VPC?

  Untuk informasi selengkapnya, lihat Keamanan di VPC Anda.
- Bagaimana toleransi kesalahan akan diterapkan?

  Untuk informasi selengkapnya, lihat Mengurangi Kegagalan.

Topik

- Persyaratan memori dan prosesor
- Konfigurasi cluster MemoryDB
- Peningkatan I/O Multiplexing
- Persyaratan penskalaan
- Persyaratan akses
- Wilayah dan Zona Ketersediaan

Persyaratan memori dan prosesor

Blok bangunan dasar MemoryDB untuk Redis adalah node. Node dikonfigurasi dalam pecahan untuk membentuk cluster. Saat menentukan jenis simpul yang akan digunakan untuk klaster Anda, pertimbangkan konfigurasi simpul dari klaster dan jumlah data yang harus disimpan.

Konfigurasi cluster MemoryDB

Cluster MemoryDB terdiri dari 1 hingga 500 pecahan. Data dalam cluster MemoryDB dipartisi di seluruh pecahan di cluster. Aplikasi Anda terhubung dengan cluster MemoryDB menggunakan alamat jaringan yang disebut Endpoint. Selain titik akhir node, cluster MemoryDB sendiri memiliki titik akhir yang disebut titik akhir cluster. Aplikasi Anda dapat menggunakan endpoint ini untuk membaca dari atau menulis ke cluster, meninggalkan penentuan node mana yang akan dibaca atau ditulis hingga MemoryDB.

Peningkatan I/O Multiplexing

Jika Anda menjalankan Redis versi 7.0 atau lebih tinggi, Anda akan mendapatkan akselerasi tambahan dengan multiplexing I/O yang disempurnakan, di mana setiap jaringan pipa thread IO khusus memerintahkan dari beberapa klien ke mesin Redis, memanfaatkan kemampuan Redis untuk memproses perintah secara efisien dalam batch. Untuk informasi selengkapnya, lihat Performa ultra-cepat danthe section called “Tipe simpul yang didukung”.

Persyaratan penskalaan

Semua cluster dapat ditingkatkan tipe node yang lebih besar. Saat Anda meningkatkan kluster MemoryDB, Anda dapat melakukannya secara online sehingga cluster tetap tersedia atau Anda dapat menyemai cluster baru dari snapshot dan menghindari cluster baru mulai kosong.

Untuk informasi lain, lihat Penskalaan dalam panduan ini.

Persyaratan akses

Secara desain, cluster MemoryDB diakses dari instans Amazon EC2. Akses jaringan ke cluster MemoryDB terbatas pada akun yang membuat cluster. Oleh karena itu, sebelum Anda dapat mengakses cluster dari instans Amazon EC2, Anda harus mengotorisasi ingress ke cluster. Untuk instruksi terperinci, lihat Langkah 2: Otorisasi akses ke cluster dalam panduan ini.

Wilayah dan Zona Ketersediaan

Dengan menemukan cluster MemoryDB Anda di AWS Wilayah yang dekat dengan aplikasi Anda, Anda dapat mengurangi latensi. Jika klaster Anda memiliki beberapa node, menempatkan node Anda di Availability Zone yang berbeda dapat mengurangi dampak kegagalan pada klaster Anda.

Untuk informasi selengkapnya, lihat berikut ini:

- Memilih Wilayah dan Availability Zone
- Mengurangi Kegagalan

Membuat klaster

MemoryDB for Redis menawarkan tiga cara untuk membuat cluster. Untuk informasi selengkapnya, lihat Langkah 1: Buat cluster.

Melihat detail klaster

Anda dapat melihat informasi detail tentang satu atau beberapa cluster menggunakan konsol MemoryDB,AWS CLI, atau MemoryDB API.

Melihat detail untuk cluster MemoryDB (Konsol)

Prosedur berikut merinci cara melihat detail cluster MemoryDB menggunakan konsol MemoryDB.

1. Masuk ke AWS Management Console dan buka konsol MemoryDB for Redis di https://console.aws.amazon.com/memorydb/.
2. Untuk melihat detail cluster, pilih tombol radio di sebelah kiri nama cluster dan kemudian pilih Lihat detail. Anda juga dapat mengklik langsung pada cluster untuk melihat halaman detail cluster.

   Halaman detail Cluster menampilkan detail tentang cluster, termasuk titik akhir cluster. Anda dapat melihat detail lebih lanjut menggunakan beberapa tab yang tersedia di halaman detail Cluster.
3. Pilih tab Pecahan dan node untuk melihat daftar pecahan cluster dan jumlah node di setiap pecahan.
4. Untuk melihat informasi spesifik pada sebuah node, perluas pecahan pada tabel di bawah ini. Atau Anda juga dapat mencari pecahan menggunakan kotak pencarian.

   Melakukan hal ini menampilkan informasi tentang setiap node, termasuk Availability Zone, slot/keyspaces dan statusnya.
5. Pilih tab Metrik untuk memantau prosesnya masing-masing, seperti Pemanfaatan CPU dan Pemanfaatan CPU Mesin. Untuk informasi selengkapnya, lihat Metrik untuk MemoryDB.
6. Pilih tab Jaringan dan keamanan untuk melihat detail grup subnet dan grup keamanan.

   a. Di grup Subnet, Anda dapat melihat nama grup subnet, tautan ke VPC yang dimiliki subnet dan Nama Sumber Daya Amazon (ARN) grup subnet.

   b. Di Grup keamanan, Anda dapat melihat ID grup keamanan, nama, dan deskripsi.
7. Pilih tab Maintenace dan snapshot untuk melihat detail pengaturan snapshot.

   a. Di Snapshot, Anda dapat melihat apakah Snapshot Otomatis diaktifkan, periode retensi snapshot, dan jendela snapshot.

b. Di Snapshots, Anda akan melihat daftar snapshot apa pun ke cluster ini, termasuk nama snapshot, ukuran, jumlah pecahan, dan status.

Untuk informasi selengkapnya, lihat Snapshot dan pulihkan .

8. Pilih tab Maintenace dan snapshot untuk melihat detail Jendela Pemeliharaan, bersama dengan pembaruan ACL, Resharding, atau Layanan yang tertunda. Untuk informasi selengkapnya, lihat Mengelola pemeliharaan.
9. Pilih tab Pembaruan Layanan untuk melihat detail pembaruan layanan apa pun yang berlaku untuk klaster ini. Untuk informasi selengkapnya, lihat Layanan update di MemoryDB untuk Redis.
10. Pilih tab Tag untuk melihat detail tag sumber daya atau alokasi biaya apa pun yang terkait dengan klaster ini. Untuk informasi selengkapnya, lihat Menandai snapshot.

Melihat detail cluster (AWSCLI)

Anda dapat melihat detail untuk klaster menggunakan perintah AWS CLI `describe-clusters`. Jika parameter `--cluster-name` dihilangkan, perincian untuk beberapa klaster, hingga `--max-results`, akan dikeluarkan. Jika parameter `--cluster-name` disertakan, perincian untuk klaster tertentu akan dikeluarkan. Anda dapat membatasi jumlah catatan yang dikeluarkan dengan parameter `--max-results`.

Kode berikut menampilkan detail untuk `my-cluster`.

```
aws memorydb describe-clusters --cluster-name my-cluster
```

Kode berikut menampilkan detail hingga 25 klaster.

```
aws memorydb describe-clusters --max-results 25
```

Example

Untuk Linux, macOS, atau Unix:

```
aws memorydb describe-clusters \
    --cluster-name my-cluster \
    --show-shard-details
```

Untuk Windows:

```
aws memorydb describe-clusters ^
    --cluster-name my-cluster ^
    --show-shard-details
```

Output JSON berikut menunjukkan respons:

```
{
    "Clusters": [
        {
            "Name": "my-cluster",
            "Description": "my cluster",
            "Status": "available",
            "NumberOfShards": 1,
            "Shards": [
                {
                    "Name": "0001",
                    "Status": "available",
                    "Slots": "0-16383",
                    "Nodes": [
                        {
                            "Name": "my-cluster-0001-001",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1a",
                            "CreateTime": 1629230643.961,
                            "Endpoint": {
                                "Address": "my-cluster-0001-001.my-
cluster.abcdef.memorydb.us-east-1.amazonaws.com",
                                "Port": 6379
                            }
                        },
                        {
                            "Name": "my-cluster-0001-002",
                            "Status": "available",
                            "CreateTime": 1629230644.025,
                            "Endpoint": {
                                "Address": "my-cluster-0001-002.my-
cluster.abcdef.memorydb.us-east-1.amazonaws.com",
                                "Port": 6379
                            }
                        }
                    ],
                    "NumberOfNodes": 2
                }
```

```
            ],
            "ClusterEndpoint": {
                "Address": "clustercfg.my-cluster.abcdef.memorydb.us-
east-1.amazonaws.com",
                "Port": 6379
            },
            "NodeType": "db.r6g.large",
            "EngineVersion": "6.2",
            "EnginePatchVersion": "6.2.6",
            "ParameterGroupName": "default.memorydb-redis6",
            "ParameterGroupStatus": "in-sync",
            "SubnetGroupName": "default",
            "TLSEnabled": true,
            "ARN": "arn:aws:memorydb:us-east-1:000000000:cluster/my-cluster",
            "SnapshotRetentionLimit": 0,
            "MaintenanceWindow": "sat:06:30-sat:07:30",
            "SnapshotWindow": "04:00-05:00",
            "ACLName": "open-access",
            "DataTiering": "false",
            "AutoMinorVersionUpgrade": true,
        }
```

Untuk informasi lebih lanjut, lihat topik AWS CLI untuk MemoryDB. `describe-clusters`

Melihat detail cluster (MemoryDB API)

Anda dapat melihat detail untuk cluster menggunakan aksi API `DescribeClusters` MemoryDB. Jika parameter `ClusterName` disertakan, detail untuk klaster yang ditentukan akan dikeluarkan. Jika parameter `ClusterName` dihilangkan, detail untuk hingga `MaxResults` klaster (defaultnya 100) akan dikeluarkan. Nilai untuk `MaxResults` tidak boleh kurang dari 20 atau lebih besar dari 100.

Kode berikut menampilkan detail untuk `my-cluster`.

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=DescribeClusters
   &ClusterName=my-cluster
   &Version=2021-01-01
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20210802T192317Z
   &X-Amz-Credential=<credential>
```

Kode berikut menampilkan detail hingga 25 klaster.

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=DescribeClusters
   &MaxResults=25
   &Version=2021-02-02
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20210802T192317Z
   &X-Amz-Credential=<credential>
```

Untuk informasi selengkapnya, lihat topik referensi API MemoryDB. `DescribeClusters`

Memodifikasi cluster MemoryDB

Selain menambahkan atau menghapus node dari cluster, mungkin ada saat-saat di mana Anda perlu membuat perubahan lain ke cluster yang ada, seperti menambahkan grup keamanan, mengubah jendela pemeliharaan atau grup parameter.

Sebaiknya Anda mengatur jendela pemeliharaan Anda pada waktu penggunaan terendah. Jadi ini mungkin perlu diubah dari waktu ke waktu.

Saat Anda mengubah parameter klaster, perubahan tersebut diterapkan pada klaster tersebut dalam waktu segera. Ini berlaku terlepas dari apakah Anda mengubah grup parameter klaster itu sendiri atau nilai parameter di dalam grup parameter klaster.

Anda juga dapat memperbarui versi mesin cluster Anda. Misalnya, Anda dapat memilih versi minor engine baru dan MemoryDB akan segera memperbarui cluster Anda.

Menggunakan AWS Management Console

Untuk mengubah klaster

1. Masuk ke AWS Management Console dan buka konsol MemoryDB for Redis di https://console.aws.amazon.com/memorydb/.
2. Dari daftar di sudut kanan atas, pilih Wilayah AWS sesuai lokasi klaster yang akan diubah.
3. Dari navigasi kiri, pergi ke Clusters. Dari detail Clusters, pilih cluster menggunakan tombol radio dan pergi ke Actions dan kemudian Modify.
4. Halaman Modify muncul.
5. Di jendela Modify, buat modifikasi yang Anda inginkan. Pilihannya meliputi:

   - Deskripsi
   - Grup subnet
   - Grup Keamanan VPC
   - Tipe simpul

   > **Note**
   >
   > Jika cluster menggunakan tipe node dari keluarga r6gd, Anda hanya dapat memilih ukuran node yang berbeda dari dalam keluarga itu. Jika Anda memilih tipe node

dari keluarga r6gd, tiering data akan diaktifkan secara otomatis. Untuk informasi selengkapnya, lihat Tingkatan data.

- Kompatibilitas versi Redis
- Aktifkan snapshot otomatis
- Periode Retensi Snapshot
- Jendela Snapshot
- Jendela pemeliharaan
- Topik untuk Notifikasi SNS

6. Pilih Simpan perubahan.

Anda juga dapat pergi ke halaman detail Cluster dan klik modifikasi untuk membuat modifikasi pada cluster. Jika Anda ingin memodifikasi bagian tertentu dari cluster, Anda dapat pergi ke tab masing-masing di halaman detail Cluster dan klik Ubah.

Menggunakan AWS CLI

Anda dapat mengubah klaster yang telah ada menggunakan operasi AWS CLI `update-cluster`. Untuk mengubah nilai konfigurasi klaster, tentukan ID klaster, parameter yang akan diubah dan nilai baru parameter. Contoh berikut mengubah jendela pemeliharaan untuk klaster bernama `my-cluster` dan menerapkan perubahan itu segera.

Untuk Linux, macOS, atau Unix:

```
aws memorydb update-cluster \
    --cluster-name my-cluster \
    --preferred-maintenance-window sun:23:00-mon:02:00
```

Untuk Windows:

```
aws memorydb update-cluster ^
    --cluster-name my-cluster ^
    --preferred-maintenance-window sun:23:00-mon:02:00
```

Untuk informasi selengkapnya, lihat update-cluster di Command Reference. AWS CLI

Menggunakan MemoryDB API

Anda dapat memodifikasi cluster yang ada menggunakan operasi API UpdateClusterMemoryDB. Untuk mengubah nilai konfigurasi klaster, tentukan ID klaster, parameter yang akan diubah dan nilai baru parameter. Contoh berikut mengubah jendela pemeliharaan untuk klaster bernama `my-cluster` dan menerapkan perubahan itu segera.

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=UpdateCluster
    &ClusterName=my-cluster
    &PreferredMaintenanceWindow=sun:23:00-mon:02:00
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210801T220302Z
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Date=20210802T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

Menambahkan/Menghapus node dari cluster

Anda dapat menambahkan atau menghapus node dari cluster menggunakan APIAWS Management Console, theAWS CLI, atau MemoryDB.

Menggunakan AWS Management Console

1. Masuk ke AWS Management Console dan buka konsol MemoryDB for Redis di https://console.aws.amazon.com/memorydb/.
2. Dari daftar cluster, pilih nama cluster dari mana Anda ingin menambahkan atau menghapus node.
3. Di bawah tab Pecahan dan node, pilih Tambah/Hapus node
4. Di Jumlah node baru, masukkan jumlah node yang Anda inginkan.
5. Pilih Konfirmasi.

> **Important**
>
> Jika Anda mengatur jumlah node ke 1, Anda tidak akan lagi diaktifkan Multi-AZ. Anda juga dapat memilih untuk mengaktifkan Auto failover.

Menggunakan AWS CLI

1. Identifikasi nama-nama node yang ingin Anda hapus. Untuk informasi selengkapnya, lihat Melihat detail klaster .
2. Gunakan operasi CLI `update-cluster` dengan daftar simpul yang akan dihapus, seperti dalam contoh berikut.

   Untuk menghapus simpul dari klaster menggunakan antarmuka baris perintah, gunakan perintah `update-cluster` dengan parameter berikut:

   - `--cluster-name`ID dari cluster yang ingin Anda hapus node dari.
   - `--replica-configuration`— Memungkinkan Anda untuk mengatur jumlah replika:
     - `ReplicaCount`— Tetapkan properti ini untuk menentukan jumlah node replika yang Anda inginkan.
   - `--region` Menentukan Wilayah AWS dari klaster yang ingin dihapus simpulnya.

Untuk Linux, macOS, atau Unix:

```
aws memorydb update-cluster \
    --cluster-name my-cluster \
    --replica-configuration \
        ReplicaCount=1 \
    --region us-east-1
```

Untuk Windows:

```
aws memorydb update-cluster ^
    --cluster-name my-cluster ^
    --replica-configuration ^
        ReplicaCount=1 ^
    --region us-east-1
```

Untuk informasi lain, lihat topik AWS CLI `update-cluster`.

Menggunakan MemoryDB API

Untuk menghapus node menggunakan API MemoryDB, panggil operasi `UpdateCluster` API dengan nama cluster dan daftar node yang akan dihapus, seperti yang ditunjukkan:

- `ClusterName`ID dari cluster yang ingin Anda hapus node dari.
- `ReplicaConfiguration`— Memungkinkan Anda untuk mengatur jumlah replika:
  - `ReplicaCount`— Tetapkan properti ini untuk menentukan jumlah node replika yang Anda inginkan.
- `Region` Menentukan Wilayah AWS dari klaster yang ingin dihapus simpulnya.

Untuk informasi lebih lanjut, lihat UpdateCluster.

Mengakses klaster Anda

Instans MemoryDB untuk Redis Anda dirancang untuk diakses melalui instans Amazon EC2.

Anda dapat mengakses node MemoryDB Anda dari instans Amazon EC2 di Amazon VPC yang sama. Atau, dengan menggunakan peering VPC, Anda dapat mengakses node MemoryDB Anda dari Amazon EC2 di Amazon VPC yang berbeda.

Topik

Berikan akses ke klaster Anda

Anda dapat terhubung ke klaster MemoryDB 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.

Untuk memberikan masuknya 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. Di panel navigasi kiri, di bawahJaringan & KeamananpilihKelompok Keamanan.
3. Dari daftar grup keamanan, pilih grup keamanan untuk Amazon VPC Anda. Kecuali Anda membuat grup keamanan untuk penggunaan MemoryDB, grup keamanan ini akan diberi namadefault.
4. Pilih tab Masuk, dan kemudian lakukan hal berikut:
    a. Pilih Edit.
    b. Pilih Add rule (Tambahkan aturan).
    c. Pada kolom Jenis, pilih Aturan TCP khusus.
    d. Pada 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. DiSumberkotak, pilihDimana Sajayang memiliki rentang port (0.0.0.0/0) sehingga setiap instans Amazon EC2 yang Anda luncurkan dalam Amazon VPC dapat terhubung ke node MemoryDB Anda.

> **Important**
>
> Membuka klaster MemoryDB ke 0.0.0.0/0 tidak mengekspos cluster ke Internet karena tidak memiliki alamat IP publik dan karena itu 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 kebetulan menjalankan sesuatu pada port default, maka layanan tersebut dapat terekspos secara tak disengaja. Oleh karena itu, kami sarankan membuat Grup Keamanan VPC yang akan digunakan secara eksklusif oleh MemoryDB. Untuk informasi lain, lihat .Grup Keamanan Khusus.

f. Pilih Save (Simpan).

Saat Anda meluncurkan instans Amazon EC2 ke Amazon VPC Anda, instans tersebut akan dapat terhubung ke klaster MemoryDB Anda.

Mengakses sumber daya MemoryDB dari luarAWS

MemoryDB adalah layanan yang dirancang untuk digunakan secara internal untuk VPC Anda. Akses eksternal tidak disarankan karena latensi lalu lintas Internet dan masalah keamanan. Namun, jika akses eksternal ke MemoryDB diperlukan untuk tujuan pengujian atau pengembangan, itu dapat dilakukan melalui VPN.

MenggunakanAWSClient VPN, Anda mengizinkan akses eksternal ke node MemoryDB Anda dengan manfaat sebagai berikut:

- Akses terbatas hanya untuk pengguna atau kunci autentikasi yang disetujui;
- Lalu lintas terenkripsi antara Klien VPN dan titik akhir VPN AWS;
- Akses yang terbatas ke subnet atau simpul tertentu;
- Pencabutan akses dengan mudah dari pengguna atau kunci autentikasi;
- Mengaudit koneksi;

Prosedur berikut menunjukkan cara untuk:

Topik

Membuat otoritas sertifikat

Dimungkinkan untuk membuat Otoritas Sertifikat (CA) menggunakan teknik atau alat yang berbeda. Dianjurkan menggunakan utilitas easy-rsa, yang disediakan oleh proyek OpenVPN. Terlepas opsi pilihan Anda, pastikan untuk menjaga kunci tetap aman. Prosedur berikut mengunduh skrip easy-RSA, membuat Otoritas Sertifikat dan kunci untuk autentikasi klien VPN pertama:

- Untuk membuat sertifikat awal, buka terminal dan lakukan hal berikut:
  - `git clone` https://github.com/OpenVPN/easy-rsa
  - `cd easy-rsa`
  - `./easyrsa3/easyrsa init-pki`
  - `./easyrsa3/easyrsa build-ca nopass`

- `./easyrsa3/easyrsa build-server-full server nopass`
- `./easyrsa3/easyrsa build-client-full client1.domain.tld nopass`

Subdirektori pki yang berisi sertifikat akan dibuat di bawah easy-rsa.

- Kirim sertifikat server ke AWS Certificate Manager (ACM):
  - Pada konsol ACM, pilih Certificate Manager.
  - Pilih Impor Sertifikat.
  - Masukkan sertifikat kunci publik yang tersedia di dalam file `easy-rsa/pki/issued/server.crt` pada bidang Tubuh sertifikat.
  - Tempelkan kunci privat yang tersedia di dalam `easy-rsa/pki/private/server.key` pada bidang Kunci privat sertifikat. Pastikan untuk mengeblok semua baris di antara `BEGIN AND END PRIVATE KEY` (termasuk baris `BEGIN` dan `END`).
  - Tempelkan kunci publik CA yang tersedia pada file `easy-rsa/pki/ca.crt` pada bidang Rantai sertifikat.
  - Pilih Tinjau dan impor.
  - Pilih Impor.

  Untuk mengirim sertifikat server ke ACM menggunakan CLI AWS, jalankan perintah berikut:
  `aws acm import-certificate --certificate fileb://easy-rsa/pki/issued/server.crt --private-key file://easy-rsa/pki/private/server.key --certificate-chain file://easy-rsa/pki/ca.crt --region region`

  Perhatikan ARN sertifikat untuk penggunaan di masa mendatang.

Membuat konfigurasi komponen AWS client VPN

MenggunakanAWSKonsol

Pada konsol AWS, pilih Layanan dan kemudian VPC.

Di bawah Virtual Private Network, pilih Titik akhir Client VPN dan lakukan hal berikut:

KonfigurasiAWSKomponen Client VPN

- Pilih Buat Titik Akhir Client VPN.
- Tentukan opsi berikut:

- CIDR IPv4 Klien: gunakan jaringan privat dengan netmask setidaknya pada rentang /22. Pastikan bahwa subnet yang dipilih tidak bertentangan dengan alamat dari jaringan VPC. Contoh: 10.0.0.0/22.
- Pada ARN sertifikat server, pilih ARN dari sertifikat yang sebelumnya diimpor.
- Pilih Gunakan autentikasi bersama.
- Pada ARN sertifikat klien, pilih ARN dari sertifikat yang sebelumnya diimpor.
- Pilih Buat Titik Akhir Client VPN.

MenggunakanAWS CLI

Jalankan perintah berikut:

```
aws ec2 create-client-vpn-endpoint --client-cidr-block
"10.0.0.0/22" --server-certificate-arn arn:aws:acm:us-
east-1:012345678912:certificate/0123abcd-ab12-01a0-123a-123456abcdef --
authentication-options Type=certificate-
authentication,,MutualAuthentication={ClientRootCertificateChainArn=arn:aws:acm:
east-1:012345678912:certificate/123abcd-ab12-01a0-123a-123456abcdef} --
connection-log-options Enabled=false
```

Contoh keluaran:

```
"ClientVpnEndpointId": "cvpn-endpoint-0123456789abcdefg",
"Status": { "Code": "pending-associate" }, "DnsName": "cvpn-
endpoint-0123456789abcdefg.prod.clientvpn.us-east-1.amazonaws.com" }
```

Kaitkan jaringan target ke titik akhir VPN

- Pilih titik akhir baru VPN, dan kemudian pilih tab Asosiasi.
- Pilih Associate dan tentukan opsi berikut.
  - VPC: Pilih VPC MemoryDB Cluster.
  - Pilih salah satu jaringan klaster MemoryDB. Jika ragu, tinjau jaringan diGrup Subjaringandi dasbor MemoryDB.
  - Pilih Associate. Jika perlu, ulangi langkah tersebut untuk jaringan yang tersisa.

MenggunakanAWS CLI

Jalankan perintah berikut:

```
aws ec2 associate-client-vpn-target-network --client-vpn-endpoint-id cvpn-
endpoint-0123456789abcdefg --subnet-id subnet-0123456789abdcdef
```

Contoh keluaran:

```
"Status": { "Code": "associating" }, "AssociationId": "cvpn-
assoc-0123456789abdcdef" }
```

Tinjau grup keamanan VPN

Titik akhir VPN akan secara otomatis mengadopsi grup keamanan default dari VPC. Periksa aturan masuk dan keluar dan konfirmasikan apakah grup keamanan mengizinkan lalu lintas dari jaringan VPN (didefinisikan pada pengaturan Endpoint VPN) ke jaringan MemoryDB pada port layanan (secara default, 6379 untuk Redis).

Jika Anda perlu mengubah grup keamanan yang ditetapkan untuk titik akhir VPN, lanjutkan sebagai berikut:

- Pilih grup keamanan saat ini.
- Pilih Terapkan Grup Keamanan.
- Pilih Grup Keamanan baru.

MenggunakanAWS CLI

Jalankan perintah berikut:

```
aws ec2 apply-security-groups-to-client-vpn-target-network --
client-vpn-endpoint-id cvpn-endpoint-0123456789abcdefga  --vpc-id
vpc-0123456789abdcdef --security-group-ids sg-0123456789abdcdef
```

Contoh keluaran:

```
"SecurityGroupIds": [ "sg-0123456789abdcdef" ] }
```

> **Note**
>
> Grup keamanan MemoryDB juga perlu mengizinkan lalu lintas yang berasal dari klien VPN. Alamat klien akan ditutupi dengan alamat titik akhir VPN, sesuai Jaringan VPC. Oleh karena

itu, pertimbangkan jaringan VPC (bukan jaringan Klien VPN) saat membuat aturan masuk pada grup keamanan MemoryDB.

Otorisasi akses VPN ke jaringan tujuan

Pada tab Otorisasi, pilih Izinkan Masuk dan tentukan hal berikut:

- Jaringan tujuan untuk mengaktifkan akses: Gunakan 0.0.0.0/0 untuk mengizinkan akses ke jaringan apa pun (termasuk Internet) atau membatasi jaringan/host MemoryDB.
- Di bawah Berikan akses ke:, pilih Izinkan akses ke semua pengguna.
- Pilih Tambahkan Aturan Otorisasi.

MenggunakanAWS CLI

Jalankan perintah berikut:

```
aws ec2 authorize-client-vpn-ingress --client-vpn-endpoint-id cvpn-
endpoint-0123456789abcdefg --target-network-cidr 0.0.0.0/0 --authorize-all-
groups
```

Contoh keluaran:

```
{ "Status": { "Code": "authorizing" } }
```

Mengizinkan akses ke Internet dari klien VPN

Jika Anda perlu menelusuri Internet melalui VPN, Anda perlu membuat rute tambahan. Pilih tab Tabel Rute dan kemudian pilih Buat Rute:

- Tujuan rute: 0.0.0.0/0
- Target VPC Subnet ID: Pilih salah satu subnet terkait dengan akses ke Internet.
- Pilih Buat Rute.

MenggunakanAWS CLI

Jalankan perintah berikut:

```
aws ec2 create-client-vpn-route --client-vpn-endpoint-id cvpn-
endpoint-0123456789abcdefg --destination-cidr-block 0.0.0.0/0 --target-vpc-
subnet-id subnet-0123456789abdcdef
```

Contoh keluaran:

```
{ "Status": { "Code": "creating" } }
```

Konfigurasi klien VPN

Pada Dasbor AWS Client VPN, pilih titik akhir VPN yang baru saja dibuat dan pilih Unduh Konfigurasi Klien. Salin file konfigurasi, serta file `easy-rsa/pki/issued/client1.domain.tld.crt` dan `easy-rsa/pki/private/client1.domain.tld.key`. Edit file konfigurasi dan ubah atau tambahkan parameter berikut:

- cert: tambahkan baris baru dengan parameter cert menunjuk ke file `client1.domain.tld.crt`. Gunakan jalur lengkap ke file. Contoh: `cert /home/user/.cert/client1.domain.tld.crt`
- cert: key: tambahkan baris baru dengan kunci parameter menunjuk ke file `client1.domain.tld.key`. Gunakan jalur lengkap ke file. Contoh: `key /home/user/.cert/client1.domain.tld.key`

Tetapkan koneksi VPN dengan perintah: `sudo openvpn --config downloaded-client-config.ovpn`

Mencabut akses

Jika Anda perlu untuk membatalkan akses dari kunci klien tertentu, kunci tersebut perlu dicabut di CA. Kemudian kirimkan daftar pencabutan ke AWS Client VPN.

Mencabut kunci dengan easy-rsa:

- `cd easy-rsa`
- `./easyrsa3/easyrsa revoke client1.domain.tld`
- Masukkan “ya” untuk melanjutkan, atau masukkan input lain apa pun untuk membatalkan.

  `Continue with revocation: `yes` ... * `./easyrsa3/easyrsa gen-crl`
- CRL yang diperbarui telah dibuat. File CRL: `/home/user/easy-rsa/pki/crl.pem`

Mengimpor daftar pencabutan ke AWS Client VPN:

- Pada AWS Management Console, pilih Layanan dan kemudian VPC.
- Pilih Titik Akhir Client VPN.
- Pilih Titik Akhir Client VPN dan kemudian pilih Tindakan -> Impor CRL Sertifikat Klien.
- Tempelkan isi dari file `crl.pem`.

MenggunakanAWS CLI

Jalankan perintah berikut:

```
aws ec2 import-client-vpn-client-certificate-revocation-list --certificate-revocation-list file://./easy-rsa/pki/crl.pem --client-vpn-endpoint-id cvpn-endpoint-0123456789abcdefg
```

Contoh keluaran:

```
Example output: { "Return": true }
```

Menemukan titik akhir sambungan

Aplikasi Anda terhubung ke klaster Anda menggunakan titik akhir. Titik akhir adalah alamat unik klaster. Gunakan klasterTitik Akhir klasteruntuk semua operasi.

Bagian berikut memandu Anda menemukan titik akhir yang Anda perlukan.

Menemukan Endpoint untuk Cluster MemoryDB (AWS Management Console)

Untuk menemukan endpoint cluster MemoryDB

1. Masuk keAWS Management Consoledan buka MemoryDB untuk konsol Redis dihttps://console.aws.amazon.com/memorydb/.
2. Di panel navigasi, pilihklaster.

   Layar klaster akan muncul dengan daftar klaster. Pilih klaster yang ingin Anda sambungkan.
3. Untuk menemukan endpoint klaster, pilih nama klaster (bukan tombol radio).
4. ParameterEndpoint klasterditampilkan di bawahDetail klaster. Untuk menyalinnya, pilihmenyontekikon di sebelah kiri titik akhir.

Menemukan Endpoint untuk Cluster MemoryDB (AWSCLI)

Anda dapat menggunakan`describe-clusters`perintah untuk menemukan endpoint untuk sebuah cluster. Perintah tersebut mengembalikan endpoint klaster.

Operasi berikut mengambil endpoint, yang dalam contoh ini direpresentasikan sebagai*`sampel`*, untuk cluster`mycluster`.

Ia mengembalikan respon JSON berikut:

```
aws memorydb describe-clusters \
  --cluster-name mycluster
```

Untuk Windows:

```
aws memorydb describe-clusters ^
   --cluster-name mycluster
```

```
{
    "Clusters": [
        {
            "Name": "my-cluster",
            "Status": "available",
            "NumberOfShards": 1,
            "ClusterEndpoint": {
```

```
                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                "Port": 6379
            },
            "NodeType": "db.r6g.large",
            "EngineVersion": "6.2",
            "EnginePatchVersion": "6.2.4",
            "ParameterGroupName": "default.memorydb-redis6",
            "ParameterGroupStatus": "in-sync",
            "SubnetGroupName": "my-sg",
            "TLSEnabled": true,
            "ARN": "arn:aws:memorydb:us-east-1:zzzexamplearn:cluster/my-cluster",
            "SnapshotRetentionLimit": 0,
            "MaintenanceWindow": "wed:03:00-wed:04:00",
            "SnapshotWindow": "04:30-05:30",
            "ACLName": "my-acl",
            "AutoMinorVersionUpgrade": true
        }
    ]
}
```

Untuk informasi selengkapnya, lihatdescribe-klaster.

Menemukan Endpoint untuk Cluster MemoryDB (MemoryDB API)

Anda dapat menggunakan MemoryDB for Redis API untuk menemukan titik akhir klaster.

Menemukan Endpoint untuk Cluster MemoryDB (MemoryDB API)

Anda dapat menggunakan API MemoryDB untuk menemukan endpoint untuk cluster dengan`DescribeClusters`tindakan. Tindakan tersebut mengembalikan endpoint klaster.

Operasi berikut mengambil endpoint cluster untuk klaster`mycluster`.

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=DescribeClusters
    &ClusterName=mycluster
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210802T192317Z
    &Version=2021-01-01
    &X-Amz-Credential=<credential>
```

Untuk informasi selengkapnya, lihatDescribeClusters.

Bekerja dengan Serpihan

Pecahan adalah kumpulan satu sampai 6 node. Anda dapat membuat klaster dengan jumlah serpihan lebih banyak dan jumlah replika lebih sedikit dengan jumlah hingga 500 simpul per klaster. Konfigurasi klaster ini dapat berkisar dari 500 serpihan dan 0 replika hingga 100 serpihan dan 5 replika, yang merupakan jumlah replika maksimum yang diizinkan. Data klaster dipartisi di seluruh serpihan klaster. Jika di dalam serpihan terdapat lebih dari satu simpul, maka serpihan mengimplementasikan replikasi dengan satu simpul menjadi simpul primer baca/tulis dan simpul lain menjadi simpul replika baca-saja.

Saat Anda membuat klaster MemoryDB menggunakanAWS Management Console, Anda menentukan jumlah serpihan dalam klaster dan jumlah simpul dalam serpihan. Untuk informasi selengkapnya, lihat Membuat cluster MemoryDB.

Setiap simpul dalam serpihan memiliki spesifikasi penyimpanan dan memori komputasi yang sama. API MemoryDB memungkinkan Anda mengontrol atribut di tingkat klaster, seperti jumlah simpul, pengaturan keamanan, dan jendela pemeliharaan sistem.

Untuk informasi selengkapnya, lihat Resharding dan penyeimbangan ulang serpihan secara offline untuk MemoryDB dan Resharding dan penyeimbangan ulang serpihan secara online untuk MemoryDB.

Menemukan nama serpihan

Anda dapat menemukan nama shard menggunakanAWS Management Console, AWS CLI atau API MemoryDB.

Menggunakan AWS Management Console

Prosedur berikut menggunakan AWS Management Console untuk menemukan nama pecahan cluster MemoryDB ini.

1. Masuk ke AWS Management Console dan buka konsol MemoryDB untuk Redis di https://console.aws.amazon.com/memorydb/.
2. Di panel navigasi sebelah kiri, pilih Klaster.
3. Pilih cluster di bawah Nama yang nama beling yang ingin Anda temukan.
4. Di bawah tab Pecahan dan node, lihat daftar pecahan di bawah Nama. Anda juga dapat memperluas masing-masing untuk melihat detail node mereka.

Menggunakan AWS CLI

Untuk menemukan shard (shard) nama untuk cluster MemoryDB menggunakan AWS CLI operasi `describe-clusters` dengan parameter opsional berikut.

- **`--cluster-name`**—Parameter opsional yang bila digunakan akan membatasi output pada detail klaster yang ditentukan. Jika parameter ini dihilangkan, detail hingga 100 klaster akan dihasilkan.
- **`--show-shard-details`**—Mengembalikan rincian pecahan, termasuk nama mereka.

Perintah ini akan menghasilkan detail untuk `my-cluster`.

Untuk Linux, macOS, atau Unix:

```
aws memorydb describe-clusters \
    --cluster-name my-cluster
    --show-shard-details
```

Untuk Windows:

```
aws memorydb describe-clusters ^
    --cluster-name my-cluster
    --show-shard-details
```

Ia mengembalikan respon JSON berikut:

Jeda baris ditambahkan untuk kemudahan membaca.

```
{
    "Clusters": [
        {
            "Name": "my-cluster",
            "Status": "available",
            "NumberOfShards": 1,
            "Shards": [
                {
                    "Name": "0001",
                    "Status": "available",
                    "Slots": "0-16383",
                    "Nodes": [
                        {
                            "Name": "my-cluster-0001-001",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1a",
                            "CreateTime": "2021-08-21T20:22:12.405000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        },
                        {
                            "Name": "my-cluster-0001-002",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1b",
                            "CreateTime": "2021-08-21T20:22:12.405000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
```

```
                            }
                        }
                    ],
                    "NumberOfNodes": 2
                }
            ],
            "ClusterEndpoint": {
                "Address": "clustercfg.my-cluster.xxxxx.memorydb.us-
east-1.amazonaws.com",
                "Port": 6379
            },
            "NodeType": "db.r6g.large",
            "EngineVersion": "6.2",
            "EnginePatchVersion": "6.2.6",
            "ParameterGroupName": "default.memorydb-redis6",
            "ParameterGroupStatus": "in-sync",
            "SubnetGroupName": "my-sg",
            "TLSEnabled": true,
            "ARN": "arn:aws:memorydb:us-east-1:xxxxxexamplearn:cluster/my-cluster",
            "SnapshotRetentionLimit": 0,
            "MaintenanceWindow": "wed:03:00-wed:04:00",
            "SnapshotWindow": "04:30-05:30",
            "ACLName": "my-acl",
            "DataTiering": "false",
            "AutoMinorVersionUpgrade": true
        }
    ]
}
```

Menggunakan API MemoryDB

Untuk menemukan id pecahan untuk cluster MemoryDB menggunakan operasi API `DescribeClusters` dengan parameter opsional berikut.

- **`ClusterName`**—Parameter opsional yang bila digunakan akan membatasi output pada detail klaster yang ditentukan. Jika parameter ini dihilangkan, detail hingga 100 klaster akan dihasilkan.
- **`ShowShardDetails`**—Mengembalikan rincian pecahan, termasuk nama mereka.

Example

Perintah ini akan menghasilkan detail untuk `my-cluster`.

Untuk Linux, macOS, atau Unix:

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=DescribeClusters
   &ClusterName=sample-cluster
   &ShowShardDetails=true
   &Version=2021-01-01
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20210802T192317Z
   &X-Amz-Credential=<credential>
```

Mengelola implementasi MemoryDB Anda

Di bagian ini, Anda dapat menemukan detail tentang cara mengelola berbagai komponen implementasi MemoryDB Anda.

Topik

Versi mesin Redis

Bagian ini mencakup versi mesin Redis yang didukung.

Topik

MemoryDB untuk Redis versi 7.1 (ditingkatkan)

MemoryDB untuk Redis versi 7.1 menambahkan dukungan untuk kemampuan pencarian vektor dalam pratinjau untuk wilayah tertentu, serta perbaikan bug penting dan peningkatan kinerja.

- Fitur Pencarian Vektor: Pencarian vektor dapat digunakan dengan fungsionalitas MemoryDB yang ada. Aplikasi yang tidak menggunakan pencarian vektor tidak akan terpengaruh oleh keberadaannya. Pratinjau pencarian vektor tersedia di MemoryDB untuk Redis versi 7.1 dan seterusnya di wilayah berikut: AS Timur (Virginia N. dan Ohio), AS Barat (Oregon), UE (Irlandia), dan Asia Pasifik (Tokyo). Silakan lihat dokumentasi di sini untuk cara mengaktifkan pratinjau pencarian vektor dan kemampuan fitur terkait.

> **Note**
>
> MemoryDB untuk Redis versi 7.1 kompatibel dengan OSS Redis v7.0. Untuk informasi lebih lanjut tentang rilis Redis 7.0, lihat Catatan Rilis Redis 7.0 di Redis on. GitHub

MemoryDB untuk Redis versi 7.0 (ditingkatkan)

MemoryDB untuk Redis 7.0 menambahkan sejumlah perbaikan dan dukungan untuk fungsionalitas baru:

- Fungsi Redis: MemoryDB untuk Redis 7 menambahkan dukungan untuk Fungsi Redis, dan memberikan pengalaman terkelola yang memungkinkan pengembang untuk mengeksekusi skrip LUA dengan logika aplikasi yang disimpan di cluster MemoryDB, tanpa mengharuskan klien untuk mengirim ulang skrip ke server dengan setiap koneksi.
- Perbaikan ACL: MemoryDB untuk Redis 7 menambahkan dukungan untuk versi berikutnya dari Redis Access Control Lists (ACL). Dengan MemoryDB untuk Redis 7, klien sekarang dapat menentukan beberapa set izin pada kunci atau ruang kunci tertentu di Redis.
- Sharded Pub/Sub: MemoryDB untuk Redis 7 menambahkan dukungan untuk menjalankan fungsionalitas Redis Pub/Sub dengan cara sharded saat menjalankan MemoryDB di Cluster Mode Enabled (CME). Kemampuan Redis Pub/Sub memungkinkan penerbit untuk mengeluarkan pesan ke sejumlah pelanggan di saluran. Dengan Amazon MemoryDB untuk Redis 7, saluran terikat pada pecahan di cluster MemoryDB, menghilangkan kebutuhan untuk menyebarkan informasi saluran di seluruh pecahan. Ini menghasilkan peningkatan skalabilitas.
- Multiplexing I/O yang disempurnakan: MemoryDB untuk 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 cluster MemoryDB. Misalnya, saat menggunakan cluster node r6g.4xlarge dan menjalankan 5200 klien bersamaan, Anda dapat mencapai peningkatan throughput hingga 46% (operasi baca dan

tulis per detik) dan latensi P99 menurun hingga 21%, dibandingkan dengan MemoryDB untuk Redis versi 6.

Untuk informasi lebih lanjut tentang rilis Redis 7.0, lihat Catatan Rilis Redis 7.0 di Redis on. GitHub

MemoryDB untuk Redis versi 6.2 (ditingkatkan)

MemoryDB memperkenalkan versi berikutnya dari mesin Redis, yang meliputi, dukungan upgrade versi otomatisMengautentikasi pengguna dengan Daftar Kontrol Akses (ACL), caching sisi klien dan peningkatan operasional yang signifikan.

Redis engine versi 6.2.6 juga memperkenalkan dukungan untuk format asli JavaScript Object Notation (JSON), cara sederhana dan tanpa skema untuk menyandikan kumpulan data kompleks di dalam kluster Redis. Dengan dukungan JSON, Anda dapat memanfaatkan performa dan API Redis untuk aplikasi yang beroperasi melalui JSON. Untuk informasi selengkapnya, lihat Memulai dengan JSON. Juga termasuk metrik terkait JSON `JsonBasedCmds` yang dimasukkan ke dalam CloudWatch untuk memantau penggunaan tipe data ini. Untuk informasi selengkapnya, lihat Metrik untuk MemoryDB.

Dengan Redis 6, MemoryDB akan menawarkan satu versi untuk setiap rilis minor Redis OSS, daripada menawarkan beberapa versi patch. Ini dirancang untuk meminimalkan kebingungan dan ambiguitas karena harus memilih dari beberapa versi minor. MemoryDB juga akan secara otomatis mengelola versi minor dan patch dari cluster yang sedang berjalan, memastikan peningkatan kinerja dan keamanan yang ditingkatkan. Ini akan ditangani melalui saluran pemberitahuan pelanggan standar melalui kampanye pembaruan layanan. Untuk informasi selengkapnya, lihat Layanan update di MemoryDB untuk Redis.

Jika Anda tidak menentukan versi mesin selama pembuatan, MemoryDB akan secara otomatis memilih versi Redis yang disukai untuk Anda. Di sisi lain, jika Anda menentukan versi mesin dengan menggunakan`6.2`, MemoryDB akan secara otomatis memanggil versi patch pilihan Redis 6.2 yang tersedia.

Misalnya, saat Anda membuat cluster, Anda mengatur `--engine-version` parameter ke`6.2`. Cluster akan diluncurkan dengan versi patch pilihan yang tersedia saat ini pada waktu pembuatan. Setiap permintaan dengan nilai versi mesin lengkap akan ditolak, pengecualian akan dilemparkan dan prosesnya akan gagal.

Saat memanggil `DescribeEngineVersions` API, nilai `EngineVersion` parameter akan disetel ke 6.2 dan versi mesin lengkap yang sebenarnya akan dikembalikan di `EnginePatchVersion` bidang.

Untuk informasi lebih lanjut tentang rilis Redis 6.2, lihat Catatan Rilis Redis 6.2 di Redis on. GitHub

Meningkatkan versi mesin

MemoryDB secara default secara otomatis mengelola versi patch cluster Anda yang sedang berjalan melalui pembaruan layanan. Anda juga dapat memilih keluar dari upgrade versi auto minor jika Anda menyetel `AutoMinorVersionUpgrade` properti cluster Anda ke false. Namun, Anda tidak dapat memilih keluar dari upgrade versi patch otomatis.

Anda dapat mengontrol jika dan kapan perangkat lunak yang sesuai dengan protokol yang mendukung klaster Anda ditingkatkan ke versi baru yang didukung oleh MemoryDB sebelum peningkatan otomatis dimulai. 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.

Anda dapat memulai upgrade versi engine ke cluster Anda dengan cara berikut:

- Dengan memperbaruinya dan menentukan versi mesin baru. Untuk informasi selengkapnya, lihat Memodifikasi cluster MemoryDB.
- Menerapkan pembaruan layanan untuk versi mesin yang sesuai. Untuk informasi selengkapnya, lihat Layanan update di MemoryDB untuk Redis.

Perhatikan hal berikut:

- Anda dapat meningkatkan ke versi mesin yang lebih baru, tetapi Anda tidak dapat menurunkan ke versi mesin yang lebih lama. Jika Anda ingin menggunakan versi mesin sebelumnya, Anda harus menghapus klaster yang telah ada dan membuatnya lagi dengan versi mesin sebelumnya.
- Kami merekomendasikan upgrade secara berkala ke versi utama terbaru, karena sebagian besar perbaikan besar tidak kembali porting ke versi yang lebih lama. Karena MemoryDB memperluas ketersediaan ke AWS Wilayah baru, MemoryDB mendukung dua `MAJOR.MINOR` versi terbaru pada waktu itu untuk Wilayah baru. Misalnya, jika AWS wilayah baru diluncurkan dan `MAJOR.MINOR` MemoryDB terbaru untuk versi Redis adalah 7.0 dan 6.2, MemoryDB untuk Redis akan mendukung versi 7.0 dan 6.2 di Wilayah baru. AWS Ketika `MAJOR.MINOR` versi baru dari MemoryDB untuk Redis dirilis, MemoryDB akan terus menambahkan dukungan untuk MemoryDB yang baru dirilis untuk Versi Redis. Untuk mempelajari lebih lanjut tentang memilih Wilayah untuk MemoryDB, lihat. Wilayah & titik akhir yang didukung

- Pengelolaan versi mesin dirancang agar Anda dapat memiliki kontrol sebanyak mungkin terkait cara melakukan patching. Namun, MemoryDB berhak untuk menambal cluster Anda atas nama Anda jika terjadi kerentanan keamanan kritis dalam sistem atau perangkat lunak.
- MemoryDB akan menawarkan satu versi untuk setiap rilis minor Redis OSS, daripada menawarkan beberapa versi patch. Ini dirancang untuk meminimalkan kebingungan dan ambiguitas karena harus memilih dari beberapa versi. MemoryDB juga akan secara otomatis mengelola versi minor dan patch dari cluster yang sedang berjalan, memastikan peningkatan kinerja dan keamanan yang ditingkatkan. Ini akan ditangani melalui saluran pemberitahuan pelanggan standar melalui kampanye pembaruan layanan. Untuk informasi selengkapnya, lihat Layanan update di MemoryDB untuk Redis.
- Anda dapat meng-upgrade versi cluster Anda dengan downtime 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.
- Kami menyarankan Anda melakukan upgrade mesin selama periode lalu lintas tulis masuk rendah.

  Cluster dengan beberapa pecahan diproses dan ditambal sebagai berikut:

  - Hanya satu operasi peningkatan yang dilakukan per pecahan 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.

Topik

- Cara meningkatkan versi mesin
- Menyelesaikan peningkatan mesin Redis yang diblokir

Cara meningkatkan versi mesin

Anda memulai upgrade versi ke cluster Anda dengan memodifikasinya menggunakan konsol MemoryDB, API AWS CLI, atau MemoryDB API dan menentukan versi engine yang lebih baru. Untuk informasi selengkapnya, lihat topik berikut.

- Menggunakan AWS Management Console

- Menggunakan AWS CLI
- Menggunakan MemoryDB API

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

Memulai dengan JSON

MemoryDB mendukung asli JavaScript Object Notation (JSON) format, sederhana, cara schemaless untuk mengkodekan dataset kompleks dalam cluster Redis. Anda dapat menyimpan dan mengakses data secara native menggunakan format JavaScript Object Notation (JSON) di dalam klaster Redis dan memperbarui data JSON yang tersimpan dalam klaster tersebut, tanpa perlu mengelola kode kustom untuk membuat serial dan deserialisasi.

Selain memanfaatkan Redis API untuk aplikasi yang beroperasi melalui JSON, Anda sekarang dapat secara efisien mengambil dan memperbarui bagian-bagian tertentu dari dokumen JSON tanpa perlu memanipulasi seluruh objek, yang dapat meningkatkan kinerja dan mengurangi biaya. Anda juga dapat mencari isi dokumen JSON Anda menggunakan kueri gaya GoessnerJSONPath.

Setelah membuat cluster dengan versi mesin yang didukung, tipe data JSON dan perintah terkait secara otomatis tersedia. Ini kompatibel dengan API dan kompatibel dengan RDB dengan modul RedisJSON versi 2, sehingga Anda dapat dengan mudah memigrasi aplikasi Redis berbasis JSON yang ada ke MemoryDB. Untuk informasi selengkapnya tentang Redis Redis yang didukung, lihatPerintah yang didukung.

Metrik terkait JSON `JsonBasedCmds` dimasukkan ke dalam CloudWatch untuk memantau penggunaan tipe data ini. Untuk informasi selengkapnya, lihat Metrik untuk MemoryDB.

> **Note**
>
> Untuk menggunakan JSON, Anda harus menjalankan Redis versi 6.2.6 atau versi versi versi 6.2.6 atau versi versi versi 6.2.6 atau versi versi versi versi 6.2.6 atau versi versi versi versi

Topik

Ikhtisar Redis JSON Datatype

MemoryDB mendukung sejumlah perintah Redis untuk bekerja dengan tipe data JSON. Berikut ini adalah gambaran dari tipe data JSON dan daftar rinci perintah Redis yang didukung.

Terminologi

Jangka waktu	Deskripsi
Dokumen JSON	mengacu pada nilai kunci Redis JSON
Nilai JSON	mengacu pada subset dari Dokumen JSON, termasuk akar yang mewakili seluruh dokumen. Sebuah nilai bisa berupa wadah atau entri dalam sebuah kontainer
Elemen JSON	setara dengan nilai JSON

Standar JSON yang didukung

Format JSON sesuai dengan standar pertukaran data RFC 7159 dan ECMA-404 JSON. UTF-8 Unicode dalam teks JSON didukung.

Elemen akar

Elemen akar dapat berupa jenis data JSON. Perhatikan bahwa di RFC 4627 sebelumnya, hanya objek atau array yang diizinkan sebagai nilai root. Sejak pembaruan ke RFC 7159, akar dokumen JSON dapat berupa tipe data JSON apa pun.

Batas ukuran dokumen

Dokumen JSON disimpan secara internal dalam format yang dioptimalkan untuk akses cepat dan modifikasi. Format ini biasanya menghasilkan mengkonsumsi memori yang agak lebih banyak daripada representasi serial setara dari dokumen yang sama. Konsumsi memori oleh satu dokumen JSON dibatasi hingga 64MB, yang merupakan ukuran struktur data dalam memori, bukan string JSON. Jumlah memori yang dikonsumsi oleh dokumen JSON dapat diperiksa dengan menggunakan perintah`JSON.DEBUG MEMORY`.

ACL JSON

- JSON datatype sepenuhnya terintegrasi ke dalam Redis Access Control Lists (ACL) kemampuan. Mirip dengan kategori per-datatype yang ada (@string, @hash, dll.) Kategori baru @json ditambahkan untuk menyederhanakan pengelolaan akses ke perintah dan data JSON. Tidak ada perintah Redis lain yang ada adalah anggota dari kategori @json. Semua perintah JSON memberlakukan batasan dan izin keyspace 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 di bawah 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		

Perintah JSON	@read	@write	@fast	@slow	@admin
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.LUPA		y	y		
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		

Perintah JSON	@read	@write	@fast	@slow	@admin
JSON.SET		y		y	
JSON.STRAPPEND		y	y		
JSON.STRLEN	y		y		
JSON.STRLEN	y		y		
JSON.TOGGLE		y	y		
JSON.JENIS	y		y		
JSON.NUMINCRBY		y	y		

Batas kedalaman tumpuk

Ketika sebuah objek JSON atau array memiliki elemen yang itu sendiri objek JSON lain atau array, bahwa objek batin atau array dikatakan “sarang” dalam objek luar atau array. Batas kedalaman bersarang maksimum adalah 128. Setiap upaya untuk membuat dokumen yang berisi kedalaman bersarang lebih besar dari 128 akan ditolak dengan kesalahan.

Sintaks perintah

Kebanyakan perintah memerlukan nama kunci Redis sebagai argumen pertama. Beberapa perintah juga memiliki argumen path. Argumen path default ke root jika opsional dan tidak disediakan.

Notasi:

- Argumen yang diperlukan tertutup dalam kurung sudut, mis. <key>
- Argumen opsional tertutup dalam tanda kurung siku, misalnya [path]
- Argumen opsional tambahan ditunjukkan oleh..., misalnya [json...]

Sintaks jalur

JSON-Redis mendukung dua jenis sintaks jalur:

- Sintaks yang disempurnakan - Mengikuti sintaks JSONPath yang dijelaskan oleh Goessner, seperti yang ditunjukkan pada tabel di bawah ini. Kami telah menyusun ulang dan memodifikasi deskripsi dalam tabel untuk kejelasan.
- Sintaks terbatas - Memiliki kemampuan kueri terbatas.

> **Note**
>
> Hasil dari beberapa perintah sensitif jenis sintaks path yang digunakan.

Jika jalur query dimulai dengan '$', menggunakan sintaks ditingkatkan. Jika tidak, sintaktaktaktaktaktaks terbatas digunakan.

Sintaks yang Ditingkatkan

Simbol ekspresi	Deskripsi
$	elemen root
. atau []	Operator anak
..	keturunan rekursif
*	wildcard. Semua elemen dalam sebuah objek atau array.
[]	Operator array subscript. Indeks berbasis 0.
[,]	operator serikat
[mulai: akhir: langkah]	Operator irisan array
?()	menerapkan filter (script) ekspresi ke array saat ini atau objek

Simbol ekspresi	Deskripsi
()	ekspresi filter
@	digunakan dalam ekspresi filter mengacu pada node saat sedang diproses
==	sama dengan, digunakan dalam ekspresi filter.
!=	tidak sama dengan, digunakan dalam ekspresi filter.
>	lebih besar 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.
&&	logis DAN, digunakan untuk menggabungkan beberapa ekspresi filter.
||	logis OR, digunakan untuk menggabungkan beberapa ekspresi filter.

Contoh

Contoh di bawah ini dibangun di atas contoh data XML-Goessner, yang telah kita modifikasi dengan menambahkan field 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 [*] .penulis	penulis semua buku di toko
$.. penulis	semua penulis

Jalur	Deskripsi
$.toko. *	semua anggota toko
$ ["toko"] . *	semua anggota toko
$.store.. harga	harga segala sesuatu di toko
$..*	semua anggota rekursif dari struktur JSON
$.. buku [*]	semua buku
$.. buku [0]	buku pertama
$.. buku [-1]	buku terakhir
$.. buku [0:2]	dua buku pertama
$.. buku [0,1]	dua buku pertama
$.. buku [0:4]	buku dari indeks 0 sampai 3 (indeks akhir tidak inklusif)
$.. buku [0:4:2]	buku dan indeks 0, 2
$.. buku [? (@isbn)]	semua buku dengan nomor isbn
$.. buku [? (@.harga<10)]	semua buku lebih murah dari $10
'$.. buku [? (@.harga < 10)] '	semua buku lebih murah dari $10. (Jalur harus dikutip jika berisi spasi putih)
'$.. buku [? (@ ["harga"] < 10)] '	semua buku lebih murah dari $10
'$.. buku [? (@. ["harga"] < 10)] '	semua buku lebih murah dari $10
$.. buku [? (@.harga>= 10 &&@.harga <= 100)]	semua buku di kisaran harga $10 sampai $100, inklusif

Jalur	Deskripsi
'$.. buku [? (@.harga>= 10 && @.harga<= 100)] '	semua buku di kisaran harga $10 sampai $100, inklusif. (Jalur harus dikutip jika berisi spasi putih)
$.. buku [? (@.dijual = benar ||@.dalam stok = salah)]	semua buku terjual atau habis stok
'$.. buku [? (@.sold == true || @.in-stock == false)] '	semua buku terjual atau habis stok. (Jalur harus dikutip jika berisi spasi putih)
'$.store.book [? (@. ["kategori"] == “fiksi”)] '	semua buku dalam kategori fiksi
'$.store.book [? (@. ["kategori"]! = “fiksi”)] '	semua buku dalam kategori non-fiksi

Contoh ekspresi filter lainnya:

```
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 anak
[]	Operator array subscript. Indeks berbasis 0.

Contoh

Jalur	Deskripsi
.store.book [0] .penulis	penulis buku pertama
.store.book [-1] .penulis	penulis buku terakhir
.address.city	nama kota
["toko"] ["buku"] [0] ["judul"]	judul buku pertama
["toko"] ["buku"] [-1] ["judul"]	judul buku terakhir

> **Note**
>
> Semua konten Goessner yang dikutip dalam dokumentasi ini tunduk pada Lisensi Creative Commons.

Awalan kesalahan umum

Setiap pesan kesalahan memiliki awalan. Berikut ini adalah daftar awalan kesalahan umum:

Prefiks	Deskripsi
SESAT	kesalahan umum
MEMBATASI	batas ukuran melebihi kesalahan. misalnya, batas ukuran dokumen atau batas kedalaman bersarang terlampaui
TIDAK ADA	kunci atau jalur tidak ada
OUTOFBOUNDARIES	indeks array di luar batas
SINTAKSIS	Kesalahan sintaks
TIPE-KESALAHAN	jenis nilai yang salah

Metrik terkait JSON

Metrik info JSON berikut disediakan:

Info	Deskripsi
json_total_memory_bytes	total memori yang dialokasikan ke objek JSON
json_num_documents	jumlah total dokumen di Redis

Untuk query metrik inti, jalankan perintah Redis:

```
info json_core_metrics
```

Bagaimana MemoryDB berinteraksi dengan JSON

Berikut ini menggambarkan bagaimana MemoryDB berinteraksi dengan tipe data JSON.

Operator diutamakan

Ketika mengevaluasi ekspresi kondisional untuk penyaringan, &&s diutamakan terlebih dahulu, dan kemudian ||s dievaluasi, seperti yang umum di sebagian besar bahasa. Operasi dalam tanda kurung akan dieksekusi pertama.

Perilaku batas bersarang jalur maksimum

Batas bersarang jalur maksimum MemoryDB adalah 128. Jadi nilai seperti hanya `$.a.b.c.d...` bisa mencapai 128 level.

Menangani nilai numerik

JSON tidak memiliki tipe data terpisah untuk bilangan bulat dan angka floating point. Mereka semua disebut angka.

Ketika nomor JSON diterima, itu disimpan dalam salah satu dari dua format. Jika nomor cocok menjadi 64-bit integer ditandatangani, maka dikonversi ke format itu; jika tidak, itu disimpan sebagai string. Operasi aritmatika pada dua nomor JSON (misalnya JSON.NUMINCRBY dan JSON.NUMMULTBY) berusaha mempertahankan presisi sebanyak mungkin. Jika dua operan dan nilai yang dihasilkan masuk ke dalam integer ditandatangani 64-bit, maka aritmatika integer dilakukan. Jika tidak, operan masukan diubah menjadi nomor floating point presisi ganda IEEE 64-bit, operasi aritmatika dilakukan, dan hasilnya diubah kembali menjadi string.

Perintah aritmatika dan: `NUMINCRBY` `NUMMULTBY`

- Jika kedua angka adalah bilangan bulat, dan hasilnya berada di luar jangkauan int64, maka secara otomatis akan menjadi angka floating point presisi ganda.
- Jika setidaknya salah satu angka adalah floating point, hasilnya akan menjadi angka floating point presisi ganda.
- Jika hasilnya melebihi kisaran ganda, perintah akan mengembalikan `OVERFLOW` kesalahan.

> **Note**
>
> Sebelum mesin Redis versi 6.2.6.R2 ketika nomor JSON diterima pada input, itu diubah menjadi salah satu dari dua representasi biner internal: bilangan bulat ditandatangani 64-bit atau titik mengambang presisi ganda IEEE 64-bit. String asli dan semua formatnya tidak dipertahankan. Jadi, ketika angka adalah output sebagai bagian dari respon JSON, itu dikonversi dari representasi biner internal untuk string dicetak yang menggunakan aturan format generik. Aturan-aturan ini mungkin menghasilkan string yang berbeda yang dihasilkan dari yang diterima.
>
> - Jika kedua angka adalah bilangan bulat dan hasilnya adalah di luar jangkauan`int64`, secara otomatis menjadi 64-bit IEEE presisi ganda nomor floating point.

- Jika setidaknya satu dari angka adalah floating point, hasilnya adalah 64-bit IEEE presisi ganda nomor floating point.
- Jika hasilnya melebihi kisaran 64-bit IEEE ganda, perintah mengembalikan `OVERFLOW` kesalahan.

Untuk daftar lengkap metrik yang tersedia, lihatPerintah yang didukung.

Etaks yang ketat

MemoryDB tidak mengizinkan jalur JSON dengan sintaks yang tidak valid, bahkan jika bagian dari jalur berisi jalur yang valid. Ini untuk menjaga perilaku yang benar bagi pelanggan kami.

Perintah yang didukung

Perintah Redis JSON berikut didukung:

Topik

- JSON.ARRAPPEND
- JSON.ARRINDEX
- JSON.ARRINSERT
- JSON.ARRLEN
- JSON.ARRPOP
- JSON.ARRTRIM
- JSON.CLEAR
- JSON.DEBUG
- JSON.DEL
- JSON.LUPA
- JSON.GET
- JSON.MGET
- JSON.NUMINCRBY
- JSON.NUMMULTBY
- JSON.OBJLEN
- JSON.OBJKEYS

- JSON.RESP
- JSON.SET
- JSON.STRAPPEND
- JSON.STRLEN
- JSON.TOGGLE
- JSON.JENIS

JSON.ARRAPPEND

Tambahkan satu atau lebih nilai ke nilai array di jalan.

Sintaksis

```
JSON.ARRAPPEND <key> <path> <json> [json ...]
```

- kunci (wajib) - Kunci Redis dari jenis dokumen JSON
- path (wajib) - jalur JSON
- json (required) - nilai JSON yang akan ditambahkan ke array

Kembali

Jika jalan ditingkatkan sintaks:

- Array bilangan bulat, mewakili panjang baru dari array di setiap jalan.
- Jika nilai tidak array, nilai kembali yang sesuai adalah null.
- `SYNTAXERR`kesalahan jika salah satu argumen masukan json bukan string JSON valid.
- `NONEXISTENT`kesalahan jika jalur tidak ada.

Jika jalan dibatasi sintaks:

- Integer, panjang baru array.
- Jika beberapa nilai array yang dipilih, perintah mengembalikan panjang baru dari array diperbarui terakhir.
- `WRONGTYPE`kesalahan jika nilai di jalan tidak array.
- `SYNTAXERR`kesalahan jika salah satu argumen masukan json bukan string JSON valid.

- `NONEXISTENT`kesalahan jika jalur tidak ada.

Contoh

Sintaks 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\"]]"
```

Sintaks 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

Cari kejadian pertama dari nilai JSON skalar dalam array di jalan.

- Di luar jangkauan kesalahan diperlakukan dengan pembulatan indeks ke awal dan akhir array.
- Jika start > end, kembali -1 (tidak ditemukan).

Sintaksis

```
JSON.ARRINDEX <key> <path> <json-scalar> [start [end]]
```

- kunci (wajib) - Kunci Redis dari jenis dokumen JSON
- path (wajib) - jalur JSON

- json-skalar (diperlukan) - nilai skalar untuk mencari; JSON skalar mengacu pada nilai-nilai yang bukan objek atau array. yaitu, String, number, boolean dan null adalah nilai skalar.
- mulai (opsional) - mulai indeks, inklusif. Default ke 0 jika tidak diberikan.
- akhir (opsional) - indeks akhir, eksklusif. Default 0 jika tidak disediakan, yang berarti elemen terakhir disertakan. 0 atau -1 berarti elemen terakhir disertakan.

Kembali

Jika jalan ditingkatkan sintaks:

- Array bilangan bulat. Setiap nilai adalah indeks dari elemen yang cocok dalam array di jalan. Nilainya -1 jika tidak ditemukan.
- Jika nilai tidak array, nilai kembali yang sesuai adalah null.

Jika jalan dibatasi sintaks:

- Integer, indeks pencocokan elemen, atau -1 jika tidak ditemukan.
- `WRONGTYPE`kesalahan jika nilai di jalan tidak array.

Contoh

Sintaks 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
```

Sintaks 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

Masukkan satu atau lebih nilai ke dalam nilai array di jalan sebelum indeks.

Sintaksis

```
JSON.ARRINSERT <key> <path> <index> <json> [json ...]
```

- kunci (wajib) - Kunci Redis dari jenis dokumen JSON
- path (wajib) - jalur JSON
- index (required) - indeks array sebelum nilai dimasukkan.
- json (required) - nilai JSON yang akan ditambahkan ke array

Kembali

Jika jalan ditingkatkan sintaks:

- Array bilangan bulat, mewakili panjang baru dari array di setiap jalan.
- Jika nilai adalah array kosong, nilai kembali yang sesuai adalah null.
- Jika nilai tidak array, nilai kembali yang sesuai adalah null.
- `OUTOFBOUNDARIES`kesalahan jika argumen indeks di luar batas.

Jika jalan dibatasi sintaks:

- Integer, panjang baru dari array.
- `WRONGTYPE`kesalahan jika nilai di jalan tidak array.
- `OUTOFBOUNDARIES`kesalahan jika argumen indeks di luar batas.

Contoh

Sintaks 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\"]]"
```

Sintaks 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

Dapatkan panjang nilai array di jalan.

Sintaksis

```
JSON.ARRLEN <key> [path]
```

- kunci (wajib) - Kunci Redis dari jenis dokumen JSON
- path (opsional) - jalur JSON. Default ke akar jika tidak diberikan

Kembali

Jika jalan ditingkatkan sintaks:

- Array bilangan bulat, mewakili panjang array di setiap jalan.
- Jika nilai tidak array, nilai kembali yang sesuai adalah null.
- Null jika kunci dokumen tidak ada.

Jika jalan dibatasi sintaks:

- Array string massal. Setiap elemen adalah nama kunci dalam objek.

- Integer, panjang array.
- Jika beberapa objek yang dipilih, perintah mengembalikan panjang array pertama.
- `WRONGTYPE`kesalahan jika nilai di jalan tidak array.
- `WRONGTYPE`kesalahan jika jalur tidak ada.
- Null jika kunci dokumen tidak ada.

Contoh

Sintaks 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)
```

Sintaks 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

Hapus dan kembali elemen pada indeks dari array. Muncul array kosong mengembalikan null.

Sintaksis

```
JSON.ARRPOP <key> [path [index]]
```

- kunci (wajib) - Kunci Redis dari jenis dokumen JSON
- path (opsional) - jalur JSON. Default ke akar jika tidak diberikan
- index (opsional) - posisi dalam array untuk mulai bermunculan dari.
  - Default -1 jika tidak disediakan, yang berarti elemen terakhir.
  - Nilai negatif berarti posisi dari elemen terakhir.
  - Dari indeks batas dibulatkan ke batas-batas array masing-masing.

Kembali

Jika jalan ditingkatkan sintaks:

- Array string massal, mewakili nilai-nilai muncul di setiap jalur.
- Jika nilai adalah array kosong, nilai kembali yang sesuai adalah null.
- Jika nilai tidak array, nilai kembali yang sesuai adalah null.

Jika jalan dibatasi sintaks:

- String massal, mewakili nilai JSON yang muncul
- Null jika susunan kosong.
- `WRONGTYPE`kesalahan jika nilai di jalan tidak array.

Contoh

Sintaks 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\"]]"
```

Sintaks 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

Trim array di jalan sehingga menjadi subarray [start, end], keduanya inklusif.

- Jika susunan kosong, jangan lakukan apa pun, kembalikan 0.
- Jika mulai <0, memperlakukannya sebagai 0.
- Jika akhir >= ukuran (ukuran array), memperlakukannya sebagai ukuran-1.
- Jika start >= size atau start > end, kosongkan array dan kembalikan 0.

Sintaksis

```
JSON.ARRINSERT <key> <path> <start> <end>
```

- kunci (wajib) - Kunci Redis dari jenis dokumen JSON
- path (wajib) - jalur JSON
- mulai (diperlukan) - mulai indeks, inklusif.
- akhir (diperlukan) - indeks akhir, inklusif.

Kembali

Jika jalan ditingkatkan sintaks:

- Array bilangan bulat, mewakili panjang baru dari array di setiap jalan.
- Jika nilai adalah array kosong, nilai kembali yang sesuai adalah null.
- Jika nilai tidak array, nilai kembali yang sesuai adalah null.
- `OUTOFBOUNDARIES`kesalahan jika argumen indeks di luar batas.

Jika jalan dibatasi sintaks:

- Integer, panjang baru dari array.
- Null jika susunan kosong.
- `WRONGTYPE`kesalahan jika nilai di jalan tidak array.
- `OUTOFBOUNDARIES`kesalahan jika argumen indeks di luar batas.

Contoh

Sintaks 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\"]]"
```

Sintaks 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

Kosongkan array atau objek di jalan.

Sintaksis

```
JSON.CLEAR <key> [path]
```

- kunci (wajib) - Kunci Redis dari jenis dokumen JSON
- path (opsional) - jalur JSON. Default ke akar jika tidak diberikan

Kembali

- Integer, jumlah kontainer dibersihkan.
- Kliring array kosong atau objek account untuk 0 kontainer dibersihkan.

> **Note**
>
> Piror ke Redis versi 6.2.6.R2, membersihkan array kosong atau objek account untuk 1 kontainer dibersihkan.

- Menghapus nilai non-kontainer mengembalikan 0.
- Jika tidak ada array atau objek nilai terletak oleh jalan, perintah 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) 6
127.0.0.1:6379> JSON.CLEAR k1  $[*]
(integer) 0
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

Laporkan informasi. Subperintah yang didukung adalah:

- MEMORY <key>[path] - laporkan penggunaan memori dalam byte nilai JSON. Jalur default ke root jika tidak disediakan.
- KEDALAMAN <key>[jalur] - Melaporkan kedalaman jalur maksimum dokumen JSON.

> **Note**
>
> Subperintah ini hanya tersedia menggunakan mesin Redis versi 6.2.6.R2 atau yang lebih baru.

- BIDANG <key>[jalur] - laporkan jumlah bidang di jalur dokumen yang ditentukan. Jalur 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 yang mengandung. Setiap nilai kontainer, kecuali wadah root, dihitung sebagai satu bidang tambahan.
- BANTUAN — cetak pesan bantuan dari perintah.

Sintaksis

```
JSON.DEBUG <subcommand & arguments>
```

Tergantung pada subkomando:

MEMORI

- Jika jalan ditingkatkan sintaks:
  - mengembalikan array bilangan bulat, mewakili ukuran memori (dalam byte) dari nilai JSON di setiap jalur.
  - mengembalikan array kosong jika kunci Redis tidak ada.
- Jika jalan dibatasi sintaks:
  - mengembalikan integer, ukuran memori nilai JSON dalam byte.
  - mengembalikan null jika kunci Redis tidak ada.

KEDALAMAN

- Mengembalikan integer yang mewakili kedalaman jalur maksimum dokumen JSON.
- Mengembalikan null jika kunci Redis tidak ada.

LADANG

- Jika jalan ditingkatkan sintaks:
  - mengembalikan array bilangan bulat, yang mewakili jumlah bidang nilai JSON di setiap jalur.
  - mengembalikan array kosong jika kunci Redis tidak ada.
- Jika jalan dibatasi sintaks:
  - mengembalikan integer, jumlah bidang dari nilai JSON.
  - mengembalikan null jika kunci Redis tidak ada.

BANTUAN — mengembalikan larik pesan bantuan.

Contoh

Sintaks 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
```

Sintaks 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

Hapus nilai JSON di jalur dalam kunci dokumen. Jika path adalah root, itu setara dengan menghapus kunci dari Redis.

Sintaksis

```
JSON.DEL <key> [path]
```

- kunci (wajib) - Kunci Redis dari jenis dokumen JSON
- path (opsional) - jalur JSON. Default ke akar jika tidak diberikan

Kembali

- Jumlah elemen dihapus.
- 0 jika kunci Redis tidak ada.
- 0 jika jalur JSON tidak valid atau tidak ada.

Contoh

Sintaks 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\":[]}"
```

Sintaks 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.LUPA

Alias dari JSON.DEL

JSON.GET

Kembalikan JSON serial pada satu atau beberapa jalur.

Sintaksis

```
JSON.GET <key>
[INDENT indentation-string]
[NEWLINE newline-string]
[SPACE space-string]
[NOESCAPE]
[path ...]
```

- kunci (wajib) - Kunci Redis dari jenis dokumen JSON
- INDENT/NEWLINE/SPACE (opsional) - mengontrol format string JSON yang dikembalikan, yaitu, “pretty print”. Nilai default masing-masing adalah string kosong. Mereka bisa berlebihan dalam kombinasi apapun. Mereka dapat ditentukan dalam urutan apa pun.
- NOESCAPE - opsional, diizinkan untuk hadir untuk kompatibilitas warisan dan tidak memiliki efek lain.
- path (opsional) - nol atau lebih jalur JSON, default ke root jika tidak ada yang diberikan. Jalur argumen harus ditempatkan pada akhir.

Kembali

Sintaks jalur yang ditingkatkan:

Jika satu jalur diberikan:

- Mengembalikan string serial dari larik nilai.
- Jika tidak ada nilai yang dipilih, perintah mengembalikan array kosong.

Jika beberapa jalur diberikan:

- Kembalikan objek JSON stringified, di mana setiap jalur adalah kunci.
- Jika ada sintaks jalur ditingkatkan dan dibatasi campuran, hasilnya sesuai dengan sintaks ditingkatkan.
- Jika jalan tidak ada, nilai yang sesuai adalah array kosong.

Contoh

Sintaks 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]"
```

Sintaks 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

Dapatkan JSONs serial di jalur dari beberapa kunci dokumen. Kembali null untuk kunci yang tidak ada atau jalur JSON.

Sintaksis

```
JSON.MGET <key> [key ...] <path>
```

- kunci (wajib) - Satu atau lebih kunci Redis dari jenis dokumen.
- path (wajib) - jalur JSON

Kembali

- Array Massal Strings. Ukuran array sama dengan jumlah kunci dalam perintah. Setiap elemen dari array diisi dengan baik (a) JSON serial seperti yang terletak oleh jalan atau (b) Null jika kunci tidak ada atau jalan tidak ada dalam dokumen atau jalan tidak valid (sintaks error).
- Jika salah satu kunci yang ditentukan ada dan bukan kunci JSON, perintah mengembalikan `WRONGTYPE` kesalahan.

Contoh

Sintaks 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\"]"
```

Sintaks 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

Kenaikan nilai angka di jalan dengan nomor tertentu.

Sintaksis

```
JSON.NUMINCRBY <key> <path> <number>
```

- kunci (wajib) - Kunci Redis dari jenis dokumen JSON
- path (wajib) - jalur JSON
- nomor (wajib) — nomor

Kembali

Jika jalan ditingkatkan sintaks:

- Array Strings massal mewakili nilai yang dihasilkan di setiap jalur.
- Jika nilai tidak angka, nilai kembali yang sesuai adalah null.
- `WRONGTYPE`kesalahan jika nomor tidak dapat diurai.
- `OVERFLOW`kesalahan jika hasilnya berada di luar kisaran 64-bit IEEE ganda.
- `NONEXISTENT`jika kunci dokumen tidak ada.

Jika jalan dibatasi sintaks:

- Massal String mewakili nilai yang dihasilkan.
- Jika beberapa nilai dipilih, perintah mengembalikan hasil dari nilai terbaru terakhir.
- `WRONGTYPE`kesalahan jika nilai di jalan bukan angka.
- `WRONGTYPE`kesalahan jika nomor tidak dapat diurai.
- `OVERFLOW`kesalahan jika hasilnya berada di luar kisaran 64-bit IEEE ganda.
- `NONEXISTENT`jika kunci dokumen tidak ada.

Contoh

Sintaks 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}}"
```

Sintaks 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

Kalikan nilai angka di jalur dengan nomor tertentu.

Sintaksis

```
JSON.NUMMULTBY <key> <path> <number>
```

- kunci (wajib) - Kunci Redis dari jenis dokumen JSON
- path (wajib) - jalur JSON
- nomor (wajib) — nomor

Kembali

Jika jalan ditingkatkan sintaks:

- Array Strings massal mewakili nilai yang dihasilkan di setiap jalur.
- Jika nilai tidak angka, nilai kembali yang sesuai adalah null.
- `WRONGTYPE`kesalahan jika nomor tidak dapat diurai.
- `OVERFLOW`kesalahan jika hasilnya berada di luar kisaran 64-bit IEEE ganda.
- `NONEXISTENT`jika kunci dokumen tidak ada.

Jika jalan dibatasi sintaks:

- Massal String mewakili nilai yang dihasilkan.
- Jika beberapa nilai dipilih, perintah mengembalikan hasil dari nilai terbaru terakhir.
- `WRONGTYPE`kesalahan jika nilai di jalan bukan angka.

- `WRONGTYPE`kesalahan jika nomor tidak dapat diurai.
- `OVERFLOW`kesalahan jika hasilnya berada di luar kisaran 64-bit IEEE ganda.
- `NONEXISTENT`jika kunci dokumen tidak ada.

Contoh

Sintaks 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]"
```

Sintaks 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

Dapatkan jumlah kunci dalam nilai objek di jalan.

Sintaksis

```
JSON.OBJLEN <key> [path]
```

- kunci (wajib) - Kunci Redis dari jenis dokumen JSON
- path (opsional) - jalur JSON. Default ke akar jika tidak diberikan

Kembali

Jika jalan ditingkatkan sintaks:

- Array bilangan bulat, mewakili panjang objek di setiap jalur.
- Jika nilai tidak objek, nilai kembali yang sesuai adalah null.
- Null jika kunci dokumen tidak ada.

Jika jalan dibatasi sintaks:

- Integer, jumlah kunci dalam objek.
- Jika beberapa objek yang dipilih, perintah mengembalikan panjang objek pertama.
- `WRONGTYPE`kesalahan jika nilai di jalan bukan objek.
- `WRONGTYPE`kesalahan jika jalur tidak ada.
- Null jika kunci dokumen tidak ada.

Contoh

Sintaks 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)
```

Sintaks 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

Dapatkan nama kunci dalam nilai objek di jalan.

Sintaksis

```
JSON.OBJKEYS <key> [path]
```

- kunci (wajib) - Kunci Redis dari jenis dokumen JSON
- path (opsional) - jalur JSON. Default ke akar jika tidak diberikan

Kembali

Jika jalan ditingkatkan sintaks:

- Array array string massal. Setiap elemen adalah array kunci dalam objek yang cocok.
- Jika nilai tidak objek, nilai kembali yang sesuai adalah nilai kosong.
- Null jika kunci dokumen tidak ada.

Jika jalan dibatasi sintaks:

- Array string massal. Setiap elemen adalah nama kunci dalam objek.
- Jika beberapa objek yang dipilih, perintah mengembalikan kunci dari objek pertama.
- `WRONGTYPE`kesalahan jika nilai di jalan bukan objek.
- `WRONGTYPE`kesalahan jika jalur tidak ada.
- Null jika kunci dokumen tidak ada.

Contoh

Sintaks 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"
```

Sintaks 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

Kembalikan nilai JSON di jalur yang diberikan di Redis Serialization Protocol (RESP). Jika nilainya adalah wadah, responnya adalah array RESP atau array bersarang.

- JSON null dipetakan ke RESP Null Massal String.
- nilai-nilai boolean JSON dipetakan ke masing-masing RESP Strings Sederhana.
- Bilangan integer dipetakan ke RESP Integer.
- Angka floating point ganda IEEE 64-bit dipetakan ke RESP Massal Strings.
- JSON String dipetakan ke RESP Massal Strings.
- JSON Array direpresentasikan sebagai RESP Array, di mana elemen pertama adalah string sederhana [, diikuti oleh elemen array.
- JSON Objects direpresentasikan sebagai RESP Array, di mana elemen pertama adalah string sederhana {, diikuti oleh pasangan kunci-nilai, yang masing-masing adalah string massal RESP.

Sintaksis

```
JSON.RESP <key> [path]
```

- kunci (wajib) - Kunci Redis dari jenis dokumen JSON
- path (opsional) - jalur JSON. Default ke akar jika tidak diberikan

Kembali

Jika jalan ditingkatkan sintaks:

- Array array. Setiap elemen array mewakili bentuk RESP dari nilai pada satu jalur.
- Array kosong jika kunci dokumen tidak ada.

Jika jalan dibatasi sintaks:

- Array, mewakili bentuk RESP dari nilai di jalan.
- Null jika kunci dokumen tidak ada.

Contoh

Sintaks 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"
```

Sintaks 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

Mengatur nilai JSON di jalan.

Jika jalan panggilan untuk anggota objek:

- Jika elemen induk tidak ada, perintah akan kembali kesalahan tidak ada.
- Jika elemen induk ada tetapi bukan objek, perintah akan kembali 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 anak terakhir di jalan. Jika tidak, perintah akan mengembalikan tidak ada kesalahan yang tidak ada.
  - Jika anggota ada, nilainya akan digantikan oleh nilai JSON.

Jika jalan panggilan untuk indeks array:

- Jika elemen induk tidak ada, perintah akan mengembalikan kesalahan tidak ada.
- Jika elemen induk ada tetapi tidak array, perintah akan kembali ERROR.
- Jika elemen induk ada tetapi indeks di luar batas, perintah akan kembali kesalahan OUTOFBOUNDS.
- Jika elemen induk ada dan indeks valid, elemen akan digantikan oleh nilai JSON baru.

Jika jalur panggilan untuk objek atau array, nilai (objek atau array) akan digantikan oleh nilai JSON baru.

Sintaksis

```
JSON.SET <key> <path> <json> [NX | XX]
```

[NX | XX] Di mana Anda dapat memiliki 0 atau 1 dari [NX | XX] pengidentifikasi

- kunci (wajib) - Kunci Redis dari jenis dokumen JSON
- path (wajib) - jalur JSON. Untuk kunci Redis baru, jalur JSON harus menjadi root “.”.
- NX (opsional) - Jika jalur adalah root, tetapkan nilainya hanya jika kunci Redis tidak ada, yaitu, masukkan dokumen baru. Jika path tidak root, mengatur nilai hanya jika jalan tidak ada, yaitu, memasukkan nilai ke dalam dokumen.

- XX (opsional) - Jika jalurnya adalah root, atur nilainya hanya jika kunci Redis ada, yaitu ganti dokumen yang ada. Jika path tidak root, menetapkan nilai hanya jika jalan ada, yaitu, memperbarui nilai yang ada.

Kembali

- String sederhana 'OK' pada kesuksesan.
- Null jika kondisi NX atau XX tidak terpenuhi.

Contoh

Sintaks 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]}"
```

Jalur sintaks 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 jalan.

Sintaksis

```
JSON.STRAPPEND <key> [path] <json_string>
```

- kunci (wajib) - Kunci Redis dari jenis dokumen JSON
- path (opsional) - jalur JSON. Default ke akar jika tidak diberikan
- json_string (wajib) - JSON representasi string. Perhatikan bahwa string JSON harus dikutip, yaitu, '"foo"'.

Kembali

Jika jalan ditingkatkan sintaks:

- Array bilangan bulat, mewakili panjang baru dari string di setiap jalan.
- Jika nilai di jalan tidak string, nilai kembali yang sesuai adalah null.
- `SYNTAXERR` kesalahan jika argumen masukan json bukan string JSON valid.
- `NONEXISTENT` kesalahan jika jalur tidak ada.

Jika jalan dibatasi sintaks:

- Integer, panjang baru string.
- Jika beberapa nilai string yang dipilih, perintah mengembalikan panjang baru dari string diperbarui terakhir.
- `WRONGTYPE`error jika nilai di jalan bukan string.
- `WRONGTYPE`kesalahan jika argumen masukan json bukan string JSON valid.
- `NONEXISTENT`kesalahan jika jalur tidak ada.

Contoh

Sintaks 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)
```

Jalur sintaks 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

Dapatkan panjang nilai string JSON di jalur.

Sintaksis

```
JSON.STRLEN <key> [path]
```

- kunci (wajib) - Kunci Redis dari jenis dokumen JSON
- path (opsional) - jalur JSON. Default ke akar jika tidak diberikan

Kembali

Jika jalan ditingkatkan sintaks:

- Array bilangan bulat, mewakili panjang nilai string di setiap jalur.
- Jika nilai tidak string, nilai kembali yang sesuai adalah null.
- Null jika kunci dokumen tidak ada.

Jika jalan dibatasi sintaks:

- Integer, panjang string.
- Jika beberapa nilai string yang dipilih, perintah mengembalikan panjang string pertama ini.
- `WRONGTYPE`error jika nilai di jalan bukan string.
- `NONEXISTENT`kesalahan jika jalur tidak ada.
- Null jika kunci dokumen tidak ada.

Contoh

Sintaks 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)
```

Jalur sintaks 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

Beralih nilai boolean antara true dan false di path.

Sintaksis

```
JSON.TOGGLE <key> [path]
```

- kunci (wajib) - Kunci Redis dari jenis dokumen JSON
- path (opsional) - jalur JSON. Default ke akar jika tidak diberikan

Kembali

Jika jalan ditingkatkan sintaks:

- Array bilangan bulat (0 - false, 1 - true) mewakili nilai boolean yang dihasilkan di setiap jalur.
- Jika nilai adalah tidak boolean, nilai kembali yang sesuai adalah null.
- `NONEXISTENT`jika kunci dokumen tidak ada.

Jika jalan dibatasi sintaks:

- String (“true” /"false”) mewakili nilai boolean yang dihasilkan.
- `NONEXISTENT`jika kunci dokumen tidak ada.
- `WRONGTYPE`error jika nilai di jalan bukan boolean.

Contoh

Sintaks 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)
```

Jalur sintaks 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.JENIS

Laporkan jenis nilai di jalur yang diberikan.

Sintaksis

```
JSON.TYPE <key> [path]
```

- kunci (wajib) - Kunci Redis dari jenis dokumen JSON
- path (opsional) - jalur JSON. Default ke akar jika tidak diberikan

Kembali

Jika jalan ditingkatkan sintaks:

- Array string, mewakili jenis nilai di setiap jalan. Jenisnya adalah salah satu dari {"null”, “boolean”, “string”, “number”, “integer”, “object” dan “array"}.
- Jika jalan tidak ada, nilai kembali yang sesuai adalah null.
- Array kosong jika kunci dokumen tidak ada.

Jika jalan dibatasi sintaks:

- String, jenis nilai
- Null jika kunci dokumen tidak ada.

- Null jika jalur JSON tidak valid atau tidak ada.

Contoh

Sintaks 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
```

Jalur sintaks 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 MemoryDB Anda

Untuk membantu Anda mengelola klaster dan sumber daya MemoryDB lainnya, Anda dapat menetapkan metadata Anda sendiri ke setiap sumber daya dalam bentuk tanda. Tanda memungkinkan Anda untuk mengategorikan sumber daya AWS Anda dalam berbagai cara, misalnya, berdasarkan tujuan, pemilik, atau lingkungan. Hal ini berguna jika Anda memiliki banyak sumber daya dengan jenis yang sama—Anda dapat dengan cepat mengidentifikasi sumber daya tertentu berdasarkan tag yang telah Anda tetapkan. Topik ini menjelaskan tanda dan menunjukkan kepada Anda cara membuatnya.

> **Warning**
>
> Sebagai praktik terbaik, sebaiknya Anda tidak menyertakan data sensitif ke dalam tanda Anda.

Dasar tanda

Tanda adalah sebuah label yang Anda tetapkan ke sebuah sumber daya AWS. Setiap tanda terdiri dari sebuah kunci dan sebuah nilai opsional, yang keduanya Anda tentukan. Tanda memungkinkan Anda untuk mengategorikan sumber daya AWS Anda dengan berbagai cara, misalnya, berdasarkan tujuan atau pemilik. Misalnya, Anda dapat menentukan serangkaian tag untuk klaster MemoryDB akun yang membantu Anda melacak pemilik dan grup pengguna dari setiap klaster.

Sebaiknya Anda merancang seperangkat kunci tanda yang memenuhi kebutuhan Anda untuk setiap jenis sumber daya. Penggunaan seperangkat kunci tanda yang konsisten akan mempermudah Anda dalam mengelola sumber daya Anda. Anda dapat mencari dan menyaring sumber daya berdasarkan tanda yang Anda tambahkan. Untuk informasi selengkapnya tentang cara implementasi strategi penandaan sumber daya yang efektif, lihat laporan resmi AWSPraktik Terbaik Penandaan.

Tanda tidak memiliki makna semantik pada MemoryDB dan diterjemahkan sebagai serangkaian karakter saja. Selain itu, tanda tidak secara otomatis ditetapkan ke sumber daya Anda. Anda dapat mengedit kunci dan nilai tanda, dan Anda dapat menghapus tanda dari sumber daya pada saat kapan pun. Anda dapat menetapkan nilai tanda menjadi `null`. Jika Anda menambahkan tanda yang memiliki kunci yang sama dengan tanda yang telah ada pada sumber daya tersebut, nilai yang baru akan menimpa nilai yang lama. Jika Anda menghapus sumber daya, semua tanda untuk sumber daya tersebut juga dihapus.

Anda dapat bekerja dengan menggunakan tandaAWS Management Console, yangAWS CLI, dan MemoryDB API.

Jika Anda menggunakan IAM, Anda dapat mengontrol pengguna mana diAWSmemiliki 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 MemoryDB yang telah ada di 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 untuk menentukan tanda untuk sebuah sumber daya saat Anda membuat sumber daya itu; misalnya, tanda dengan kunci dari Nama dan nilai yang Anda tentukan. Dalam kebanyakan kasus, konsol menerapkan tanda segera setelah sumber daya dibuat (bukan selama pembuatan sumber daya). Konsol dapat mengatur sumber daya sesuai denganNamatanda, tetapi tanda ini tidak memiliki makna semantik pada layanan MemoryDB.

Selain itu, beberapa tindakan pembuatan sumber daya memungkinkan Anda untuk menentukan tanda untuk sumber daya saat sumber daya itu dibuat. Jika tanda tidak dapat diterapkan selama pembuatan sumber daya, maka proses pembuatan sumber daya akan dirollback. Hal ini untuk memastikan bahwa sumber daya dibuat dengan tanda atau tidak dibuat sama sekali, dan tidak akan ada sumber daya yang dibiarkan tidak diberi tanda pada waktu kapan pun. Dengan menandai sumber daya pada saat pembuatan, Anda dapat menghilangkan kebutuhan untuk menjalankan skrip penandaan khusus setelah pembuatan sumber daya.

Jika Anda menggunakan API Amazon MemoryDB,AWSCLI, atauAWSSDK, Anda dapat menggunakan`Tags`parameter pada tindakan MemoryDB API yang relevan untuk menerapkan tag. File tersebut adalah:

- `CreateCluster`
- `CopySnapshot`
- `CreateParameterGroup`
- `CreateSubnetGroup`
- `CreateSnapshot`
- `CreateACL`
- `CreateUser`

Tabel berikut menjelaskan sumber daya MemoryDB yang dapat ditandai, dan sumber daya yang dapat ditandai saat dibuat menggunakan API MemoryDB,AWSCLI, atauAWSSDK.

Dukungan penandaan untuk sumber daya MemoryDB

Mendukung tanda	Mendukung penandaan saat pembuatan
Ya	Ya
Ya	Ya
Ya	Ya
Ya	Ya
Ya	Ya
Ya	Ya

Anda dapat menerapkan izin tingkat sumber daya berbasis tag dalam kebijakan IAM Anda untuk tindakan MemoryDB API yang mendukung pemberian tag saat pembuatan untuk mengimplementasikan kontrol terperinci atas pengguna dan grup yang dapat memberi tag pada sumber daya saat pembuatan. Sumber daya Anda diamankan dari pembuatan dengan tepat—tanda yang diterapkan segera pada sumber daya Anda. Oleh karena itu, izin tingkat sumber daya berbasis tanda apa pun yang mengontrol penggunaan sumber daya akan efektif segera. Sumber daya Anda dapat dilacak dan dilaporkan dengan lebih akurat. Anda dapat menerapkan penggunaan penandaan pada sumber daya baru, dan mengontrol kunci dan nilai tanda mana yang ditetapkan pada sumber daya Anda.

Untuk informasi selengkapnya, lihat Contoh menandai sumber daya.

Untuk informasi selanjutnya tentang penandaan sumber daya Anda untuk penagihan, lihat Memantau biaya dengan tanda alokasi biaya.

Menandai klaster dan snapshot

Aturan berikut berlaku untuk penandaan sebagai bagian dari operasi permintaan:

- CreateCluster :
  - Jika `--cluster-name` disediakan:

    Jika tanda disertakan dalam permintaan, klaster akan ditandai.
  - Jika `--snapshot-name` disediakan:

    Jika tanda disertakan dalam permintaan, klaster akan ditandai hanya dengan tanda tersebut. Jika tidak ada tanda yang disertakan dalam permintaan, tanda snapshot akan ditambahkan ke klaster.
- CreateSnapshot :
  - Jika `--cluster-name` disediakan:

    Jika tanda disertakan dalam permintaan, hanya tanda permintaan yang akan ditambahkan ke snapshot. Jika tidak ada tanda yang disertakan dalam permintaan, tanda klaster akan ditambahkan ke snapshot.
  - Untuk snapshot otomatis:

    Tanda akan disebarkan dari tanda klaster.
- 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.
- TagResourcedanUntagResource:

  Tag akan ditambahkan/dihapus dari sumber daya.

Pembatasan 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 MemoryDB memungkinkan karakter apa pun di dalam tandanya, layanan lain mungkin lebih membatasi. Karakter yang diizinkan di semua layanan adalah huruf, angka, dan spasi yang dapat diwakili dalam UTF-8, serta karakter berikut: + - = . _ : / @
- Kunci dan nilai tanda peka huruf besar dan kecil.
- Prefiks `aws:` disimpan untuk penggunaan AWS. Jika sebuah tanda memiliki sebuah kunci tanda dengan prefiks ini, maka Anda tidak dapat mengedit atau menghapus kunci atau nilai tanda tersebut. Tanda dengan prefiks `aws:` tidak mengurangi batas tanda per batas sumber daya Anda.

Anda tidak dapat mengakhiri, menghentikan, atau menghapus sumber daya hanya berdasarkan tandanya; Anda harus menentukan pengidentifikasi sumber daya tersebut. Misalnya, untuk menghapus snapshot yang Anda tandai dengan tanda kunci yang disebut `DeleteMe`, maka Anda harus menggunakan tindakan `DeleteSnapshot` dengan pengidentifikasi sumber daya dari snapshot itu, seperti `snap-1234567890abcdef0`.

Untuk informasi lebih lanjut tentang sumber daya MemoryDB yang dapat Anda tandai, lihatSumber daya yang dapat Anda tandai.

Contoh menandai sumber daya

- Menambahkan tanda ke klaster.

```
aws memorydb tag-resource \
--resource-arn arn:aws:memorydb:us-east-1:111111222233:cluster/my-cluster \
--tags Key="project",Value="XYZ" Key="memorydb",Value="Service"
```

- Membuat klaster menggunakan tanda.

```
aws memorydb create-cluster \
--cluster-name testing-tags \
--description cluster-test \
--subnet-group-name test \
--node-type db.r6g.large \
--acl-name open-access \
--tags Key="project",Value="XYZ" Key="memorydb",Value="Service"
```

- Membuat Snapshot dengan tanda.

  Untuk kasus ini, jika Anda menambahkan tanda pada permintaan, meskipun klaster berisi tanda, snapshot hanya akan menerima tanda permintaan.

```
aws memorydb create-snapshot \
--cluster-name testing-tags \
--snapshot-name bkp-testing-tags-mycluster \
--tags Key="work",Value="foo"
```

Memantau biaya dengan tanda alokasi biaya

Ketika Anda menambahkan tanda alokasi biaya ke sumber daya Anda di MemoryDB untuk Redis, Anda dapat melacak biaya dengan mengelompokkan pengeluaran pada faktur Anda berdasarkan nilai tanda sumber daya.

Tanda alokasi biaya MemoryDB adalah pasangan kunci dan nilai yang Anda tentukan dan kaitkan dengan sumber daya MemoryDB. Kunci dan nilainya peka terhadap huruf besar dan kecil. Anda dapat menggunakan kunci tanda untuk menentukan kategori, dan nilai tanda dapat berupa item dalam kategori tersebut. Sebagai contoh, Anda dapat menentukan kunci tanda `CostCenter` dan nilai tanda `10010`, yang menunjukkan bahwa sumber daya itu ditetapkan ke pusat pembiayaan 10010. Anda juga dapat menggunakan tanda untuk menunjuk sumber daya sebagai tanda yang digunakan untuk pengujian atau produksi dengan menggunakan kunci seperti `Environment` dan nilai seperti `test` atau `production`. Sebaiknya Anda menggunakan seperangkat kunci tanda yang konsisten untuk mempermudah pelacakan biaya terkait dengan sumber daya Anda.

Gunakan tanda alokasi biaya untuk mengorganisasi tagihan AWS Anda guna mencerminkan struktur biaya Anda sendiri. Untuk melakukannya, daftar untuk mendapatkan tagihan akun AWS Anda dengan menyertakan nilai kunci tag. Lalu, untuk melihat biaya sumber daya gabungan, kelola informasi penagihan Anda sesuai dengan sumber daya Anda dengan nilai kunci tag yang sama. Misalnya, Anda dapat menandai beberapa sumber daya dengan nama aplikasi tertentu, kemudian organisasikan informasi penagihan Anda 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`. Pada salah satu sumber daya Anda mungkin memiliki nilai `MemoryDB` dan `Asia Pacific (Singapore)`, dan pada sumber daya lain Anda mempunyai nilai `MemoryDB` dan `Europe (Frankfurt)`. Anda kemudian dapat melihat total biaya MemoryDB Anda yang dibagi menurut wilayah. Untuk informasi selengkapnya, lihat Gunakan Tag Alokasi Biaya dalam AWS BillingPanduan Pengguna.

Anda dapat menambahkan tanda alokasi biaya MemoryDB. Saat Anda menambah, menampilkan daftar, mengubah, menyalin, atau menghapus tanda, operasi itu diterapkan hanya pada klaster yang ditentukan.

Karakteristik tag alokasi biaya MemoryDB

- Tanda alokasi biaya diterapkan ke sumber daya MemoryDB yang ditentukan dalam operasi CLI dan API sebagai ARN. Jenis sumber daya akan menjadi "klaster".

  Format ARN:`arn:aws:memorydb:`*`<region>`*`:`*`<customer-id>`*`:`*`<resource-type>`*`/`*`<resource-name>`*

  ARN Sampel: `arn:aws:memorydb:us-east-1:1234567890:cluster/my-cluster`
- Kunci tanda adalah nama yang diperlukan untuk tanda. Nilai string dari tanda 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 dua (:), garis miring terbalik (\), tanda sama dengan (=), tanda plus (+), tanda hubung (-), atau tanda a keong (@).
- Nilai tanda adalah nilai opsional dari tanda. Nilai string dari nilai tanda 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 dua (:), garis miring terbalik (\), tanda sama dengan (=), tanda plus (+), tanda hubung (-), atau tanda a keong (@).
- Sumber daya MemoryDB dapat memiliki maksimum tanda 50.
- Nilai tidak harus unik di dalam kumpulan tanda. Misalnya, Anda dapat memiliki kumpulan tanda di mana kunci `Service` dan `Application` keduanya memiliki nilai `MemoryDB`.

AWS tidak menerapkan makna semantik apa pun pada tanda Anda. Tanda ditafsirkan dengan ketat sebagai string karakter.AWStidak secara otomatis mengatur tag apapun pada sumber daya MemoryDB.

Mengelola tanda alokasi biaya Anda menggunakan AWS CLI

Anda dapat menggunakan AWS CLI untuk menambah, mengubah, atau menghapus tanda alokasi biaya.

arn Sampel: `arn:aws:memorydb:us-east-1:1234567890: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 menggunakanAWS CLIuntuk daftar tag pada sumber daya MemoryDB yang ada dengan menggunakandaftar-tandaoperasi.

Kode berikut menggunakanAWS CLIuntuk daftar tag pada cluster MemoryDB`my-clusterdi` wilayah us-east-1.

Untuk Linux, macOS, atau Unix:

```
aws memorydb list-tags \
  --resource-arn arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster
```

Untuk Windows:

```
aws memorydb list-tags ^
  --resource-arn arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster
```

Keluaran 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 kosongTagList.

```
{
```

```
   "TagList": []
}
```

Untuk informasi lebih lanjut, lihatAWS CLIuntuk MemoryDBdaftar-tanda.

Menambah tanda menggunakan AWS CLI

Anda dapat menggunakanAWS CLIuntuk menambahkan tanda ke sumber daya MemoryDB yang ada dengan menggunakantag-resourceOperasi CLI. Jika kunci tanda tidak ada pada sumber daya, maka kunci dan nilai ditambahkan ke sumber daya itu. Jika kunci sudah ada pada sumber daya, maka nilai yang terkait dengan kunci itu diperbarui ke nilai yang baru.

Kode berikut menggunakanAWS CLIuntuk menambahkan kunci`Service`dan`Region`dengan nilai-nilai`memorydb`dan`us-east-1`masing-masing ke cluster`my-cluster`di wilayah us-east-1.

Untuk Linux, macOS, atau Unix:

```
aws memorydb tag-resource \
 --resource-arn arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster \
 --tags Key=Service,Value=memorydb \
        Key=Region,Value=us-east-1
```

Untuk Windows:

```
aws memorydb tag-resource ^
 --resource-arn arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster ^
 --tags Key=Service,Value=memorydb ^
        Key=Region,Value=us-east-1
```

Keluaran dari operasi ini akan terlihat seperti berikut ini, daftar dari semua tanda pada sumber daya mengikuti operasi.

```
{
   "TagList": [
      {
         "Value": "memorydb",
         "Key": "Service"
      },
      {
         "Value": "us-east-1",
```

```
        "Key": "Region"
      }
   ]
}
```

Untuk informasi lebih lanjut, lihatAWS CLIuntuk MemoryDBtag-resource.

Anda juga dapat menggunakanAWS CLIuntuk menambahkan tanda ke klaster ketika Anda membuat klaster baru dengan menggunakan operasicreate-cluster.

Mengubah tanda menggunakan AWS CLI

Anda dapat menggunakanAWS CLIuntuk memodifikasi tag pada cluster MemoryDB.

Untuk mengubah tanda:

- Gunakantag-resourceuntuk menambahkan tanda dan nilai baru atau untuk mengubah nilai yang terkait dengan tanda yang ada.
- Gunakanuntag-resourceuntuk menghapus tag tertentu dari sumber daya.

Keluaran dari kedua operasi akan berupa daftar tanda dan nilai-nilainya pada klaster yang ditentukan.

Menghapus tanda menggunakan AWS CLI

Anda dapat menggunakanAWS CLIuntuk menghapus tanda dari klaster MemoryDB dengan menggunakanuntag-resourceoperasi.

Kode berikut menggunakanAWS CLIuntuk menghapus tag dengan tombol`Service`dan`Region`dari clustermy-clusterdi wilayah us-east-1.

Untuk Linux, macOS, atau Unix:

```
aws memorydb untag-resource \
 --resource-arn arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster \
 --tag-keys Region Service
```

Untuk Windows:

```
aws memorydb untag-resource ^
```

```
  --resource-arn arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster ^
  --tag-keys Region Service
```

Keluaran dari operasi ini akan terlihat seperti berikut ini, daftar dari semua tanda pada sumber daya mengikuti operasi.

```
{
   "TagList": []
}
```

Untuk informasi lebih lanjut, lihatAWS CLIuntuk MemoryDBuntag-resource.

Mengelola tanda alokasi biaya Anda menggunakan API MemoryDB

Anda dapat menggunakan API MemoryDB untuk menambahkan, mengubah, atau menghapus tanda alokasi biaya.

Tanda alokasi biaya diterapkan ke MemoryDB untuk klaster. Klaster yang akan ditandai ditentukan menggunakan ARN (Amazon Resource Name).

arn Sampel: `arn:aws:memorydb:us-east-1:1234567890:cluster/my-cluster`

Topik

- Menandai menggunakan API MemoryDB
- Menambahkan tanda menggunakan API MemoryDB
- Memodifikasi tanda menggunakan API MemoryDB
- Menghapus tanda menggunakan API MemoryDB

Menandai menggunakan API MemoryDB

Anda dapat menggunakan API MemoryDB untuk daftar tanda pada sumber daya yang ada dengan menggunakanListTagsoperasi.

Kode berikut menggunakan MemoryDB API untuk daftar tag pada sumber dayamy-clusterdi wilayah us-east-1.

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=ListTags
```

```
   &ResourceArn=arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Version=2021-01-01
   &Timestamp=20210802T192317Z
   &X-Amz-Credential=<credential>
```

Menambahkan tanda menggunakan API MemoryDB

Anda dapat menggunakan API MemoryDB untuk menambahkan tanda ke klaster MemoryDB yang ada dengan menggunakanTagResourceoperasi. Jika kunci tanda tidak ada pada sumber daya, maka kunci dan nilai ditambahkan ke sumber daya itu. Jika kunci sudah ada pada sumber daya, maka nilai yang terkait dengan kunci itu diperbarui ke nilai yang baru.

Kode berikut menggunakan API MemoryDB untuk menambahkan kunci`Service`dan`Region`dengan nilai-nilai`memorydb`dan`us-east-1`masing-masing ke sumber daya`my-cluster`di wilayah us-east-1.

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=TagResource
   &ResourceArn=arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Tags.member.1.Key=Service
   &Tags.member.1.Value=memorydb
   &Tags.member.2.Key=Region
   &Tags.member.2.Value=us-east-1
   &Version=2021-01-01
   &Timestamp=20210802T192317Z
   &X-Amz-Credential=<credential>
```

Untuk informasi selengkapnya, lihatTagResource.

Memodifikasi tanda menggunakan API MemoryDB

Anda dapat menggunakan API MemoryDB untuk mengubah tanda pada klaster MemoryDB.

Untuk mengubah nilai tanda:

- Gunakan operasi TagResource baik untuk menambahkan tanda dan nilai baru ataupun untuk mengubah nilai tanda yang telah ada.

- Gunakan UntagResource untuk menghapus tanda dari sumber daya.

Keluaran dari kedua operasi akan berupa daftar tanda dan nilai-nilainya pada sumber daya yang ditentukan tersebut.

Menghapus tanda menggunakan API MemoryDB

Anda dapat menggunakan API MemoryDB untuk menghapus tanda dari klaster MemoryDB yang ada dengan menggunakanUntagResourceoperasi.

Kode berikut menggunakan MemoryDB API untuk menghapus tag dengan kunci`Service`dan`Region`dari clustermy-clusterdi wilayah us-east-1.

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=UntagResource
   &ResourceArn=arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &TagKeys.member.1=Service
   &TagKeys.member.2=Region
   &Version=2021-01-01
   &Timestamp=20210802T192317Z
   &X-Amz-Credential=<credential>
```

Mengelola pemeliharaan

Setiap klaster memiliki jendela pemeliharaan mingguan di mana pada waktu itu setiap perubahan sistem akan diterapkan. Jika Anda tidak menentukan jendela pemeliharaan yang diinginkan saat Anda membuat atau mengubah klaster, MemoryDB menetapkan jendela pemeliharaan 60 menit pada hari yang dipilih secara acak dalam seminggu.

Jendela pemeliharaan selama 60 menit dipilih secara acak dari blok waktu 8 jam di setiap Wilayah. Daftar tabel berikut mencantumkan blok waktu untuk setiap wilayah tempat jendela pemeliharaan default 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

Kode Wilayah	Nama Wilayah	Jendela Pemeliharaan Wilayah
ap-northeast-2	Wilayah Asia Pacific (Seoul)	12:00–20:00 UTC
ap-south-1	Wilayah Asia Pacific (Mumbai)	17:30–1:30 UTC
ap-southeast-1	Wilayah Asia Pasifik (Singapore)	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
cn-north-1	Wilayah China (Beijing)	14:00–22:00 UTC
cn-northwest-1	Wilayah China (Ningxia)	14:00–22:00 UTC
eu-west-3	Wilayah EU (Paris)	23:59–07:29 UTC
eu-central-1	Wilayah Europe (Frankfurt)	23:00–07:00 UTC
eu-west-1	Wilayah Eropa (Irlandia)	22:00–06:00 UTC
eu-west-2	Wilayah Europe (London)	23:00–07:00 UTC
sa-east-1	Wilayah South America (São Paulo)	01:00–09:00 UTC
ca-central-1	Wilayah Canada (Central)	03.00–11.00 UTC
us-east-1	Wilayah AS Timur (N. Virginia)	03:00–11:00 UTC
us-east-1	Wilayah US East (Ohio)	04:00 — 12:00 UTC
us-west-1	Wilayah Barat AS (N. California)	06:00–14:00 UTC
us-west-2	Wilayah US West (Oregon)	06.00–14:00 UTC

Mengubah Jendela Pemeliharaan Klaster Anda

Jendela pemeliharaan harus berada pada waktu penggunaan terendah dan karena itu mungkin memerlukan perubahan dari waktu ke waktu. Anda dapat mengubah klaster Anda untuk menentukan

rentang waktu hingga durasi 24 jam di mana pada rentang waktu itu semua kegiatan pemeliharaan yang Anda minta harus terjadi. Setiap perubahan klaster yang Anda minta yang ditangguhkan atau ditunda akan terjadi selama waktu ini.

Informasi lain

Untuk informasi tentang jendela pemeliharaan dan penggantian simpul, lihat hal berikut:

- Mengganti simpul—Mengelola penggantian simpul
- Memodifikasi cluster MemoryDB—Mengubah jendela pemeliharaan klaster

Praktik terbaik

Berikutnya, Anda dapat menemukan praktik terbaik yang direkomendasikan untuk MemoryDB untuk Redis. Mengikuti langkah ini meningkatkan kinerja dan keandalan klaster Anda.

Topik

- Perintah Redis Terbatas
- Ketahanan dalam MemoryDB untuk Redis
- Praktik terbaik: Pub/Sub dan Peningkatan I/O Multiplexing
- Praktik terbaik: Mengubah ukuran klaster secara online

Perintah Redis Terbatas

Untuk memberikan pengalaman layanan terkelola, MemoryDB membatasi akses ke perintah tertentu yang memerlukan hak istimewa lanjutan. Perintah berikut tidak tersedia:

- `acl deluser`
- `acl load`
- `acl save`
- `acl setuser`
- `bgrewriteaof`
- `bgsave`
- `cluster addslot`
- `cluster delslot`
- `cluster setslot`
- `config`
- `debug`
- `migrate`
- `module`
- `psync`
- `replicaof`
- `save`
- `shutdown`
- `slaveof`
- `sync`

Ketahanan dalam MemoryDB untuk Redis

Infrastruktur global AWS dibangun di sekitar Wilayah AWS dan Availability Zone. AWS Wilayah menyediakan beberapa Availability Zone yang terpisah dan terisolasi secara fisik, yang terhubung dengan jaringan yang memiliki latensi rendah, throughput tinggi, dan sangat berkelebihan. Dengan Availability Zone, Anda dapat merancang dan mengoperasikan aplikasi dan basis data yang secara otomatis melakukan fail over di antara Availability Zone tanpa gangguan. Availability Zone memiliki ketersediaan yang lebih baik, menoleransi kegagalan, dan dapat diskalakan dibandingkan satu atau beberapa infrastruktur pusat data tradisional.

Untuk informasi selengkapnya tentang Wilayah AWS dan Availability Zone, lihat AWS Infrastruktur Global.

SelainAWSinfrastruktur global, MemoryDB untuk Redis menawarkan beberapa fitur untuk membantu mendukung ketahanan data dan kebutuhan snapshot Anda.

Topik

Mengurangi Kegagalan

Ketika merencanakan MemoryDB Anda untuk implementasi Redis, 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.

Kegagalan: klaster MemoryDB

Sebuah cluster MemoryDB terdiri dari node primer tunggal yang aplikasi Anda berdua dapat membaca dari dan menulis ke, dan dari 0 sampai 5 read-only replika node. Namun, kami sangat menyarankan untuk menggunakan setidaknya 1 replika untuk ketersediaan tinggi. Setiap kali data ditulis ke node primer itu tetap ke log transaksi dan asynchronously diperbarui pada node replika.

Ketika salah satu replika baca gagal

1. MemoryDB mendeteksi replika gagal.
2. MemoryDB mengambil node gagal offline.
3. MemoryDB meluncurkan dan menyediakan node pengganti di AZ yang sama.

4. Node baru disinkronkan dengan log transaksi.

Selama proses ini, aplikasi Anda dapat terus membaca dan menulis menggunakan simpul lain.

Multi-AZ

Jika Multi-AZ diaktifkan pada klaster MemoryDB Anda, primer yang gagal akan terdeteksi dan diganti secara otomatis.

1. MemoryDB mendeteksi kegagalan node primer.
2. MemoryDB gagal ke replika setelah memastikan itu konsisten dengan primer gagal.
3. MemoryDB berputar replika di AZ primer gagal.
4. Node baru disinkronkan dengan log transaksi.

Melakukan fail over ke simpul replika umumnya lebih cepat daripada membuat dan menetapkan simpul primer baru. Ini berarti aplikasi Anda dapat melanjutkan menulis ke node utama Anda lebih cepat.

Untuk informasi selengkapnya, lihat Meminimalkan waktu henti di MemoryDB dengan Multi-AZ.

Praktik terbaik: Pub/Sub dan Peningkatan I/O Multiplexing

Saat menggunakan Redis versi 7 atau yang lebih baru, sebaiknya gunakan Sharded Pub/Sub. Anda juga meningkatkan throughput dan latensi menggunakan multiplexing I/O yang disempurnakan, yang secara otomatis tersedia saat menggunakan Redis versi 7 atau yang lebih baru dan tidak memerlukan perubahan klien. Ini sangat ideal untuk pub/sub beban kerja, yang sering throughput-terikat dengan beberapa koneksi klien.

Praktik terbaik: Mengubah ukuran klaster secara online

Resharding melibatkan proses menambahkan dan menghapus serpihan atau simpul ke klaster Anda dan mendistribusikan ulang keyspace. Akibatnya, beberapa hal memiliki dampak pada operasi resharding, seperti beban pada klaster, pemanfaatan memori, dan ukuran keseluruhan data. Untuk pengalaman terbaik, sebaiknya Anda mengikuti 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 menjaga pemanfaatan CPU di bawah 80 persen pada instans multicore dan kurang dari 50 persen pada instans inti tunggal selama resharding. Pantau metrik MemoryDB dan mulai resharding sebelum aplikasi Anda mulai mengalami masalah penskalaan. Metrik yang berguna untuk melacak adalah `CPUUtilization`, `NetworkBytesIn`, `NetworkBytesOut`, `CurrConnections`, `NewConnections`, `FreeableMemory`, `SwapUsage`, dan `BytesUsedForMemoryDB`.
- Pastikan memori bebas yang cukup tersedia sebelum penskalaan masuk - Jika Anda melakukan penskalaan masuk, pastikan bahwa memori bebas yang tersedia di serpihan akan disimpan setidaknya 1,5 kali memori yang digunakan pada serpihan yang akan dihapus.
- Memulai 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 redistribusi slot.
- Tinjau perilaku timeout klien — Beberapa klien mungkin mengalami latensi yang lebih tinggi selama perubahan ukuran klaster secara online. Membuat konfigurasi pustaka klien Anda dengan batas waktu yang lebih tinggi dapat membantu dengan memberikan waktu sistem untuk menyambung

bahkan di bawah kondisi beban yang lebih tinggi pada server. Dalam beberapa kasus, Anda mungkin membuka sejumlah besar koneksi ke server. Dalam kasus ini, pertimbangkan untuk menambahkan backoff eksponensial untuk menyambung kembali logika. Melakukan hal ini dapat membantu mencegah lonjakan koneksi baru di server pada waktu yang sama.

Selama resharding, sebaiknya melakukan hal berikut:

- Hindari perintah yang mahal — Hindari menjalankan operasi yang intensif dalam hal komputasi dan I/O, seperti perintah dan. `KEYS` `SMEMBERS` Pendekatan ini disarankan karena operasi ini meningkatkan beban pada klaster dan memiliki dampak pada kinerja klaster. Alih-alih, gunakan perintah `SCAN` dan `SSCAN`.
- Ikuti praktik terbaik Lua - Hindari menjalankan script Lua terlalu lama, dan selalu menyatakan 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.

Setelah resharding, perhatikan hal berikut:

- Penskalaan masuk mungkin berhasil sebagian jika memori tidak cukup terjadi pada serpihan target. Jika hasil seperti itu terjadi, tinjau memori yang tersedia dan coba lagi operasi itu, jika perlu.
- Slot dengan item besar tidak dimigrasikan. Secara khusus, slot dengan item yang lebih besar dari 256 MB pasca-serialisasi tidak dimigrasikan.
- `FLUSHALL`dan `FLUSHDB` perintah tidak didukung dalam skrip Lua selama operasi resharding.

Memahami replikasi MemoryDB

MemoryDB mengimplementasikan replikasi dengan data yang dipartisi hingga 500 pecahan.

Setiap serpihan di dalam klaster memiliki simpul primer baca/tulis tunggal dan hingga 5 simpul replika baca-saja. Setiap node utama dapat mempertahankan hingga 100 MB/s. Anda dapat membuat klaster dengan jumlah serpihan lebih banyak dan jumlah replika lebih sedikit dengan jumlah hingga 500 simpul per klaster. Konfigurasi klaster ini dapat berkisar dari 500 serpihan dan 0 replika hingga 100 serpihan dan 4 replika, yang merupakan jumlah replika maksimum yang diperbolehkan.

Konsistensi

Dalam MemoryDB, node primer sangat konsisten. Operasi penulisan yang berhasil disimpan secara tahan lama dalam log transaksional Multi-AZ terdistribusi sebelum kembali ke klien. Operasi baca pada pendahuluan selalu mengembalikan sebagian besar up-to-date data yang mencerminkan efek dari semua operasi penulisan yang berhasil sebelumnya. Konsistensi yang kuat seperti itu dipertahankan di seluruh failovers primer.

Di MemoryDB, simpul replika pada akhirnya konsisten. Operasi baca dari replika (menggunakan `READONLY` perintah) mungkin tidak selalu mencerminkan efek operasi penulisan terbaru yang berhasil, dengan metrik lag yang dipublikasikan. CloudWatch Namun, operasi baca dari satu replika konsisten secara berurutan. Operasi menulis sukses berlaku pada setiap replika dalam urutan yang sama mereka dieksekusi pada primer.

Replikasi dalam sebuah cluster

Setiap replika baca dalam serpihan memelihara salinan data dari simpul primer pada serpihan. Mekanisme replikasi asinkron menggunakan log transaksi 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. Sejak MemoryDB menyimpan data dalam log transaksi tahan lama, tidak ada risiko bahwa data akan hilang. Data dipartisi di seluruh serpihan di dalam klaster MemoryDB.

Aplikasi menggunakan titik akhir klaster MemoryDB klaster untuk terhubung dengan simpul di dalam klaster. Untuk informasi selengkapnya, lihat Menemukan titik akhir sambungan.

cluster MemoryDB bersifat regional dan dapat berisi node hanya dari satu Region. Untuk meningkatkan toleransi kesalahan, Anda harus menyediakan primer dan membaca replika di beberapa Availability Zones di beberapa Availability Zones di dalam wilayah tersebut.

Menggunakan replikasi, yang menyediakan Anda dengan Multi-AZ, sangat dianjurkan untuk semua cluster MemoryDB. Untuk informasi selengkapnya, lihat Meminimalkan waktu henti di MemoryDB dengan Multi-AZ.

Meminimalkan waktu henti di MemoryDB dengan Multi-AZ

Ada sejumlah contoh di mana MemoryDB mungkin perlu mengganti simpul utama; ini termasuk jenis pemeliharaan terencana tertentu dan kejadian yang tidak mungkin terjadi pada node utama atau kegagalan Availability Zone.

Respon terhadap kegagalan node tergantung pada node mana yang gagal. Namun, dalam semua kasus, MemoryDB memastikan bahwa tidak ada data yang hilang selama penggantian node atau failover. Misalnya, jika replika gagal, node gagal diganti dan data disinkronkan dari log transaksi. Jika node utama gagal, failover dipicu ke replika konsisten yang memastikan tidak ada data yang hilang selama failover. Penulisan sekarang disajikan dari node primer baru. Node primer lama kemudian diganti dan disinkronkan dari log transaksi.

Jika node utama gagal pada beling node tunggal (tidak ada replika), MemoryDB berhenti menerima penulisan sampai node utama diganti dan disinkronkan dari log transaksi.

Penggantian node dapat mengakibatkan beberapa waktu henti untuk klaster, tetapi jika Multi-AZ aktif, waktu henti diminimalkan. Peran node primer secara otomatis akan gagal ke salah satu replika. Tidak perlu membuat dan menyediakan node primer baru, karena MemoryDB akan menangani ini secara transparan. Failover dan promosi replika ini memastikan Anda dapat melanjutkan menulis ke primer baru segera setelah promosi selesai.

Dalam kasus penggantian node yang direncanakan dimulai karena pembaruan pemeliharaan atau pembaruan layanan, perhatikan penggantian node yang direncanakan selesai sementara cluster melayani permintaan tulis yang masuk.

Multi-AZ pada cluster MemoryDB Anda meningkatkan toleransi kesalahan Anda. Hal ini berlaku terutama dalam kasus di mana node utama klaster Anda menjadi tidak terjangkau atau gagal karena alasan apa pun. Multi-AZ pada cluster MemoryDB mengharuskan setiap pecahan memiliki lebih dari satu node, dan diaktifkan secara otomatis.

Topik

Skenario kegagalan dengan respons Multi-AZ

Jika Multi-AZ aktif, node primer yang gagal gagal ke replika yang tersedia. Replika secara otomatis disinkronkan dengan log transaksi dan menjadi primer, yang jauh lebih cepat daripada membuat

dan mereprovisioning node primer baru. Proses ini biasanya memakan waktu hanya beberapa detik hingga Anda dapat menulis lagi ke klaster.

Ketika Multi-AZ aktif, MemoryDB terus memonitor keadaan node utama. Jika simpul primer gagal, salah satu tindakan berikut akan dilakukan tergantung pada jenis kegagalan.

Topik

- Skenario kegagalan ketika hanya simpul primer yang gagal
- Skenario kegagalan ketika node utama dan beberapa replika gagal
- Skenario kegagalan ketika seluruh klaster gagal

Skenario kegagalan ketika hanya simpul primer yang gagal

Jika hanya node utama gagal, replika secara otomatis akan menjadi primer. Replika pengganti kemudian dibuat dan disediakan di Availability Zone yang sama dengan primer yang gagal.

Ketika hanya node utama gagal, MemoryDB Multi-AZ melakukan hal berikut:

1. Simpul primer yang gagal akan dibuat offline.
2. up-to-dateReplika secara otomatis menjadi primer.

   Menulis dapat melanjutkan segera setelah proses failover selesai, biasanya hanya beberapa detik.
3. Replika pengganti diluncurkan dan disediakan.

   Replika pengganti diluncurkan di Availability Zone tempat node utama gagal berada sehingga distribusi node dipertahankan.
4. Replika disinkronkan dengan log transaksi.

Untuk informasi tentang menemukan titik akhir klaster, lihat topik berikut:

- Menemukan Endpoint untuk Cluster MemoryDB (MemoryDB API)

Skenario kegagalan ketika node utama dan beberapa replika gagal

Jika primer dan setidaknya satu replika gagal, up-to-date replika dipromosikan ke klaster primer. Replika baru juga dibuat dan disediakan di Availability Zone yang sama dengan node yang gagal.

Ketika node utama dan beberapa replika gagal, MemoryDB Multi-AZ melakukan hal berikut:

1. Node primer yang gagal dan replika gagal diambil secara offline.
2. Replika yang tersedia akan menjadi simpul utama.

   Menulis dapat melanjutkan segera setelah failover selesai, biasanya hanya beberapa detik.
3. Replika pengganti dibuat dan ditetapkan.

   Replika pengganti dibuat di Availability Zone dari simpul yang gagal sehingga distribusi simpul tetap terpelihara.
4. Semua node disinkronkan dengan log transaksi.

Untuk informasi tentang menemukan titik akhir klaster, lihat topik berikut:

- Menemukan Endpoint untuk Cluster MemoryDB (AWSCLI)
- Menemukan Endpoint untuk Cluster MemoryDB (MemoryDB API)

Skenario kegagalan ketika seluruh klaster gagal

Jika semuanya gagal, maka semua simpul dibuat kembali dan ditetapkan pada Availability Zone yang sama dengan simpul asli.

Tidak ada kehilangan data dalam skenario ini karena data tetap ada di log transaksi.

Ketika seluruh cluster gagal, MemoryDB Multi-AZ melakukan hal berikut:

1. Node primer dan replika yang gagal diambil secara offline.
2. Node primer pengganti dibuat dan disediakan, disinkronkan dengan log transaksi.
3. Replika pengganti dibuat dan disediakan, disinkronkan dengan log transaksi.

   Penggantinya dibuat di Availability Zone dari simpul yang gagal sehingga distribusi simpul tetap terpelihara.

Untuk informasi tentang menemukan titik akhir klaster, lihat topik berikut:

- Menemukan Endpoint untuk Cluster MemoryDB (AWSCLI)

- Menemukan Endpoint untuk Cluster MemoryDB (MemoryDB API)

Menguji failover otomatis

Anda dapat menguji failover otomatis menggunakan konsol MemoryDB,AWS CLI, dan API MemoryDB.

Saat menguji, perhatikan hal berikut:

- Anda dapat menggunakan operasi ini hingga lima kali dalam periode 24 jam.
- Jika Anda memanggil operasi ini pada pecahan di kluster yang berbeda, Anda dapat melakukan panggilan secara bersamaan.
- Dalam beberapa kasus, Anda mungkin memanggil operasi ini beberapa kali pada pecahan yang berbeda di cluster MemoryDB yang sama. Dalam kasus seperti itu, penggantian simpul pertama harus selesai sebelum panggilan berikutnya dapat dibuat.
- Untuk menentukan apakah penggantian simpul selesai, periksa peristiwa menggunakan konsol MemoryDB untuk Redis,AWS CLI, atau API MemoryDB. Cari peristiwa berikut yang terkait dengan`FailoverShard`, tercantum di sini dalam urutan kemungkinan terjadinya:

  1. pesan cluster: `FailoverShard API called for shard <shard-id>`
  2. pesan cluster: `Failover from primary node <primary-node-id> to replica node <node-id> completed`
  3. pesan cluster: `Recovering nodes <node-id>`
  4. pesan cluster: `Finished recovery for nodes <node-id>`

  Untuk informasi selengkapnya, lihat yang berikut:

  - DescribeEventsdalam Referensi API MemoryDB
- API ini dirancang untuk menguji perilaku aplikasi Anda dalam kasus failover MemoryDB. Hal ini tidak dirancang untuk menjadi alat operasional untuk memulai failover untuk mengatasi masalah dengan cluster. Selain itu, dalam kondisi tertentu seperti peristiwa operasional skala besar, AWS dapat memblokir API ini.

Topik

- Menguji failover otomatis menggunakan AWS Management Console
- Menguji failover otomatis menggunakan AWS CLI
- Menguji failover otomatis menggunakan API MemoryDB

Menguji failover otomatis menggunakan AWS Management Console

Gunakan prosedur berikut untuk menguji failover otomatis dengan konsol.

1. Masuk ke AWS Management Console dan buka konsol MemoryDB untuk Redis di https://console.aws.amazon.com/memorydb/.
2. Pilih tombol radio di sebelah kiri cluster yang ingin Anda uji. Cluster ini harus memiliki setidaknya satu simpul replika.
3. Pada bagian Perincian, 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 Memodifikasi cluster MemoryDB.
4. Pilih nama klaster.
5. Pada halaman Shards dan node, untuk pecahan yang ingin Anda uji failover, pilih nama shard.
6. Untuk node, pilih Failover Primary.
7. Pilih Lanjutkan untuk fail over ke primer, atau Batalkan untuk membatalkan dan tidak fail over ke simpul primer.

   Selama proses failover, konsol terus menunjukkan status simpul sebagai tersedia. Untuk melacak kemajuan uji failover Anda, pilih Peristiwa dari panel navigasi konsol. Pada tab Peristiwa, perhatikan peristiwa yang menunjukkan failover Anda telah dimulai (`FailoverShard API called`) dan selesai (`Recovery completed`).

Menguji failover otomatis menggunakan AWS CLI

Anda dapat menguji failover otomatis pada klaster berkemampuan Multi-AZ menggunakan operasi failover-shard. AWS CLI

Parameter

- `--cluster-name` – Diperlukan. Cluster yang akan diuji.
- `--shard-name` – Diperlukan. Nama pecahan yang ingin Anda uji failover otomatis. Anda dapat menguji maksimal lima pecahan dalam periode 24 jam bergulir.

Contoh berikut menggunakan AWS CLI untuk memanggil pecahan `failover-shard 0001` di cluster MemoryDB. `my-cluster`

Untuk Linux, macOS, atau Unix:

```
aws memorydb failover-shard \
   --cluster-name my-cluster \
   --shard-name 0001
```

Untuk Windows:

```
aws memorydb failover-shard ^
   --cluster-name my-cluster ^
   --shard-name 0001
```

Untuk melacak kemajuan failover Anda, gunakan operasi AWS CLI `describe-events`.

Ini akan mengembalikan respon JSON berikut:

```
{
    "Events": [
        {
            "SourceName": "my-cluster",
            "SourceType": "cluster",
            "Message": "Failover to replica node my-cluster-0001-002 completed",
            "Date": "2021-08-22T12:39:37.568000-07:00"
        },
        {
            "SourceName": "my-cluster",
            "SourceType": "cluster",
            "Message": "Starting failover for shard 0001",
            "Date": "2021-08-22T12:39:10.173000-07:00"
        }
    ]
}
```

Untuk informasi selengkapnya, lihat yang berikut:

- kegagalan-beling
- menggambarkan-peristiwa

Menguji failover otomatis menggunakan API MemoryDB

Contoh berikut panggilan `FailoverShard` pada pecahan `0003` di `memorydb00` cluster.

Example Menguji failover otomatis

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=FailoverShard
    &ShardName=0003
    &ClusterName=memorydb00
    &Version=2021-01-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210801T192317Z
    &X-Amz-Credential=<credential>
```

Untuk melacak kemajuan failover Anda, gunakan operasi API MemoryDB`DescribeEvents`.

Untuk informasi selengkapnya, lihat yang berikut:

- FailoverShard
- DescribeEvents

Mengubah jumlah replika

Anda dapat secara dinamis menambah atau mengurangi jumlah replika baca dalam klaster MemoryDB Anda menggunakanAWS Management Console, atau API MemoryDB. AWS CLI Semua serpihan harus memiliki jumlah replika yang sama.

Menambah jumlah replika dalam klaster

Anda dapat menambah jumlah replika di dalam klaster MemoryDB maksimum hingga lima per serpihan. Anda dapat melakukannya menggunakanAWS Management Console,AWS CLI, atau API MemoryDB.

Topik

- Menggunakan AWS Management Console
- Menggunakan AWS CLI
- Menggunakan API MemoryDB

Menggunakan AWS Management Console

Untuk menambah jumlah replika di dalam klaster (console) MemoryDB, lihat. Menambahkan/Menghapus node dari cluster

Menggunakan AWS CLI

Untuk menambah jumlah replika di dalam klaster MemoryDB, gunakan `update-cluster` perintah dengan parameter berikut:

- `--cluster-name` – Diperlukan. Mengidentifikasi klaster yang ingin ditambah jumlah replika di dalamnya.
- `--replica-configuration` – Diperlukan. Memungkinkan Anda mengatur jumlah replika. Untuk menambah jumlah replika, atur `ReplicaCount` properti ke jumlah replika yang Anda inginkan di dalam serpihan ini pada akhir operasi ini.

Example

Contoh berikut menambah jumlah replika di dalam klaster `my-cluster` menjadi 2.

Untuk Linux, macOS, atau Unix:

```
aws memorydb update-cluster \
    --cluster-name my-cluster \
    --replica-configuration \
        ReplicaCount=2
```

Untuk Windows:

```
aws memorydb update-cluster ^
    --cluster-name my-cluster ^
    --replica-configuration ^
        ReplicaCount=2
```

Ia mengembalikan respon JSON berikut:

```
{
    "Cluster": {
        "Name": "my-cluster",
        "Status": "updating",
        "NumberOfShards": 1,
        "ClusterEndpoint": {
            "Address": "clustercfg.my-cluster.xxxxx.memorydb.us-east-1.amazonaws.com",
            "Port": 6379
        },
        "NodeType": "db.r6g.large",
        "EngineVersion": "6.2",
        "EnginePatchVersion": "6.2.6",
        "ParameterGroupName": "default.memorydb-redis6",
        "ParameterGroupStatus": "in-sync",
        "SubnetGroupName": "my-sg",
        "TLSEnabled": true,
        "ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",
        "SnapshotRetentionLimit": 0,
        "MaintenanceWindow": "wed:03:00-wed:04:00",
        "SnapshotWindow": "04:30-05:30",
        "DataTiering": "false",
        "AutoMinorVersionUpgrade": true
    }
}
```

Untuk melihat rincian klaster yang diperbarui setelah statusnya berubah dari memperbarui ke tersedia, gunakan perintah berikut:

Untuk Linux, macOS, atau Unix:

```
aws memorydb describe-clusters \
    --cluster-name my-cluster
    --show-shard-details
```

Untuk Windows:

```
aws memorydb describe-clusters ^
    --cluster-name my-cluster
    --show-shard-details
```

Ini akan mengembalikan respon JSON berikut:

```
{
    "Clusters": [
        {
            "Name": "my-cluster",
            "Status": "available",
            "NumberOfShards": 1,
            "Shards": [
                {
                    "Name": "0001",
                    "Status": "available",
                    "Slots": "0-16383",
                    "Nodes": [
                        {
                            "Name": "my-cluster-0001-001",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1a",
                            "CreateTime": "2021-08-21T20:22:12.405000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        },
                        {
                            "Name": "my-cluster-0001-002",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1b",
                            "CreateTime": "2021-08-21T20:22:12.405000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        },
                        {
                            "Name": "my-cluster-0001-003",
```

```
                            "Status": "available",
                            "AvailabilityZone": "us-east-1a",
                            "CreateTime": "2021-08-22T12:59:31.844000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        }
                    ],
                    "NumberOfNodes": 3
                }
            ],
            "ClusterEndpoint": {
                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                "Port": 6379
            },
            "NodeType": "db.r6g.large",
            "EngineVersion": "6.2",
            "EnginePatchVersion": "6.2.6",
            "ParameterGroupName": "default.memorydb-redis6",
            "ParameterGroupStatus": "in-sync",
            "SubnetGroupName": "my-sg",
            "TLSEnabled": true,
            "ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",
            "SnapshotRetentionLimit": 0,
            "MaintenanceWindow": "wed:03:00-wed:04:00",
            "SnapshotWindow": "04:30-05:30",
            "ACLName": "my-acl",
            "DataTiering": "false",
            "AutoMinorVersionUpgrade": true
        }
    ]
}
```


Untuk informasi lebih lanjut tentang meningkatkan jumlah replika menggunakan CLI, lihat update-cluster di Command Reference. AWS CLI

Menggunakan API MemoryDB

Untuk menambah jumlah replika di dalam serpihan MemoryDB, gunakan `UpdateCluster` tindakan dengan parameter berikut:

- `ClusterName` – Diperlukan. Mengidentifikasi klaster yang ingin ditambah jumlah replika di dalamnya.
- `ReplicaConfiguration` – Diperlukan. Memungkinkan Anda mengatur jumlah replika. Untuk menambah jumlah replika, atur `ReplicaCount` properti ke jumlah replika yang Anda inginkan di dalam serpihan ini pada akhir operasi ini.

Example

Contoh berikut menambah jumlah replika di dalam klaster `sample-cluster` menjadi tiga. Saat contoh ini selesai, terdapat tiga replika di setiap serpihan. Nomor ini berlaku apakah ini adalah cluster MemoryDB dengan pecahan tunggal atau cluster MemoryDB dengan beberapa pecahan.

```
https://memory-db.us-east-1.amazonaws.com/
      ?Action=UpdateCluster
      &ReplicaConfiguration.ReplicaCount=3
      &ClusterName=sample-cluster
      &Version=2021-01-01
      &SignatureVersion=4
      &SignatureMethod=HmacSHA256
      &Timestamp=20210802T192317Z
      &X-Amz-Credential=<credential>
```

Untuk informasi lain tentang menambah jumlah replika menggunakan API, lihat UpdateCluster.

Mengurangi jumlah replika dalam klaster

Anda dapat mengurangi jumlah replika di dalam klaster untuk MemoryDB. Anda dapat mengurangi jumlah replika menjadi nol, tetapi Anda tidak dapat failover ke replika jika simpul primer Anda gagal.

Anda dapat menggunakanAWS Management Console, AWS CLI atau API MemoryDB untuk mengurangi jumlah replika di dalam klaster.

Topik

- Menggunakan AWS Management Console
- Menggunakan AWS CLI
- Menggunakan API MemoryDB

Menggunakan AWS Management Console

Untuk mengurangi jumlah replika di dalam klaster (console) MemoryDB, lihat. Menambahkan/Menghapus node dari cluster

Menggunakan AWS CLI

Untuk mengurangi jumlah replika di dalam klaster MemoryDB, gunakan `update-cluster` perintah dengan parameter berikut:

- `--cluster-name` – Diperlukan. Mengidentifikasi klaster yang ingin dikurangi jumlah replika di dalamnya.
- `--replica-configuration` – Diperlukan.

  `ReplicaCount`- Mengatur properti ini untuk menentukan jumlah simpul replika yang Anda inginkan.

Example

Contoh berikut menggunakan `--replica-configuration` untuk mengurangi jumlah replika di dalam klaster dengan nilai `my-cluster` yang ditentukan.

Untuk Linux, macOS, atau Unix:

```
aws memorydb update-cluster \
```

```
    --cluster-name my-cluster \
    --replica-configuration \
        ReplicaCount=1
```

Untuk Windows:

```
aws memorydb update-cluster ^
    --cluster-name my-cluster ^
    --replica-configuration ^
        ReplicaCount=1 ^
```

Ini akan mengembalikan respon JSON berikut:

```
{
    "Cluster": {
        "Name": "my-cluster",
        "Status": "updating",
        "NumberOfShards": 1,
        "ClusterEndpoint": {
            "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-east-1.amazonaws.com",
            "Port": 6379
        },
        "NodeType": "db.r6g.large",
        "EngineVersion": "6.2",
        "EnginePatchVersion": "6.2.6",
        "ParameterGroupName": "default.memorydb-redis6",
        "ParameterGroupStatus": "in-sync",
        "SubnetGroupName": "my-sg",
        "TLSEnabled": true,
        "ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",
        "SnapshotRetentionLimit": 0,
        "MaintenanceWindow": "wed:03:00-wed:04:00",
        "SnapshotWindow": "04:30-05:30",
        "DataTiering": "false",
        "AutoMinorVersionUpgrade": true
    }
}
```

Untuk melihat rincian klaster yang diperbarui setelah statusnya berubah dari memperbarui ke tersedia, gunakan perintah berikut:

Untuk Linux, macOS, atau Unix:

```
aws memorydb describe-clusters \
    --cluster-name my-cluster
    --show-shard-details
```

Untuk Windows:

```
aws memorydb describe-clusters ^
    --cluster-name my-cluster
    --show-shard-details
```

Ini akan mengembalikan respon JSON berikut:

```
{
    "Clusters": [
        {
            "Name": "my-cluster",
            "Status": "available",
            "NumberOfShards": 1,
            "Shards": [
                {
                    "Name": "0001",
                    "Status": "available",
                    "Slots": "0-16383",
                    "Nodes": [
                        {
                            "Name": "my-cluster-0001-001",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1a",
                            "CreateTime": "2021-08-21T20:22:12.405000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        },
                        {
                            "Name": "my-cluster-0001-002",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1b",
                            "CreateTime": "2021-08-21T20:22:12.405000-07:00",
                            "Endpoint": {
```

```
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        }
                    ],
                    "NumberOfNodes": 2
                }
            ],
            "ClusterEndpoint": {
                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                "Port": 6379
            },
            "NodeType": "db.r6g.large",
            "EngineVersion": "6.2",
            "EnginePatchVersion": "6.2.6",
            "ParameterGroupName": "default.memorydb-redis6",
            "ParameterGroupStatus": "in-sync",
            "SubnetGroupName": "my-sg",
            "TLSEnabled": true,
            "ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",
            "SnapshotRetentionLimit": 0,
            "MaintenanceWindow": "wed:03:00-wed:04:00",
            "SnapshotWindow": "04:30-05:30",
            "ACLName": "my-acl",
            "DataTiering": "false",
            "AutoMinorVersionUpgrade": true
        }
    ]
}
```


Untuk informasi lebih lanjut tentang mengurangi jumlah replika menggunakan CLI, lihat update-cluster di Command Reference. AWS CLI

Menggunakan API MemoryDB

Untuk mengurangi jumlah replika di dalam klaster MemoryDB, gunakan `UpdateCluster` tindakan dengan parameter berikut:

- `ClusterName` – Diperlukan. Mengidentifikasi klaster yang ingin dikurangi jumlah replika di dalamnya.
- `ReplicaConfiguration` – Diperlukan. Memungkinkan Anda mengatur jumlah replika.

`ReplicaCount`- Mengatur properti ini untuk menentukan jumlah simpul replika yang Anda inginkan.

Example

Contoh berikut menggunakan `ReplicaCount` untuk mengurangi jumlah replika di dalam klaster `sample-cluster` menjadi satu. Saat contoh ini selesai, terdapat satu replika di setiap serpihan. Nomor ini berlaku apakah ini adalah cluster MemoryDB dengan pecahan tunggal atau cluster MemoryDB dengan beberapa pecahan.

```
https://memory-db.us-east-1.amazonaws.com/
      ?Action=UpdateCluster
      &ReplicaConfiguration.ReplicaCount=1
      &ClusterName=sample-cluster
      &Version=2021-01-01
      &SignatureVersion=4
      &SignatureMethod=HmacSHA256
      &Timestamp=20210802T192317Z
      &X-Amz-Credential=<credential>
```

Untuk informasi lain tentang menurunkan jumlah replika menggunakan API, lihat. UpdateCluster

Snapshot dan pulihkan

Cluster MemoryDB untuk Redis secara otomatis mencadangkan data ke log transaksional multi-AZ, tetapi Anda dapat memilih untuk membuat point-in-time snapshot cluster baik secara berkala maupun sesuai permintaan. Snapshot ini dapat digunakan untuk membuat ulang cluster pada titik sebelumnya atau untuk menyemai cluster baru. Snapshot terdiri dari metadata cluster, bersama dengan semua data di cluster. Semua snapshot ditulis ke Amazon Simple Storage Service (Amazon S3), yang menyediakan penyimpanan tahan lama. Kapan saja, Anda dapat memulihkan data Anda dengan membuat cluster MemoryDB baru dan mengisinya dengan data dari snapshot. Dengan MemoryDB, Anda dapat mengelola snapshot menggunakan API, AWS Command Line Interface (AWS CLI)AWS Management Console, dan MemoryDB.

Topik

- Kendala snapshot
- Biaya snapshot

- Menjadwalkan snapshot otomatis
- Membuat snapshot manual
- Membuat snapshot akhir
- Menggambarkan snapshot
- Menyalin snapshot
- Mengekspor snapshot
- Memulihkan dari snapshot
- Menyemai cluster baru dengan snapshot yang dibuat secara eksternal
- Menandai snapshot
- Menghapus snapshot

Kendala snapshot

Pertimbangkan kendala berikut saat merencanakan atau membuat snapshot:

- Untuk cluster MemoryDB, snapshot dan restore tersedia untuk semua jenis node yang didukung.
- Selama periode 24 jam yang berdekatan, Anda dapat membuat tidak lebih dari 20 snapshot manual per cluster.
- MemoryDB hanya mendukung pengambilan snapshot di tingkat cluster. MemoryDB tidak mendukung pengambilan snapshot di level shard atau node.
- Selama proses snapshot, Anda tidak dapat menjalankan operasi API atau CLI lainnya di cluster.
- Jika Anda menghapus cluster dan meminta snapshot akhir, MemoryDB selalu mengambil snapshot dari node utama. Ini memastikan bahwa Anda menangkap data terbaru sebelum cluster dihapus.

Biaya snapshot

Menggunakan MemoryDB, Anda dapat menyimpan satu snapshot untuk setiap cluster MemoryDB aktif secara gratis. Ruang penyimpanan untuk snapshot tambahan dikenakan biaya sebesar $0,085/GB per bulan untuk semua Wilayah. AWS Tidak ada biaya transfer data untuk membuat snapshot, atau untuk memulihkan data dari snapshot ke cluster MemoryDB.

Menjadwalkan snapshot otomatis

Untuk cluster MemoryDB apa pun, Anda dapat mengaktifkan snapshot otomatis. Saat snapshot otomatis diaktifkan, MemoryDB membuat snapshot cluster setiap hari. Tidak ada dampak pada klaster dan perubahan itu terjadi seketika. Untuk informasi selengkapnya, lihat Memulihkan dari snapshot.

Saat Anda menjadwalkan snapshot otomatis, Anda harus merencanakan pengaturan berikut:

- Jendela snapshot — Periode selama setiap hari ketika MemoryDB mulai membuat snapshot. Panjang minimum untuk jendela snapshot adalah 60 menit. Anda dapat mengatur jendela snapshot kapan saja ketika itu paling nyaman bagi Anda, atau untuk waktu hari yang menghindari melakukan snapshot selama periode pemanfaatan yang sangat tinggi.

  Jika Anda tidak menentukan jendela snapshot, MemoryDB menetapkannya secara otomatis.
- Batas retensi snapshot - Jumlah hari snapshot dipertahankan di Amazon S3. Misalnya, jika Anda menetapkan batas retensi ke 5, maka snapshot yang diambil hari ini dipertahankan selama 5 hari. Ketika batas retensi berakhir, snapshot secara otomatis dihapus.

  Batas retensi snapshot maksimum adalah 35 hari. Jika batas retensi snapshot disetel ke 0, snapshot otomatis dinonaktifkan untuk cluster. Data MemoryDB masih sepenuhnya tahan lama bahkan dengan snapshotting otomatis dinonaktifkan.

Anda dapat mengaktifkan atau menonaktifkan snapshot otomatis saat membuat cluster MemoryDB menggunakan konsol MemoryDB,, atau API MemoryDB. AWS CLI Anda dapat mengaktifkan snapshot otomatis saat membuat cluster MemoryDB dengan mencentang kotak Aktifkan Pencadangan Otomatis di bagian Snapshots. Untuk informasi lain, Membuat cluster MemoryDB.

Membuat snapshot manual

Selain snapshot otomatis, Anda dapat membuat snapshot manual kapan saja. Tidak seperti snapshot otomatis, yang secara otomatis dihapus setelah periode retensi tertentu, snapshot manual tidak memiliki periode retensi setelah itu dihapus secara otomatis. Anda harus menghapus snapshot manual secara manual. Bahkan jika Anda menghapus cluster atau node, snapshot manual apa pun dari cluster atau node tersebut tetap dipertahankan. Jika Anda tidak lagi ingin menyimpan snapshot manual, Anda harus menghapusnya sendiri secara eksplisit.

Snapshot manual berguna untuk pengujian dan pengarsipan. Sebagai contoh, misalkan Anda sudah mengembangkan sekumpulan data garis dasar untuk tujuan pengujian. Anda dapat membuat snapshot manual data dan mengembalikannya kapan pun Anda mau. Setelah menguji aplikasi yang memodifikasi data, Anda dapat mengatur ulang data dengan membuat cluster baru dan memulihkan dari snapshot dasar Anda. Ketika cluster siap, Anda dapat menguji aplikasi Anda terhadap data dasar lagi-dan ulangi proses ini sesering yang diperlukan.

Selain langsung membuat snapshot manual, Anda dapat membuat snapshot manual dengan salah satu cara berikut:

- Menyalin snapshot— Tidak masalah apakah snapshot sumber dibuat secara otomatis atau manual.
- Membuat snapshot akhir— Buat snapshot segera sebelum menghapus cluster.

Topik penting lainnya

- Kendala snapshot
- Biaya snapshot

Anda dapat membuat snapshot manual dari sebuah node menggunakan APIAWS Management Console, theAWS CLI, atau MemoryDB.

Membuat snapshot manual (Konsol)

Untuk membuat snapshot dari cluster (konsol)

1. Masuk ke AWS Management Console dan buka konsol MemoryDB for Redis di https://console.aws.amazon.com/memorydb/.

2. dari panel navigasi kiri, pilih Clusters.

   Layar cluster MemoryDB muncul.

3. pilih tombol radio di sebelah kiri nama cluster MemoryDB yang ingin Anda cadangkan.
4. Pilih Tindakan dan kemudian Ambil snapshot.
5. Di jendela Snapshot, ketik nama untuk snapshot Anda di Nama Snapshot kotak. Kami menyarankan agar nama tersebut menunjukkan cluster mana yang dicadangkan dan tanggal serta waktu snapshot dibuat.

   Kendala penamaan cluster 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 bisa diakhiri dengan tanda hubung.

6. Di bawah Enkripsi, pilih apakah akan menggunakan kunci enkripsi default atau kunci yang dikelola pelanggan. Untuk informasi selengkapnya, lihat Enkripsi dalam transit (TLS) di MemoryDB.
7. Di bawah Tag, tambahkan tag secara opsional untuk mencari dan memfilter snapshot Anda atau melacak biaya AndaAWS.
8. Pilih Ambil snapshot.

   Status klaster berubah menjadi membuat snapshot. Ketika status kembali tersedia, snapshot selesai.

Membuat snapshot manual (AWSCLI)

Untuk membuat snapshot manual cluster menggunakanAWS CLI, gunakan `create-snapshot` AWS CLI operasi dengan parameter berikut:

- `--cluster-name`— Nama cluster MemoryDB untuk digunakan sebagai sumber untuk snapshot. Gunakan parameter ini saat membuat cadangan cluster MemoryDB.

  Kendala penamaan cluster 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 bisa diakhiri dengan tanda hubung.

- `--snapshot-name`— Nama snapshot yang akan dibuat.

Topik terkait

Untuk informasi selengkapnya, lihat `create-snapshot` di Referensi Perintah AWS CLI.

Membuat snapshot manual (MemoryDB API)

Untuk membuat snapshot manual cluster menggunakan API MemoryDB, gunakan operasi API `CreateSnapshot` MemoryDB dengan parameter berikut:

- `ClusterName`— Nama cluster MemoryDB untuk digunakan sebagai sumber untuk snapshot. Gunakan parameter ini saat membuat cadangan cluster MemoryDB.

  Kendala penamaan cluster 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 bisa diakhiri dengan tanda hubung.
- `SnapshotName`— Nama snapshot yang akan dibuat.

Topik terkait

Untuk informasi lebih lanjut, lihat CreateSnapshot.

Membuat snapshot akhir

Anda dapat membuat snapshot akhir menggunakan konsol MemoryDB, APIAWS CLI, atau MemoryDB.

Membuat snapshot akhir (Konsol)

Anda dapat membuat snapshot akhir saat Anda menghapus cluster MemoryDB menggunakan konsol MemoryDB.

Untuk membuat snapshot akhir saat menghapus cluster MemoryDB, pada halaman hapus, pilih Ya dan beri nama snapshot di. Langkah 4: Menghapus cluster

Membuat snapshot akhir (AWSCLI)

Anda dapat membuat snapshot akhir saat menghapus cluster MemoryDB menggunakan file. AWS CLI

Saat menghapus cluster MemoryDB

Untuk membuat snapshot akhir saat menghapus cluster, gunakan `delete-cluster` AWS CLI operasi, dengan parameter berikut:

- `--cluster-name`— Nama cluster yang dihapus.
- `--final-snapshot-name`— Nama snapshot terakhir.

Kode berikut mengambil snapshot terakhir `bkup-20210515-final` saat menghapus cluster. `myCluster`

Untuk Linux, macOS, atau Unix:

```
aws memorydb delete-cluster \
        --cluster-name myCluster \
        --final-snapshot-name bkup-20210515-final
```

Untuk Windows:

```
aws memorydb delete-cluster ^
        --cluster-name myCluster ^
        --final-snapshot-name bkup-20210515-final
```

Untuk informasi selengkapnya, lihat menghapus-cluster di Referensi AWS CLI Perintah.

Membuat snapshot akhir (MemoryDB API)

Anda dapat membuat snapshot akhir saat menghapus cluster MemoryDB menggunakan API MemoryDB.

Saat menghapus cluster MemoryDB

Untuk membuat snapshot akhir, gunakan operasi API `DeleteCluster` MemoryDB dengan parameter berikut.

- `ClusterName`— Nama cluster yang dihapus.
- `FinalSnapshotName`— Nama snapshot.

Operasi API MemoryDB berikut membuat snapshot `bkup-20210515-final` saat menghapus cluster. `myCluster`

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=DeleteCluster
    &ClusterName=myCluster
    &FinalSnapshotName=bkup-20210515-final
    &Version=2021-01-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210515T192317Z
    &X-Amz-Credential=<credential>
```

Untuk informasi lebih lanjut, lihat DeleteCluster.

Menggambarkan snapshot

Prosedur berikut menunjukkan cara menampilkan daftar snapshot Anda. Jika mau, Anda juga dapat melihat detail snapshot tertentu.

Menjelaskan snapshot (Konsol)

Untuk menampilkan snapshot menggunakan AWS Management Console

1. Masuk ke konsol
2. dari panel navigasi kiri, pilih Snapshots.
3. Gunakan pencarian untuk memfilter secara manual, otomatis, atau semua snapshot.
4. Untuk melihat detail snapshot tertentu, pilih tombol radio di sebelah kiri nama snapshot. Pilih Tindakan dan kemudian Lihat detail.
5. Secara opsional, di halaman Lihat detail, Anda dapat melakukan tindakan snapshot tambahan seperti menyalin, memulihkan, atau menghapus. Anda juga dapat menambahkan tag ke snapshot

Menjelaskan snapshot (CLIAWS)

Untuk menampilkan daftar snapshot dan detail opsional tentang snapshot tertentu, gunakan operasi CLI`describe-snapshots`.

Contoh

Operasi berikut menggunakan parameter `--max-results` untuk mencantumkan hingga 20 snapshot yang terkait dengan akun Anda. Menghilangkan `--max-results` daftar parameter hingga 50 snapshot.

```
aws memorydb describe-snapshots --max-results 20
```

Operasi berikut menggunakan parameter `--cluster-name` untuk daftar hanya snapshot yang terkait dengan clustermy`-cluster`.

```
aws memorydb describe-snapshots --cluster-name my-cluster
```

Operasi berikut menggunakan parameter `--snapshot-name` untuk menampilkan detail snapshotmy`-snapshot`.

```
aws memorydb describe-snapshots --snapshot-name my-snapshot
```

Untuk informasi selengkapnya, lihat deskripsi-snapshot.

Menjelaskan snapshot (MemoryDB API)

Untuk menampilkan daftar snapshot, gunakan `DescribeSnapshots` operasi.

Contoh

Operasi berikut menggunakan parameter `MaxResults` untuk mencantumkan hingga 20 snapshot yang terkait dengan akun Anda. Menghilangkan `MaxResults` daftar parameter hingga 50 snapshot.

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=DescribeSnapshots
    &MaxResults=20
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Timestamp=20210801T220302Z
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Date=20210801T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

Operasi berikut menggunakan parameter `ClusterName` untuk daftar semua snapshot yang terkait dengan cluster`MyCluster`.

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=DescribeSnapshots
    &ClusterName=MyCluster
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Timestamp=20210801T220302Z
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Date=20210801T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Credential=<credential>
```

```
    &X-Amz-Signature=<signature>
```

Operasi berikut menggunakan parameter `SnapshotName` untuk menampilkan detail untuk snapshot`MyBackup`.

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=DescribeSnapshots
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &SnapshotName=MyBackup
    &Timestamp=20210801T220302Z
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Date=20210801T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

Untuk informasi lebih lanjut, lihat DescribeSnapshots.

Menyalin snapshot

Anda dapat membuat salinan snapshot apa pun, apakah itu dibuat secara otomatis atau manual. Saat menyalin snapshot, kunci enkripsi KMS yang sama dengan sumber digunakan untuk target kecuali secara khusus diganti. Anda juga dapat mengekspor snapshot Anda sehingga Anda dapat mengaksesnya dari luar MemoryDB. Untuk panduan tentang mengekspor snapshot Anda, lihat. Mengekspor snapshot

Prosedur berikut menunjukkan cara menyalin snapshot.

Menyalin snapshot (Konsol)

Untuk menyalin snapshot (konsol)

1. Masuk ke AWS Management Console dan buka konsol MemoryDB for Redis di https://console.aws.amazon.com/memorydb/.
2. Untuk melihat daftar snapshot Anda, dari panel navigasi kiri pilih Snapshots.
3. Dari daftar snapshot, pilih tombol radio di sebelah kiri nama snapshot yang ingin Anda salin.
4. Pilih Tindakan dan kemudian pilih Salin.
5. Di halaman Salin snapshot, lakukan hal berikut:

    a. Di kotak Nama snapshot baru, ketikkan nama untuk snapshot baru Anda.

    b. Biarkan kotak opsional Bucket S3 Target kosong. Bidang ini hanya boleh digunakan untuk mengekspor snapshot Anda dan memerlukan izin S3 khusus. Untuk informasi tentang mengekspor snapshot, lihat. Mengekspor snapshot

    c. Pilih apakah akan menggunakan kunci AWS KMS enkripsi default atau menggunakan kunci khusus. Untuk informasi selengkapnya, lihat Enkripsi dalam transit (TLS) di MemoryDB.

    d. Secara opsional, Anda juga dapat menambahkan tag ke salinan snapshot.

    e. Pilih Salin.

Menyalin snapshot (CLIAWS)

Untuk menyalin snapshot, gunakan `copy-snapshot` operasi.

Parameter

- `--source-snapshot-name`— Nama snapshot yang akan disalin.

- `--target-snapshot-name`— Nama salinan snapshot.
- `--target-bucket`— Dicadangkan untuk mengekspor snapshot. Jangan gunakan parameter ini saat membuat salinan snapshot. Untuk informasi selengkapnya, lihat Mengekspor snapshot.

Contoh berikut membuat salinan snapshot otomatis.

Untuk Linux, macOS, atau Unix:

```
aws memorydb copy-snapshot \
    --source-snapshot-name automatic.my-primary-2021-03-27-03-15 \
    --target-snapshot-name my-snapshot-copy
```

Untuk Windows:

```
aws memorydb copy-snapshot ^
    --source-snapshot-name automatic.my-primary-2021-03-27-03-15 ^
    --target-snapshot-name my-snapshot-copy
```

Untuk informasi selengkapnya, lihat copy-snapshot.

Menyalin snapshot (MemoryDB API)

Untuk menyalin snapshot, gunakan `copy-snapshot` operasi dengan parameter berikut:

Parameter

- `SourceSnapshotName`— Nama snapshot yang akan disalin.
- `TargetSnapshotName`— Nama salinan snapshot.
- `TargetBucket`— Dicadangkan untuk mengekspor snapshot. Jangan gunakan parameter ini saat membuat salinan snapshot. Untuk informasi selengkapnya, lihat Mengekspor snapshot.

Contoh berikut membuat salinan snapshot otomatis.

Example

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=CopySnapshot
    &SourceSnapshotName=automatic.my-primary-2021-03-27-03-15
    &TargetSnapshotName=my-snapshot-copy
```

```
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210801T220302Z
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Date=20210801T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

Untuk informasi lebih lanjut, lihat CopySnapshot.

Mengekspor snapshot

MemoryDB for Redis mendukung ekspor snapshot MemoryDB Anda ke bucket Amazon Simple Storage Service (Amazon S3), yang memberi Anda akses ke sana dari luar MemoryDB. Snapshot MemoryDB yang diekspor sepenuhnya sesuai dengan Redis open-source dan dapat dimuat dengan versi atau perkakas Redis yang sesuai. Anda dapat mengekspor snapshot menggunakan konsol MemoryDB, APIAWS CLI, atau MemoryDB.

Mengekspor snapshot dapat membantu jika Anda perlu meluncurkan cluster di Wilayah lainAWS. Anda dapat mengekspor data Anda dalam satu Wilayah AWS, menyalin file .rdb ke Wilayah AWS baru, dan kemudian menggunakan file .rdb tersebut untuk menyemai klaster baru, alih-alih menunggu klaster baru mengisi data melalui penggunaan. Untuk informasi tentang menyemai klaster baru, lihat Menyemai cluster baru dengan snapshot yang dibuat secara eksternal. Alasan lain Anda ingin mengekspor data dari klaster Anda adalah untuk menggunakan file .rdb untuk pemrosesan offline.

Important

- Snapshot MemoryDB dan bucket Amazon S3 yang ingin Anda salin harus berada di Wilayah yang sama. AWS

  Meskipun snapshot yang disalin ke bucket Amazon S3 dienkripsi, kami sangat menyarankan agar Anda tidak memberi orang lain akses ke bucket Amazon S3 tempat Anda ingin menyimpan snapshot Anda.
- Mengekspor snapshot ke Amazon S3 tidak didukung untuk cluster yang menggunakan tiering data. Untuk informasi selengkapnya, lihat Tingkatan data.

Sebelum dapat mengekspor snapshot ke bucket Amazon S3, Anda harus memiliki bucket Amazon S3 di Wilayah yang AWS sama dengan snapshot. Berikan akses MemoryDB ke ember. Dua langkah pertama menunjukkan cara melakukannya.

Warning

Skenario berikut mengungkapkan data Anda dengan cara yang mungkin tidak Anda inginkan:

- Ketika orang lain memiliki akses ke bucket Amazon S3 tempat Anda mengekspor snapshot Anda.

Untuk mengontrol akses ke snapshot Anda, hanya izinkan akses ke bucket Amazon S3 kepada mereka yang ingin mengakses data Anda. Untuk informasi tentang mengelola akses ke bucket Amazon S3, lihat Mengelola akses di Panduan Pengembang Amazon S3.

- Ketika orang lain memiliki izin untuk menggunakan operasi CopySnapshot API.

  Pengguna atau grup yang memiliki izin untuk menggunakan operasi `CopySnapshot` API dapat membuat bucket Amazon S3 mereka sendiri dan menyalin snapshot ke mereka. Untuk mengontrol akses ke snapshot Anda, gunakan kebijakan AWS Identity and Access Management (IAM) untuk mengontrol siapa yang memiliki kemampuan untuk menggunakan API. `CopySnapshot` Untuk informasi selengkapnya tentang penggunaan IAM untuk mengontrol penggunaan operasi API MemoryDB, lihat Manajemen identitas dan akses di MemoryDB untuk Redis di Panduan Pengguna MemoryDB.

Topik

Langkah 1: Buat ember Amazon S3

Prosedur berikut menggunakan konsol Amazon S3 untuk membuat bucket Amazon S3 tempat Anda mengekspor dan menyimpan snapshot MemoryDB Anda.

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 Membuat Bucket.
3. Pada Membuat Bucket - Pilih Nama Bucket dan Wilayah, lakukan hal berikut:

   a. Di Bucket Name, ketikkan nama untuk bucket Amazon S3 Anda.

   b. Dari daftar Wilayah, pilih AWS Wilayah untuk bucket Amazon S3 Anda. AWSWilayah ini harus AWS Wilayah yang sama dengan snapshot MemoryDB yang ingin Anda ekspor.

   c. Pilih Buat.

Untuk informasi selengkapnya tentang membuat bucket Amazon S3, lihat Membuat bucket di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Langkah 2: Berikan akses MemoryDB ke bucket Amazon S3 Anda

Wilayah AWS yang diperkenalkan sebelum 20 Maret 2019, diaktifkan secara default. Anda dapat mulai bekerja di Wilayah AWS ini dengan segera. Wilayah yang diperkenalkan setelah 20 Maret 2019 dinonaktifkan secara default. Anda harus mengaktifkan, atau ikut serta, ke Wilayah ini sebelum Anda dapat menggunakannya, seperti yang dijelaskan dalam Mengelola AWS wilayah.

Berikan akses MemoryDB ke Bucket S3 Anda di suatu Wilayah AWS

Untuk membuat izin yang tepat pada bucket Amazon S3 di AWS Wilayah, lakukan langkah-langkah berikut.

Untuk memberikan akses MemoryDB 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 ingin Anda salin snapshot. Ini seharusnya adalah bucket S3 yang Anda buat di Langkah 1: Buat ember Amazon S3.
3. Pilih tab Izin dan di bawah Izin, pilih Kebijakan Bucket.
4. Perbarui kebijakan untuk memberikan izin yang diperlukan MemoryDB untuk melakukan operasi:

   - Tambahkan `[ "Service" : "region-full-name.memorydb-snapshot.amazonaws.com" ]` ke `Principal`.
   - Tambahkan izin berikut yang diperlukan untuk mengekspor snapshot ke bucket Amazon S3.
     - `"s3:PutObject"`
     - `"s3:GetObject"`
     - `"s3:ListBucket"`
     - `"s3:GetBucketAcl"`
     - `"s3:ListMultipartUploadParts"`
     - `"s3:ListBucketMultipartUploads"`

   Berikut adalah contoh tampilan kebijakan yang sudah diperbarui.

```
{
```

```
    "Version": "2012-10-17",
    "Id": "Policy15397346",
    "Statement": [
        {
            "Sid": "Stmt15399483",
            "Effect": "Allow",
            "Principal": {
                "Service": "aws-region.memorydb-snapshot.amazonaws.com"
            },
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:GetBucketAcl",
                "s3:ListMultipartUploadParts",
                "s3:ListBucketMultipartUploads"
            ],
            "Resource": [
                "arn:aws:s3:::example-bucket",
                "arn:aws:s3:::example-bucket/*"
            ]
        }
    ]
}
```

Langkah 3: Ekspor snapshot MemoryDB

Sekarang Anda telah membuat bucket S3 Anda dan memberikan izin MemoryDB untuk mengaksesnya. Ubah Kepemilikan Objek S3 menjadi ACL diaktifkan - Pemilik bucket lebih disukai. Selanjutnya, Anda dapat menggunakan konsol MemoryDB, AWS CLI, atau API MemoryDB untuk mengekspor snapshot Anda ke sana. Berikut ini mengasumsikan bahwa Anda 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:::*"
  }]
}
```

Mengekspor snapshot MemoryDB (Konsol)

Proses berikut menggunakan konsol MemoryDB untuk mengekspor snapshot ke bucket Amazon S3 sehingga Anda dapat mengaksesnya dari luar MemoryDB. Bucket Amazon S3 harus berada di AWS Wilayah yang sama dengan snapshot MemoryDB.

Untuk mengekspor snapshot MemoryDB ke bucket Amazon S3

1. Masuk ke AWS Management Console dan buka konsol MemoryDB for Redis di https://console.aws.amazon.com/memorydb/.
2. Untuk melihat daftar snapshot Anda, dari panel navigasi kiri pilih Snapshots.
3. Dari daftar snapshot, pilih tombol radio di sebelah kiri nama snapshot yang ingin Anda ekspor.
4. Pilih Salin.
5. Pada Buat Salinan Backup?, lakukan hal berikut:

    a. Di kotak nama snapshot baru, ketikkan nama untuk snapshot baru Anda.

    Nama itu harus di antara 1 dan 1.000 karakter serta dapat dienkode menjadi UTF-8.

    MemoryDB menambahkan pengenal pecahan dan nilai yang `.rdb` Anda masukkan di sini. Misalnya, jika Anda masuk`my-exported-snapshot`, MemoryDB menciptakan. `my-exported-snapshot-0001.rdb`

    b. Dari daftar Lokasi Target S3, pilih nama bucket Amazon S3 yang ingin Anda salin snapshot (bucket yang Anda buat). Langkah 1: Buat ember Amazon S3

    Lokasi Target S3 harus berupa bucket Amazon S3 di Wilayah snapshot AWS dengan izin berikut agar proses ekspor berhasil.

    - Akses objek — Baca dan Tulis.
    - Akses izin — Baca.

Untuk informasi selengkapnya, lihat Langkah 2: Berikan akses MemoryDB ke bucket Amazon S3 Anda.

c. Pilih Salin.

> **Note**
>
> Jika bucket S3 Anda tidak memiliki izin yang diperlukan untuk MemoryDB untuk mengekspor snapshot ke sana, Anda menerima salah satu pesan galat berikut. Kembali ke Langkah 2: Berikan akses MemoryDB ke bucket Amazon S3 Anda untuk menambahkan izin yang ditentukan dan coba lagi mengekspor snapshot Anda.
>
> - MemoryDB belum diberikan izin BACA %s pada Bucket S3.
>
> Solusi: Tambahkan izin Baca pada bucket.
> - MemoryDB belum diberikan izin WRITE %s pada Bucket S3.
>
> Solusi: Tambahkan izin Tulis pada bucket.
> - MemoryDB belum diberikan izin READ_ACP %s pada Bucket S3.
>
> Solusi: Tambahkan Baca untuk akses Izin pada bucket.

Jika Anda ingin menyalin snapshot Anda ke AWS Wilayah lain, gunakan Amazon S3 untuk menyalinnya. Untuk informasi selengkapnya, lihat Menyalin objek di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Mengekspor snapshot MemoryDB (CLI) AWS

Ekspor snapshot ke bucket Amazon S3 menggunakan operasi `copy-snapshot` CLI dengan parameter berikut:

Parameter

- `--source-snapshot-name`— Nama snapshot yang akan disalin.
- `--target-snapshot-name`— Nama salinan snapshot.

  Nama itu harus di antara 1 dan 1.000 karakter serta dapat dienkode menjadi UTF-8.

MemoryDB menambahkan pengenal pecahan dan nilai yang `.rdb` Anda masukkan di sini. Misalnya, jika Anda masuk`my-exported-snapshot`, MemoryDB menciptakan. `my-exported-snapshot-0001.rdb`

- `--target-bucket`— Nama bucket Amazon S3 tempat Anda ingin mengekspor snapshot. Salinan snapshot dibuat di ember yang ditentukan.

  `--target-bucket`Harus berupa bucket Amazon S3 di AWS Wilayah snapshot dengan izin berikut agar proses ekspor berhasil.

  - Akses objek — Baca dan Tulis.
  - Akses izin — Baca.

  Untuk informasi selengkapnya, lihat Langkah 2: Berikan akses MemoryDB ke bucket Amazon S3 Anda.

Operasi berikut menyalin snapshot ke my-s3-bucket.

Untuk Linux, macOS, atau Unix:

```
aws memorydb copy-snapshot \
    --source-snapshot-name automatic.my-primary-2021-06-27-03-15 \
    --target-snapshot-name my-exported-snapshot \
    --target-bucket my-s3-bucket
```

Untuk Windows:

```
aws memorydb copy-snapshot ^
    --source-snapshot-name automatic.my-primary-2021-06-27-03-15 ^
    --target-snapshot-name my-exported-snapshot ^
    --target-bucket my-s3-bucket
```

> **Note**
>
> Jika bucket S3 Anda tidak memiliki izin yang diperlukan untuk MemoryDB untuk mengekspor snapshot ke sana, Anda menerima salah satu pesan galat berikut. Kembali ke Langkah 2: Berikan akses MemoryDB ke bucket Amazon S3 Anda untuk menambahkan izin yang ditentukan dan coba lagi mengekspor snapshot Anda.
>
> - MemoryDB belum diberikan izin BACA %s pada Bucket S3.

Solusi: Tambahkan izin Baca pada bucket.

- MemoryDB belum diberikan izin WRITE %s pada Bucket S3.

  Solusi: Tambahkan izin Tulis pada bucket.

- MemoryDB belum diberikan izin READ_ACP %s pada Bucket S3.

  Solusi: Tambahkan Baca untuk akses Izin pada bucket.

Untuk informasi selengkapnya, lihat `copy-snapshot` di Referensi Perintah AWS CLI.

Jika Anda ingin menyalin snapshot Anda ke AWS Wilayah lain, gunakan salinan Amazon S3. Untuk informasi selengkapnya, lihat Menyalin objek di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Mengekspor snapshot MemoryDB (MemoryDB API)

Ekspor snapshot ke bucket Amazon S3 menggunakan operasi API dengan parameter `CopySnapshot` ini.

Parameter

- `SourceSnapshotName`— Nama snapshot yang akan disalin.
- `TargetSnapshotName`— Nama salinan snapshot.

  Nama itu harus di antara 1 dan 1.000 karakter serta dapat dienkode menjadi UTF-8.

  MemoryDB menambahkan pengenal pecahan dan nilai yang `.rdb` Anda masukkan di sini. Misalnya, jika Anda memasukkan `my-exported-snapshot`, Anda memperoleh `my-exported-snapshot-0001.rdb`.

- `TargetBucket`— Nama bucket Amazon S3 tempat Anda ingin mengekspor snapshot. Salinan snapshot dibuat di ember yang ditentukan.

  `TargetBucket`Harus berupa bucket Amazon S3 di AWS Wilayah snapshot dengan izin berikut agar proses ekspor berhasil.

  - Akses objek — Baca dan Tulis.
  - Akses izin — Baca.

Untuk informasi selengkapnya, lihat Langkah 2: Berikan akses MemoryDB ke bucket Amazon S3 Anda.

Contoh berikut membuat salinan snapshot otomatis ke bucket Amazon `my-s3-bucket` S3.

Example

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=CopySnapshot
    &SourceSnapshotName=automatic.my-primary-2021-06-27-03-15
    &TargetBucket=my-s3-bucket
    &TargetSnapshotName=my-snapshot-copy
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210801T220302Z
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Date=20210801T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

Note

Jika bucket S3 Anda tidak memiliki izin yang diperlukan untuk MemoryDB untuk mengekspor snapshot ke sana, Anda menerima salah satu pesan galat berikut. Kembali ke Langkah 2: Berikan akses MemoryDB ke bucket Amazon S3 Anda untuk menambahkan izin yang ditentukan dan coba lagi mengekspor snapshot Anda.

- MemoryDB belum diberikan izin BACA %s pada Bucket S3.

  Solusi: Tambahkan izin Baca pada bucket.
- MemoryDB belum diberikan izin WRITE %s pada Bucket S3.

  Solusi: Tambahkan izin Tulis pada bucket.
- MemoryDB belum diberikan izin READ_ACP %s pada Bucket S3.

  Solusi: Tambahkan Baca untuk akses Izin pada bucket.

Untuk informasi lebih lanjut, lihat CopySnapshot.

Jika Anda ingin menyalin snapshot Anda ke AWS Wilayah lain, gunakan salinan Amazon S3 untuk menyalin snapshot yang diekspor ke bucket Amazon S3 di Wilayah lain. AWS Untuk informasi selengkapnya, lihat Menyalin objek di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Memulihkan dari snapshot

Anda dapat memulihkan data dari MemoryDB atau ElastiCache untuk file snapshot Redis .rdb ke cluster baru kapan saja.

MemoryDB untuk proses pemulihan Redis mendukung hal berikut:

- Bermigrasi dari satu atau beberapa file snapshot .rdb yang Anda buat dari Redis ke cluster ElastiCache MemoryDB.

  File .rdb harus dimasukkan ke dalam S3 untuk melakukan pemulihan.
- Menentukan sejumlah pecahan di cluster baru yang berbeda dari jumlah pecahan di cluster yang digunakan untuk membuat file snapshot.
- Menentukan jenis node yang berbeda untuk cluster baru—lebih besar atau lebih kecil. Jika melakukan penskalaan untuk jenis simpul yang lebih kecil, pastikan agar jenis simpul baru memiliki memori yang cukup untuk data Anda dan overhead dari Redis.
- Mengkonfigurasi slot cluster MemoryDB baru secara berbeda dari pada cluster yang digunakan untuk membuat file snapshot.

> **Important**
>
> - Cluster MemoryDB tidak mendukung banyak database. Oleh karena itu, saat memulihkan ke MemoryDB pemulihan Anda gagal jika file.rdb mereferensikan lebih dari satu database.
> - Anda tidak dapat mengembalikan snapshot dari cluster yang menggunakan tiering data (misalnya, tipe node r6gd) ke dalam cluster yang tidak menggunakan tiering data (misalnya, tipe node r6g).

Apakah Anda membuat perubahan saat memulihkan cluster dari snapshot diatur oleh pilihan yang Anda buat. Anda membuat pilihan ini di halaman Restore Cluster saat menggunakan konsol MemoryDB untuk memulihkan. Anda membuat pilihan ini dengan menetapkan nilai parameter saat menggunakan AWS CLI atau MemoryDB API untuk memulihkan.

Selama operasi pemulihan, MemoryDB membuat cluster baru, dan kemudian mengisinya dengan data dari file snapshot. Ketika proses ini selesai, cluster dihangatkan dan siap menerima permintaan.

> **Important**
>
> Sebelum Anda melanjutkan, pastikan Anda telah membuat snapshot dari cluster yang ingin Anda pulihkan. Untuk informasi selengkapnya, lihat Membuat snapshot manual.
> Jika Anda ingin memulihkan dari snapshot yang dibuat secara eksternal, lihat. Menyemai cluster baru dengan snapshot yang dibuat secara eksternal

Prosedur berikut menunjukkan cara mengembalikan snapshot ke cluster baru menggunakan konsol MemoryDB, APIAWS CLI, atau MemoryDB.

Memulihkan dari snapshot (Konsol)

Untuk mengembalikan snapshot ke cluster baru (konsol)

1. Masuk ke AWS Management Console dan buka konsol MemoryDB for Redis di https://console.aws.amazon.com/memorydb/.
2. Pada panel navigasi, pilih Snapshots.
3. Dalam daftar snapshot, pilih tombol di sebelah nama nama snapshot yang ingin Anda pulihkan.
4. Pilih Tindakan dan kemudian pilih Pulihkan
5. Di bawah konfigurasi Cluster, masukkan yang berikut ini:

   a. Nama cluster - Diperlukan. Nama klaster baru.

   b. Deskripsi - Opsional. Deskripsi cluster baru.
6. Lengkapi bagian grup Subnet:

   - Untuk grup Subnet, buat grup subnet baru atau pilih yang sudah ada dari daftar yang tersedia yang ingin Anda terapkan ke cluster ini. Jika Anda membuat yang baru:

     - Masukkan Nama
     - Masukkan Deskripsi
     - Jika Anda mengaktifkan Multi-AZ, grup subnet harus berisi setidaknya dua subnet yang berada dalam availability zone yang berbeda. Untuk informasi selengkapnya, lihat Subnet dan grup subnet.
     - Jika Anda membuat grup subnet baru dan tidak memiliki VPC yang ada, Anda akan diminta untuk membuat VPC. Untuk informasi lebih lanjut, lihat Apa itu Amazon VPC? di Panduan Pengguna Amazon VPC.

7. Lengkapi bagian Pengaturan Cluster:

   a. Untuk kompatibilitas versi Redis, terima default`6.0`.

   b. Untuk Port, terima port Redis default 6379 atau, jika Anda memiliki alasan untuk menggunakan port yang berbeda, masukkan nomor port..

   c. Untuk grup Parameter, terima grup `default.memorydb-redis6` parameter.

      Grup parameter mengontrol parameter waktu aktif dari klaster Anda. Untuk informasi selengkapnya tentang grup parameter, lihat Parameter spesifik Redis.

   d. Untuk tipe Node, pilih nilai untuk tipe node (bersama dengan ukuran memori terkait) yang Anda inginkan.

      Jika Anda memilih anggota keluarga tipe node r6gd, Anda akan secara otomatis mengaktifkan data-tiering di cluster Anda. Untuk informasi selengkapnya, lihat Tingkatan data.

   e. Untuk Jumlah pecahan, pilih jumlah pecahan yang Anda inginkan untuk cluster ini.

      Anda dapat mengubah jumlah pecahan di cluster Anda secara dinamis. Untuk informasi selengkapnya, lihat Menskalakan klaster MemoryDB.

   f. Untuk Replika per serpihan, pilih jumlah simpul replika baca yang Anda inginkan dalam setiap serpihan.

      Pembatasan berikut ada;.

      - Jika Anda mengaktifkan Multi-AZ, pastikan bahwa Anda memiliki setidaknya satu replika per serpihan.
      - Jumlah replika adalah sama untuk setiap serpihan saat membuat klaster menggunakan konsol.

   g. Pilih Selanjutnya

   h. Lengkapi bagian Pengaturan lanjutan:

      i. 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 selengkapnya tentang grup keamanan, lihat Grup keamanan untuk VPC Anda di Panduan Pengguna Amazon VPC.

ii. Data dienkripsi dengan cara berikut:

- Enkripsi saat istirahat - Memungkinkan enkripsi data yang disimpan pada disk. Untuk informasi lain, lihat Enkripsi at Rest.

> **Note**
>
> Anda memiliki opsi untuk menyediakan kunci enkripsi yang berbeda dengan memilih kunci AWSKMS yang Dikelola Pelanggan dan memilih kunci.

- Enkripsi dalam transit — Memungkinkan enkripsi data pada kabel. Ini diaktifkan secara default. Untuk informasi lain, lihat Enkripsi in transit.

Jika Anda memilih tidak ada enkripsi, maka daftar kontrol Akses terbuka yang disebut “akses terbuka” akan dibuat dengan pengguna default. Untuk informasi selengkapnya, lihat Mengautentikasi pengguna dengan Daftar Kontrol Akses (ACL).

iii. Untuk Snapshot secara opsional, tentukan periode retensi snapshot dan jendela snapshot. Secara default, Aktifkan snapshot otomatis dipilih.

iv. Untuk jendela Pemeliharaan opsional menentukan jendela pemeliharaan. Jendela pemeliharaan adalah waktu, umumnya satu jam panjangnya, setiap minggu ketika MemoryDB menjadwalkan pemeliharaan sistem untuk cluster Anda. Anda dapat mengizinkan MemoryDB 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 jendela Tentukan pemeliharaan dari daftar, pilih Hari mulai, Waktu mulai, dan Durasi (dalam jam) untuk jendela pemeliharaan Anda. Semua waktu adalah waktu UTC.

Untuk informasi selengkapnya, lihat Mengelola pemeliharaan.

v. Untuk Pemberitahuan, pilih topik Amazon Simple Notification Service (Amazon SNS) yang ada, atau pilih Input ARN Manual dan masukkan Nama Sumber Daya Amazon (ARN) topik. Amazon SNS memungkinkan Anda untuk mendorong pemberitahuan ke perangkat pintar yang terhubung ke Internet. Defaultnya adalah menonaktifkan notifikasi. Untuk informasi lebih lanjut, lihat https://aws.amazon.com/sns/.

i. Untuk Tag, Anda dapat menerapkan tag secara opsional untuk mencari dan memfilter cluster Anda atau melacak biaya AndaAWS.

j. Tinjau semua entri dan pilihan Anda, lalu lakukan koreksi yang diperlukan. Saat Anda siap, pilih Buat klaster untuk meluncurkan klaster Anda, atau Batalkan untuk membatalkan operasi.

Segera setelah status klaster Anda tersedia, Anda dapat memberikan akses ke klaster kepada EC2, menyambung ke klaster itu, dan mulai menggunakannya. Untuk informasi lain, lihat Langkah 2: Otorisasi akses ke cluster dan Langkah 3: Connect ke cluster.

> **Important**
>
> Segera setelah klaster Anda tersedia, Anda akan ditagih untuk setiap jam atau sebagian jam saat klaster aktif, meskipun Anda tidak sedang aktif menggunakannya. Untuk berhenti dikenakan biaya atas klaster ini, Anda harus menghapusnya. Lihat Langkah 4: Menghapus cluster.

Memulihkan dari snapshot (CLIAWS)

Saat menggunakan salah satu `create-cluster` operasi, pastikan untuk menyertakan parameter `--snapshot-name` atau `--snapshot-arns` menyemai cluster baru dengan data dari snapshot.

Untuk informasi selengkapnya, lihat berikut ini:

- Membuat klaster AWS (CLI)di Panduan Pengguna MemoryDB.
- buat-cluster di Command Reference. AWS CLI

Memulihkan dari snapshot (MemoryDB API)

Anda dapat memulihkan snapshot MemoryDB menggunakan operasi API MemoryDB. `CreateCluster`

Saat menggunakan `CreateCluster` operasi, pastikan untuk menyertakan parameter `SnapshotName` atau `SnapshotArns` menyemai cluster baru dengan data dari snapshot.

Untuk informasi selengkapnya, lihat berikut ini:

- Membuat cluster (MemoryDB API)di Panduan Pengguna MemoryDB.
- CreateClusterdi Referensi API MemoryDB.

Menyemai cluster baru dengan snapshot yang dibuat secara eksternal

Saat Anda membuat cluster MemoryDB baru, Anda dapat menyemai dengan data dari file snapshot Redis .rdb.

Untuk menyemai cluster MemoryDB baru dari snapshot MemoryDB atau ElastiCache untuk snapshot Redis, lihat. Memulihkan dari snapshot

Saat Anda menggunakan file Redis .rdb untuk menyemai cluster MemoryDB baru, Anda dapat melakukan hal berikut:

- Tentukan sejumlah pecahan di cluster baru. Angka ini bisa berbeda dari jumlah pecahan di cluster yang digunakan untuk membuat file snapshot.
- Tentukan jenis node yang berbeda untuk cluster baru—lebih besar atau lebih kecil dari yang digunakan dalam cluster yang membuat snapshot. Jika menskalakan ke jenis simpul yang lebih kecil, pastikan bahwa jenis simpul baru memiliki memori yang cukup untuk data Anda dan overhead dari Redis.

Important

- Anda harus memastikan bahwa data snapshot Anda tidak melebihi sumber daya node.

  Jika snapshot terlalu besar, cluster yang dihasilkan memiliki status. `restore-failed` Jika hal ini terjadi, Anda harus menghapus klaster itu dan memulai dari awal.

  Untuk daftar lengkap jenis dan spesifikasi node, lihatParameter spesifik tipe node MemoryDB.
- Anda dapat mengenkripsi file Redis .rdb dengan enkripsi sisi server Amazon S3 (SSE-S3) saja. Untuk informasi selengkapnya, silakan lihat Melindungi data menggunakan enkripsi sisi server.

Langkah 1: Buat snapshot redis di cluster eksternal

Untuk membuat snapshot untuk menyemai cluster MemoryDB Anda

1. Menyambung ke instans Redis yang telah ada.

2. Jalankan Redis BGSAVE atau SAVE operasi untuk membuat snapshot. Perhatikan di mana file .rdb Anda berada.

   BGSAVE bersifat asinkron dan tidak memblokir klien lain saat melakukan pemrosesan. Untuk informasi lain, lihat BGSAVE di situs web Redis.

   SAVE bersifat sinkron dan memblokir proses lainnya hingga selesai. Untuk informasi lain, lihat SAVE di situs web Redis.

Untuk informasi tambahan tentang membuat snapshot, lihat Ketekunan Redis di situs web Redis.

Langkah 2: Buat bucket dan folder Amazon S3

Ketika Anda telah membuat file snapshot, Anda perlu mengunggahnya ke folder dalam ember Amazon S3. Untuk melakukan itu, Anda harus terlebih dahulu memiliki ember dan folder Amazon S3 di dalam ember itu. Jika Anda sudah memiliki bucket dan folder Amazon S3 dengan izin yang sesuai, Anda dapat melompat ke. Langkah 3: Unggah snapshot 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 di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

   Nama bucket Amazon S3 Anda harus sesuai dengan DNS. Jika tidak, MemoryDB tidak dapat mengakses file cadangan Anda. Aturan untuk kepatuhan DNS adalah:

   - Nama harus minimal 3 dan panjangnya tidak lebih dari 63 karakter.
   - Nama harus berupa serangkaian satu atau lebih label yang dipisahkan oleh periode (.) di mana setiap label:
     - Dimulai dengan huruf kecil atau angka.
     - Berakhir dengan huruf kecil atau angka.
     - Berisi hanya huruf kecil, angka, dan tanda hubung.
   - Nama tidak dapat diformat sebagai alamat IP (misalnya, 192.0.2.0).

Kami sangat menyarankan Anda membuat bucket Amazon S3 Anda di AWS Wilayah yang sama dengan cluster MemoryDB baru Anda. Pendekatan ini memastikan bahwa kecepatan transfer data tertinggi saat MemoryDB 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 mengizinkan bucket dan isinya digunakan untuk menyemai cluster MemoryDB 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 Membuat folder.
4. Masukkan nama untuk folder baru Anda.
5. Pilih Simpan.

   Catat nama dari bucket dan folder.

Langkah 3: Unggah snapshot Anda ke Amazon S3

Sekarang saatnya mengunggah file .rdb yang Anda buat di Langkah 1: Buat snapshot redis di cluster eksternal. Anda mengunggahnya ke bucket Amazon S3 dan folder yang Anda buat. Langkah 2: Buat bucket dan folder Amazon S3 Untuk informasi selengkapnya tentang tugas ini, lihat Mengunggah objek. 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 Upload (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 file atau file yang benar tercantum di halaman 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 cluster baru dengan data dalam snapshot ini.

Untuk informasi tambahan, lihat Aturan penamaan bucket di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Langkah 4: Berikan akses baca MemoryDB ke file.rdb

Wilayah AWS yang diperkenalkan sebelum 20 Maret 2019, diaktifkan secara default. Anda dapat mulai bekerja di Wilayah AWS ini dengan segera. Wilayah yang diperkenalkan setelah 20 Maret 2019 dinonaktifkan secara default. Anda harus mengaktifkan, atau ikut serta, ke Wilayah ini sebelum dapat menggunakannya, seperti yang dijelaskan dalam Mengelola AWS wilayah.

Berikan akses baca MemoryDB ke file.rdb

Untuk memberikan akses baca MemoryDB ke file snapshot

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 snapshot .rdb Anda. Nama file yang terpilih muncul di atas tab di bagian atas halaman.
5. Pilih tab Izin.
6. Di bawah Izin, pilih Kebijakan Bucket, lalu pilih Edit.
7. Perbarui kebijakan untuk memberikan izin yang diperlukan MemoryDB untuk melakukan operasi:
   - Tambahkan `[ "Service" : "`*`region-full-name`*`.memorydb-snapshot.amazonaws.com" ]` ke `Principal`.

- Tambahkan izin berikut yang diperlukan 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": "us-east-1.memorydb-snapshot.amazonaws.com"
            },
            "Action": [
                "s3:GetObject",
                "s3:ListBucket",
                "s3:GetBucketAcl"
            ],
            "Resource": [
                "arn:aws:s3:::example-bucket",
                "arn:aws:s3:::example-bucket/snapshot1.rdb",
                "arn:aws:s3:::example-bucket/snapshot2.rdb"
            ]
        }
    ]
}
```

8. Pilih Simpan.

Langkah 5: Benih cluster MemoryDB dengan data file.rdb

Sekarang Anda siap untuk membuat cluster MemoryDB dan menyemai dengan data dari file.rdb. Untuk membuat cluster, ikuti petunjuk diMembuat cluster MemoryDB.

Metode yang Anda gunakan untuk memberi tahu MemoryDB di mana menemukan snapshot Redis yang Anda unggah ke Amazon S3 bergantung pada metode yang Anda gunakan untuk membuat cluster:

Benih cluster MemoryDB dengan data file.rdb

- Menggunakan konsol MemoryDB

  Setelah Anda memilih mesin Redis, luaskan bagian Pengaturan Redis lanjutan dan cari Impor data ke klaster. Di kotak lokasi file S3 Seed RDB, ketik jalur Amazon S3 untuk file. Jika Anda memiliki beberapa file .rdb, ketik 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-cluster` atau `create-cluster`, gunakan parameter `--snapshot-arns` untuk menentukan ARN yang memenuhi syarat sepenuhnya untuk setiap file .rdb. Sebagai contoh, `arn:aws:s3:::`*myBucket*/*myFolder*/*myBackupFilename*`.rdb`. ARN harus menyelesaikan file snapshot yang Anda simpan di Amazon S3.
- Menggunakan MemoryDB API

  Jika Anda menggunakan `CreateCluster` atau operasi API `CreateCluster` MemoryDB, gunakan parameter `SnapshotArns` untuk menentukan ARN yang sepenuhnya memenuhi syarat untuk setiap file.rdb. Sebagai contoh, `arn:aws:s3:::`*myBucket*/*myFolder*/*myBackupFilename*`.rdb`. ARN harus menyelesaikan file snapshot yang Anda simpan di Amazon S3.

Selama proses pembuatan cluster Anda, data dalam snapshot Anda ditulis ke cluster. Anda dapat memantau kemajuan dengan melihat pesan acara MemoryDB. Untuk melakukan ini, lihat konsol MemoryDB dan pilih Acara. Anda juga dapat menggunakan antarmuka baris perintah AWS MemoryDB atau API MemoryDB untuk mendapatkan pesan acara.

Menandai snapshot

Anda dapat menetapkan metadata Anda sendiri untuk setiap snapshot dalam bentuk tag. Tag memungkinkan Anda untuk mengkategorikan snapshot Anda dengan cara yang berbeda, misalnya, berdasarkan tujuan, pemilik, atau lingkungan. Hal ini berguna ketika Anda memiliki banyak sumber daya dengan jenis yang sama—Anda dapat dengan cepat mengidentifikasi sumber daya tertentu berdasarkan tag yang telah Anda berikan. Untuk informasi selengkapnya, lihat Sumber daya yang dapat Anda tandai.

Tag alokasi biaya adalah sarana untuk melacak biaya Anda di beberapa layanan AWS dengan mengelompokkan pengeluaran Anda pada faktur berdasarkan nilai tag. Untuk mempelajari lebih lanjut tentang tag alokasi biaya, lihat Gunakan tag alokasi biaya.

Menggunakan konsol MemoryDB, APIAWS CLI, atau MemoryDB Anda dapat menambahkan, membuat daftar, memodifikasi, menghapus, atau menyalin tag alokasi biaya pada snapshot Anda. Untuk informasi selengkapnya, lihat Memantau biaya dengan tanda alokasi biaya.

Menghapus snapshot

Snapshot otomatis dihapus secara otomatis ketika batas retensi berakhir. Jika Anda menghapus cluster, semua snapshot otomatisnya juga dihapus.

MemoryDB menyediakan operasi penghapusan API yang memungkinkan Anda menghapus snapshot kapan saja, terlepas dari apakah snapshot dibuat secara otomatis atau manual. Karena snapshot manual tidak memiliki batas retensi, penghapusan manual adalah satu-satunya cara untuk menghapusnya.

Anda dapat menghapus snapshot menggunakan konsol MemoryDB, APIAWS CLI, atau MemoryDB.

Menghapus snapshot (Konsol)

Prosedur berikut menghapus snapshot menggunakan konsol MemoryDB.

Menghapus snapshot

1. Masuk ke AWS Management Console dan buka konsol MemoryDB for Redis di https://console.aws.amazon.com/memorydb/.
2. Di panel navigasi kiri, pilih Snapshots.

   Layar Snapshots muncul dengan daftar snapshot Anda.
3. Pilih tombol radio di sebelah kiri nama snapshot yang ingin Anda hapus.
4. Pilih Tindakan dan kemudian pilih Hapus.
5. Jika Anda ingin menghapus snapshot ini, masukkan `delete` di kotak teks lalu pilih Hapus. Untuk membatalkan penghapusan, pilih Batal. Status berubah menjadi menghapus.

Menghapus snapshot (CLIAWS)

Gunakan AWS CLI operasi hapus-snapshot dengan parameter berikut untuk menghapus snapshot.

- `--snapshot-name`— Nama snapshot yang akan dihapus.

Kode berikut menghapus snapshot`myBackup`.

```
aws memorydb delete-snapshot --snapshot-name myBackup
```

Untuk informasi selengkapnya, lihat hapus-snapshot di Referensi Perintah. AWS CLI

Menghapus snapshot (MemoryDB API)

Gunakan operasi `DeleteSnapshot` API dengan parameter berikut untuk menghapus snapshot.

- `SnapshotName`— Nama snapshot yang akan dihapus.

Kode berikut menghapus snapshot`myBackup`.

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=DeleteSnapshot
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &SnapshotName=myBackup
   &Timestamp=20210802T192317Z
   &Version=2021-01-01
   &X-Amz-Credential=<credential>
```

Untuk informasi lebih lanjut, lihat DeleteSnapshot.

Penskalaan

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 aplikasi Anda dikelola sendiri, Anda perlu menyediakan perangkat keras yang memadai untuk puncak permintaan, yang bisa jadi mahal. Dengan menggunakan MemoryDB untuk Redis Anda dapat menskalakan untuk memenuhi permintaan saat ini, hanya membayar untuk apa yang Anda gunakan.

Berikut ini akan membantu Anda menemukan topik yang benar untuk tindakan penskalaan yang ingin Anda lakukan.

Menskalakan MemoryDB

Action	MemoriDB	
Menskalakan keluar	Resharding dan penyeimbangan ulang serpihan secara online untuk MemoryDB	

Action	MemoriDB	
Mengubah tipe simpul	Penskalaan vertikal online dengan mengubah tipe simpul	
Mengubah jumlah serpihan	Menskalakan klaster MemoryDB	

Menskalakan klaster MemoryDB

Seiring perubahan permintaan pada klaster, Anda mungkin memutuskan untuk meningkatkan performa atau mengurangi biaya dengan mengubah jumlah serpihan dalam klaster MemoryDB 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 keluar agar memiliki lebih banyak sumber daya untuk menyimpan data yang lebih baik dan melayani permintaan.

  Anda dapat menentukan apakah node Anda berada di bawah tekanan memori dengan memantau metrik berikut: FreeableMemory, SwapUsage, dan BytesUsedForMemoryDB.
- Hambatan CPU atau jaringan:

  Jika masalah latensi/throughput mengganggu klaster, Anda mungkin perlu untuk menskalakan keluar untuk menyelesaikan masalah.

  Anda dapat memantau tingkat latensi dan throughput Anda 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 Anda untuk menentukan apakah Anda dapat dengan aman menskalakan dalam menggunakan metrik berikut: FreeableMemory, SwapUsage, BytesUsedForMemoryDB, 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 MemoryDB Anda; penskalaan horizontal dan vertikal.

- Penskalaan horizontal mengizinkan Anda mengubah jumlah serpihan dalam klaster dengan menambahkan atau menghapus serpihan. Proses resharding online memungkinkan penskalaan ke dalam/luar sementara klaster terus melayani permintaan masuk.
- Penskalaan Vertikal - Ubah tipe simpul untuk mengubah ukuran klaster. Penskalaan vertikal secara online memungkinkan penskalaan ke atas/bawah sementara klaster terus melayani permintaan masuk.

Jika Anda mengurangi ukuran dan kapasitas memori klaster, baik dengan menskalakan ke dalam atau menskalakan ke bawah, pastikan bahwa konfigurasi baru memiliki memori yang cukup untuk data Anda dan overhead Redis.

Resharding dan penyeimbangan ulang serpihan secara offline untuk MemoryDB

Keuntungan utama yang Anda dapatkan dari konfigurasi ulang serpihan secara offline adalah bahwa Anda dapat melakukan lebih dari sekadar menambahkan atau menghapus serpihan dari klaster Anda. Ketika Anda melakukan reshard offline, selain mengubah jumlah serpihan di klaster, Anda dapat melakukan hal berikut:

- Mengubah tipe simpul klaster Anda.
- Tingkatkan ke versi mesin yang lebih baru.

> **Note**
>
> Resharding offline tidak didukung pada cluster dengan tiering data diaktifkan. Untuk informasi selengkapnya, lihat Tingkatan data.

Kekurangan utama dari konfigurasi ulang serpihan secara offline adalah bahwa klaster Anda offline dimulai dengan bagian pemulihan dari proses dan terus sampai Anda memperbarui titik akhir dalam aplikasi Anda. Lama waktu klaster Anda offline bervariasi dengan jumlah data dalam klaster Anda.

Untuk mengonfigurasi ulang klaster MemoryDB serpihan Anda secara offline

1. Buat snapshot manual klaster MemoryDB Anda yang ada. Untuk informasi selengkapnya, lihat Membuat snapshot manual.
2. Buat klaster baru dengan memulihkan dari snapshot. Untuk informasi selengkapnya, lihat Memulihkan dari snapshot.
3. Perbarui titik akhir dalam aplikasi Anda untuk titik akhir klaster baru. Untuk informasi selengkapnya, lihat Menemukan titik akhir sambungan.

Resharding dan penyeimbangan ulang serpihan secara online untuk MemoryDB

Dengan menggunakan resharding dan penyeimbangan ulang serpihan secara online dengan MemoryDB, Anda dapat menskalakan memori Anda secara dinamis tanpa wkatu henti. Pendekatan ini berarti bahwa klaster Anda dapat terus melayani permintaan bahkan saat penskalaan atau penyeimbangan ulang sedang dalam proses.

Anda dapat melakukan hal berikut:

- Skala keluar - Tingkatkan kapasitas baca dan tulis dengan menambahkan pecahan ke cluster MemoryDB Anda.

  Jika Anda menambahkan satu atau lebih serpihan ke klaster, jumlah simpul di setiap serpihan baru adalah sama dengan jumlah simpul dalam serpihan terkecil yang ada.
- Menskalakan ke dalam - Mengurangi kapasitas baca dan tulis, dan dengan demikian biaya, dengan menghapus serpihan dari klaster MemoryDB Anda.

Saat ini, batasan berikut berlaku untuk reshding online MemoryDB:

- Ada pembatasan dengan slot atau keyspaces dan item besar:

  Jika salah satu kunci dalam serpihan berisi item besar, kunci yang tidak dimigrasikan ke serpihan baru ketika penskalaan keluar atau penyeimbangan ulang. Fungsi ini dapat mengakibatkan serpihan yang tidak seimbang.

  Jika salah satu kunci dalam serpihan berisi item besar (item lebih besar dari 256 MB setelah serialisasi), serpihan yang tidak dihapus ketika penskalaan 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 yang ada.

Untuk informasi selengkapnya, lihat Praktik terbaik: Mengubah ukuran klaster secara online.

Anda dapat secara horizontal skala atau menyeimbangkan cluster MemoryDB Anda menggunakanAWS Management Console, theAWS CLI, dan API MemoryDB.

Menambahkan serpihan dengan resharding online

Anda dapat menambahkan pecahan ke cluster MemoryDB Anda menggunakanAWS Management Console,AWS CLI, atau API MemoryDB.

Menambahkan serpihan (Konsol)

Anda dapat menggunakanAWS Management Console untuk menambahkan satu atau lebih serpihan ke klaster MemoryDB Anda. Prosedur berikut menjelaskan prosesnya.

1. Masuk keAWS Management Console dan buka konsol MemoryDB untuk Redis di https://console.aws.amazon.com/memorydb/.
2. Pada daftar klaster, pilih nama klaster yang ingin Anda tambahkan serpihan.
3. Di bawah tab Pecahan dan node, pilih Tambah/Hapus pecahan
4. Di Jumlah pecahan baru, masukkan jumlah pecahan yang Anda inginkan.
5. Pilih Konfirmasi untuk menyimpan perubahan atau Batal untuk dibuang.

Menambahkan serpihan (AWS CLI)

Proses berikut menjelaskan cara mengonfigurasi ulang serpihan di klaster MemoryDB Anda dengan menambahkan serpihan menggunakanAWS CLI.

Gunakan parameter berikut dengan `update-cluster`.

Parameter

- `--cluster-name` – Diperlukan. Menentukan klaster (klaster) mana tempat operasi konfigurasi ulang serpihan dilakukan.
- `--shard-configuration` – Diperlukan. Memungkinkan Anda untuk menentukan jumlah serpihan.

- `ShardCount`— Mengatur properti ini untuk menentukan jumlah shard yang Anda inginkan.

Example

Contoh berikut memodifikasi jumlah pecahan di cluster`my-cluster` menjadi 2.

Untuk Linux, macOS, atau Unix:

```
aws memorydb update-cluster \
    --cluster-name my-cluster \
    --shard-configuration \
        ShardCount=2
```

Untuk Windows:

```
aws memorydb update-cluster ^
    --cluster-name my-cluster ^
    --shard-configuration ^
        ShardCount=2
```

Ia mengembalikan respon JSON berikut:

```
{
    "Cluster": {
        "Name": "my-cluster",
        "Status": "updating",
        "NumberOfShards": 2,
        "AvailabilityMode": "MultiAZ",
        "ClusterEndpoint": {
            "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-east-1.amazonaws.com",
            "Port": 6379
        },
        "NodeType": "db.r6g.large",
        "EngineVersion": "6.2",
        "EnginePatchVersion": "6.2.6",
        "ParameterGroupName": "default.memorydb-redis6",
        "ParameterGroupStatus": "in-sync",
        "SubnetGroupName": "my-sg",
        "TLSEnabled": true,
        "ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",
        "SnapshotRetentionLimit": 0,
        "MaintenanceWindow": "wed:03:00-wed:04:00",
```

```
        "SnapshotWindow": "04:30-05:30",
        "DataTiering": "false",
        "AutoMinorVersionUpgrade": true
    }
}
```

Untuk melihat rincian klaster yang diperbarui setelah statusnya berubah dari memperbarui ke tersedia, gunakan perintah berikut:

Untuk Linux, macOS, atau Unix:

```
aws memorydb describe-clusters \
    --cluster-name my-cluster
    --show-shard-details
```

Untuk Windows:

```
aws memorydb describe-clusters ^
    --cluster-name my-cluster
    --show-shard-details
```

Ini akan mengembalikan respon JSON berikut:

```
{
    "Clusters": [
        {
            "Name": "my-cluster",
            "Status": "available",
            "NumberOfShards": 2,
            "Shards": [
                {
                    "Name": "0001",
                    "Status": "available",
                    "Slots": "0-8191",
                    "Nodes": [
                        {
                            "Name": "my-cluster-0001-001",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1a",
                            "CreateTime": "2021-08-21T20:22:12.405000-07:00",
                            "Endpoint": {
```

```
                            "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                            "Port": 6379
                        }
                    },
                    {
                        "Name": "my-cluster-0001-002",
                        "Status": "available",
                        "AvailabilityZone": "us-east-1b",
                        "CreateTime": "2021-08-21T20:22:12.405000-07:00",
                        "Endpoint": {
                            "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                            "Port": 6379
                        }
                    }
                ],
                "NumberOfNodes": 2
            },
            {
                "Name": "0002",
                "Status": "available",
                "Slots": "8192-16383",
                "Nodes": [
                    {
                        "Name": "my-cluster-0002-001",
                        "Status": "available",
                        "AvailabilityZone": "us-east-1b",
                        "CreateTime": "2021-08-22T14:26:18.693000-07:00",
                        "Endpoint": {
                            "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                            "Port": 6379
                        }
                    },
                    {
                        "Name": "my-cluster-0002-002",
                        "Status": "available",
                        "AvailabilityZone": "us-east-1a",
                        "CreateTime": "2021-08-22T14:26:18.765000-07:00",
                        "Endpoint": {
                            "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                            "Port": 6379
```

```
                            }
                        }
                    ],
                    "NumberOfNodes": 2
                }
            ],
            "ClusterEndpoint": {
                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                "Port": 6379
            },
            "NodeType": "db.r6g.large",
            "EngineVersion": "6.2",
            "EnginePatchVersion": "6.2.6",
            "ParameterGroupName": "default.memorydb-redis6",
            "ParameterGroupStatus": "in-sync",
            "SubnetGroupName": "my-sg",
            "TLSEnabled": true,
            "ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",
            "SnapshotRetentionLimit": 0,
            "MaintenanceWindow": "wed:03:00-wed:04:00",
            "SnapshotWindow": "04:30-05:30",
            "ACLName": "my-acl",
            "DataTiering": "false",
            "AutoMinorVersionUpgrade": true
        }
    ]
}
```

Untuk informasi selengkapnya, lihat update-cluster diAWS CLI Command Reference.

Menambahkan serpihan (API MemoryDB)

Anda dapat menggunakan API MemoryDB untuk mengonfigurasi ulang serpihan di klaster MemoryDB Anda secara online dengan menggunakan`UpdateCluster` operasi.

Gunakan parameter berikut dengan `UpdateCluster`.

Parameter

- `ClusterName` – Diperlukan. Menentukan klaster operasi konfigurasi ulang serpihan dilakukan.
- `ShardConfiguration` – Diperlukan. Memungkinkan Anda untuk menentukan jumlah serpihan.
  - `ShardCount`— Mengatur properti ini untuk menentukan jumlah shard yang Anda inginkan.

Untuk informasi lebih lanjut, lihat UpdateCluster.

Menghapus serpihan dengan resharding online

Anda dapat menghapus pecahan dari cluster MemoryDB Anda menggunakanAWS Management Console,AWS CLI, atau API MemoryDB.

Menghapus serpihan (Konsol)

Proses berikut menjelaskan cara mengonfigurasi ulang serpihan di klaster MemoryDB Anda dengan menghapus serpihan menggunakanAWS Management Console.

> **Important**
>
> Sebelum menghapus serpihan dari klaster, MemoryDB memastikan bahwa semua data Anda akan sesuai dengan serpihan yang tersisa. Jika data akan cocok, serpihan dihapus dari klaster seperti yang diminta. Jika data tidak akan cocok dalam serpihan yang tersisa, proses dihentikan dan klaster yang tersisa dengan konfigurasi serpihan yang sama seperti sebelum permintaan dibuat.

Anda dapat menggunakanAWS Management Console untuk menghapus satu atau lebih serpihan dari klaster MemoryDB Anda. Anda tidak dapat menghapus semua serpihan dalam klaster. Sebagai gantinya, Anda harus menghapus klaster. Untuk informasi selengkapnya, lihat Langkah 4: Menghapus cluster. Prosedur berikut menjelaskan proses untuk menghapus satu atau lebih serpihan.

1. Masuk keAWS Management Console dan buka konsol MemoryDB untuk Redis di https://console.aws.amazon.com/memorydb/.
2. Pada daftar klaster, pilih nama klaster yang ingin dihapus serpihan.
3. Di bawah tab Pecahan dan node, pilih Tambah/Hapus pecahan
4. Di Jumlah pecahan baru, masukkan jumlah pecahan yang Anda inginkan (dengan minimal 1).
5. Pilih Konfirmasi untuk menyimpan perubahan atau Batal untuk dibuang.

Menghapus serpihan (AWS CLI)

Proses berikut menjelaskan cara mengonfigurasi ulang serpihan di klaster MemoryDB Anda dengan menghapus serpihan menggunakanAWS CLI.

> **Important**
>
> Sebelum menghapus serpihan dari klaster, MemoryDB memastikan bahwa semua data Anda akan sesuai dengan serpihan yang tersisa. Jika data akan cocok, serpihan dihapus dari klaster seperti yang diminta dan keyspaces mereka dipetakan ke serpihan yang tersisa. Jika data tidak akan cocok dalam serpihan yang tersisa, proses dihentikan dan klaster yang tersisa dengan konfigurasi serpihan yang sama seperti sebelum permintaan dibuat.

Anda dapat menggunakanAWS CLI untuk menghapus satu atau lebih serpihan dari klaster MemoryDB Anda. Anda tidak dapat menghapus semua serpihan dalam klaster. Sebagai gantinya, Anda harus menghapus klaster. Untuk informasi selengkapnya, lihat Langkah 4: Menghapus cluster.

Gunakan parameter berikut dengan `update-cluster`.

Parameter

- `--cluster-name` – Diperlukan. Menentukan klaster (klaster) mana tempat operasi konfigurasi ulang serpihan dilakukan.
- `--shard-configuration` – Diperlukan. Memungkinkan Anda untuk mengatur jumlah pecahan menggunakan`ShardCount` properti:

  `ShardCount`— Mengatur properti ini untuk menentukan jumlah shard yang Anda inginkan.

Example

Contoh berikut memodifikasi jumlah pecahan di cluster`my-cluster` menjadi 2.

Untuk Linux, macOS, atau Unix:

```
aws memorydb update-cluster \
    --cluster-name my-cluster \
    --shard-configuration \
        ShardCount=2
```

Untuk Windows:

```
aws memorydb update-cluster ^
```

```
    --cluster-name my-cluster ^
    --shard-configuration ^
        ShardCount=2
```

Ia mengembalikan respon JSON berikut:

```
{
    "Cluster": {
        "Name": "my-cluster",
        "Status": "updating",
        "NumberOfShards": 2,
        "AvailabilityMode": "MultiAZ",
        "ClusterEndpoint": {
            "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-east-1.amazonaws.com",
            "Port": 6379
        },
        "NodeType": "db.r6g.large",
        "EngineVersion": "6.2",
        "EnginePatchVersion": "6.2.6",
        "ParameterGroupName": "default.memorydb-redis6",
        "ParameterGroupStatus": "in-sync",
        "SubnetGroupName": "my-sg",
        "TLSEnabled": true,
        "ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",
        "SnapshotRetentionLimit": 0,
        "MaintenanceWindow": "wed:03:00-wed:04:00",
        "SnapshotWindow": "04:30-05:30",
        "DataTiering": "false",
        "AutoMinorVersionUpgrade": true
    }
}
```

Untuk melihat rincian klaster yang diperbarui setelah statusnya berubah dari memperbarui ke tersedia, gunakan perintah berikut:

Untuk Linux, macOS, atau Unix:

```
aws memorydb describe-clusters \
    --cluster-name my-cluster
    --show-shard-details
```

Untuk Windows:

```
aws memorydb describe-clusters ^
    --cluster-name my-cluster
    --show-shard-details
```

Ini akan mengembalikan respon JSON berikut:

```
{
    "Clusters": [
        {
            "Name": "my-cluster",
            "Status": "available",
            "NumberOfShards": 2,
            "Shards": [
                {
                    "Name": "0001",
                    "Status": "available",
                    "Slots": "0-8191",
                    "Nodes": [
                        {
                            "Name": "my-cluster-0001-001",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1a",
                            "CreateTime": "2021-08-21T20:22:12.405000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        },
                        {
                            "Name": "my-cluster-0001-002",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1b",
                            "CreateTime": "2021-08-21T20:22:12.405000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        }
                    ],
                    "NumberOfNodes": 2
```

```
                },
                {
                    "Name": "0002",
                    "Status": "available",
                    "Slots": "8192-16383",
                    "Nodes": [
                        {
                            "Name": "my-cluster-0002-001",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1b",
                            "CreateTime": "2021-08-22T14:26:18.693000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        },
                        {
                            "Name": "my-cluster-0002-002",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1a",
                            "CreateTime": "2021-08-22T14:26:18.765000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        }
                    ],
                    "NumberOfNodes": 2
                }
            ],
            "ClusterEndpoint": {
                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                "Port": 6379
            },
            "NodeType": "db.r6g.large",
            "EngineVersion": "6.2",
            "EnginePatchVersion": "6.2.6",
            "ParameterGroupName": "default.memorydb-redis6",
            "ParameterGroupStatus": "in-sync",
            "SubnetGroupName": "my-sg",
            "TLSEnabled": true,
```

```
            "ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",
            "SnapshotRetentionLimit": 0,
            "MaintenanceWindow": "wed:03:00-wed:04:00",
            "SnapshotWindow": "04:30-05:30",
            "ACLName": "my-acl",
            "DataTiering": "false",
            "AutoMinorVersionUpgrade": true
        }
    ]
}
```

Untuk informasi selengkapnya, lihat update-cluster diAWS CLI Command Reference.

Menghapus serpihan (API MemoryDB)

Anda dapat menggunakan API MemoryDB untuk mengonfigurasi ulang serpihan di klaster MemoryDB Anda secara online dengan menggunakan`UpdateCluster` operasi.

Proses berikut menjelaskan cara mengonfigurasi ulang serpihan di klaster MemoryDB Anda dengan menghapus serpihan menggunakan API MemoryDB.

> **Important**
>
> Sebelum menghapus serpihan rom klaster, MemoryDB memastikan bahwa semua data Anda akan sesuai dengan serpihan yang tersisa. Jika data akan cocok, serpihan dihapus dari klaster seperti yang diminta dan keyspaces mereka dipetakan ke serpihan yang tersisa. Jika data tidak akan cocok dalam serpihan yang tersisa, proses dihentikan dan klaster yang tersisa dengan konfigurasi serpihan yang sama seperti sebelum permintaan dibuat.

Anda dapat menggunakan API MemoryDB untuk menghapus satu atau lebih serpihan dari klaster MemoryDB Anda. Anda tidak dapat menghapus semua serpihan dalam klaster. Sebagai gantinya, Anda harus menghapus klaster. Untuk informasi selengkapnya, lihat Langkah 4: Menghapus cluster.

Gunakan parameter berikut dengan `UpdateCluster`.

Parameter

- `ClusterName` – Diperlukan. Menentukan klaster (klaster) mana tempat operasi konfigurasi ulang serpihan dilakukan.

- `ShardConfiguration` – Diperlukan. Memungkinkan Anda untuk mengatur jumlah pecahan menggunakan`ShardCount` properti:

  `ShardCount`— Mengatur properti ini untuk menentukan jumlah shard yang Anda inginkan.

Penskalaan vertikal online dengan mengubah tipe simpul

Dengan menggunakan penskalaan vertikal online dengan MemoryDB, Anda dapat menskalakan klaster Anda secara dinamis dengan waktu henti minimal. Hal ini mengizinkan klaster Anda untuk melayani permintaan bahkan saat penskalaan.

> **Note**
>
> Scaling tidak didukung antara klaster data tiering (misalnya, cluster menggunakan tipe node r6gd) dan cluster yang tidak menggunakan data tiering (misalnya, cluster menggunakan tipe node r6g). Untuk informasi selengkapnya, lihat Tingkatan data.

Anda dapat melakukan hal berikut:

- Menskalakan ke atas - Meningkatkan kapasitas baca dan tulis dengan menyesuaikan tipe simpul dari klaster MemoryDB Anda untuk menggunakan tipe simpul yang lebih besar.

  MemoryDB secara dinamis mengubah ukuran klaster Anda sambil tetap online dan melayani permintaan.
- Menskalakan ke bawah - Mengurangi kapasitas baca dan tulis dengan menyesuaikan tipe simpul ke bawah untuk menggunakan simpul yang lebih kecil. Lagi, MemoryDB secara dinamis mengubah ukuran klaster Anda sambil tetap online dan melayani permintaan. Dalam hal ini, Anda mengurangi biaya dengan mengurangi ukuran simpul.

> **Note**
>
> Proses menskalakan ke atas dan menskalakan ke bawah bergantung pada pembuatan klaster dengan tipe simpul yang baru dipilih dan menyinkronkan simpul baru dengan yang sebelumnya. Untuk memastikan aliran penskalaan ke atas/bawah yang mulus, lakukan hal berikut:

- Sementara proses penskalaan vertikal dirancang untuk tetap sepenuhnya online, itu tidak bergantung pada sinkronisasi data antara simpul lama dan simpul baru. Kami merekomendasikan Anda untuk memulai penskalaan ke atas/bawah selama jam ketika Anda memperkirakan lalu lintas data menjadi minimum.
- Uji perilaku aplikasi Anda selama penskalaan ke dalam di lingkungan persiapan, jika memungkinkan.

Penskalaan ke atas secara online

Topik

- Menskalakan ke atas klaster MemoryDB (Konsol)
- Menskalakan cluster MemoryDB (AWSCLI)
- Meningkatkan klaster MemoryDB (MemoryDB API)

Menskalakan ke atas klaster MemoryDB (Konsol)

Prosedur berikut menjelaskan cara menskalakan ke atas klaster MemoryDB menggunakanAWS Management Console. Selama proses ini, klaster MemoryDB Anda akan terus melayani permintaan dengan waktu henti minimal.

Untuk menskalakan ke atas klaster (konsol)

1. Masuk keAWS Management Console dan buka konsol MemoryDB untuk Redis di https://console.aws.amazon.com/memorydb/.
2. Pada daftar klaster, pilih klaster tersebut.
3. Pilih Tindakan, dan kemudian pilih Ubah.
4. Dalam dialog Modify Cluster:
   - Pilih tipe simpul yang ingin Anda skalakan dari daftar Tipe simpul. Untuk menskalakan ke atas, pilih tipe simpul yang lebih besar dari simpul yang ada.
5. Pilih Save changes (Simpan perubahan).

   Status klaster berubah menjadi modifikasi. Ketika status berubah ke tersedia, pengubahan selesai dan Anda dapat mulai menggunakan klaster baru.

Menskalakan cluster MemoryDB (AWSCLI)

Prosedur berikut menjelaskan cara menskalakan ke atas klaster MemoryDB menggunakanAWS CLI. Selama proses ini, klaster MemoryDB Anda akan terus melayani permintaan dengan waktu henti minimal.

Untuk meningkatkan klaster MemoryDB (AWSCLI)

1. Tentukan tipe simpul yang dapat Anda skalakan ke atas dengan menjalankan perintah AWS CLI `list-allowed-node-type-updates` dengan parameter berikut.

   Untuk Linux, macOS, atau Unix:

   ```
   aws memorydb list-allowed-node-type-updates \
        --cluster-name my-cluster-name
   ```

   Untuk Windows:

   ```
   aws memorydb list-allowed-node-type-updates ^
        --cluster-name my-cluster-name
   ```

   Output dari operasi di atas terlihat seperti berikut ini (format JSON).

   ```
   {
        "ScaleUpNodeTypes": [
            "db.r6g.2xlarge",
            "db.r6g.large"
        ],
        "ScaleDownNodeTypes": [
            "db.r6g.large"
        ],
   }
   ```

   Untuk informasi selengkapnya, lihat list-allowed-node-type-updates di AWS CLIReferensi.

2. Ubah klaster Anda untuk menskalakan ke atas ke tipe simpul baru yang lebih besar menggunakanAWS CLI`update-cluster` perintah dan parameter berikut.

   - `--cluster-name`— Nama klaster yang Anda skalakan ke atas.

- `--node-type`— Tipe simpul baru yang Anda inginkan untuk menskalakan klaster. Nilai ini harus berupa salah satu dari tipe simpul yang dihasilkan oleh`list-allowed-node-type-updates` perintah di langkah 1.

Untuk Linux, macOS, atau Unix:

```
aws memorydb update-cluster  \
 --cluster-name my-cluster \
 --node-type db.r6g.2xlarge
```

Untuk Windows:

```
aws memorydb update-cluster ^
     --cluster-name my-cluster ^
     --node-type db.r6g.2xlarge ^
```

Untuk informasi selengkapnya, lihat klaster pembaruan.

Meningkatkan klaster MemoryDB (MemoryDB API)

Proses berikut menskalakan klaster simpul saat ini ke tipe simpul baru yang lebih besar menggunakan API MemoryDB. Selama proses ini, MemoryDB memperbarui entri DNS sehingga menunjuk ke simpul baru. Anda dapat menskalakan klaster dengan failover otomatis diaktifkan saat klaster terus tetap online dan melayani permintaan masuk.

Jumlah waktu yang dibutuhkan untuk menskalakan ke atas ke tipe simpul yang lebih besar bervariasi, bergantung pada tipe simpul dan jumlah data dalam klaster Anda saat ini.

Untuk meningkatkan skala klaster MemoryDB (MemoryDB API)

1. Tentukan tipe simpul mana yang dapat Anda tingkatkan skalanya menggunakan`ListAllowedNodeTypeUpdates` tindakan API MemoryDB dengan parameter berikut.

   - `ClusterName`— nama cluster. Gunakan parameter ini untuk mendeskripsikan klaster tertentu daripada semua klaster.

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=ListAllowedNodeTypeUpdates
    &ClusterName=MyCluster
    &Version=2021-01-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210802T192317Z
    &X-Amz-Credential=<credential>
```

Untuk informasi lebih lanjut, lihat ListAllowedNodeTypeUpdatesdi MemoryDB for Redis API Reference.

2. Skalakan klaster Anda saat ini ke atas ke tipe simpul baru menggunakan tindakan APIUpdateCluster MemoryDB dan parameter berikut.

   - `ClusterName`— nama cluster.
   - `NodeType`— tipe simpul baru yang lebih besar dari klaster ini. Nilai ini harus berupa salah satu dari tipe instans yang dihasilkan oleh tindakan `ListAllowedNodeTypeUpdates` di langkah 1.

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=UpdateCluster
    &NodeType=db.r6g.2xlarge
    &ClusterName=myCluster
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210801T220302Z
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Date=20210801T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

Untuk informasi lebih lanjut, lihat UpdateCluster.

Penskalaan ke bawah secara online

Topik

- Menskalakan ke bawah klaster MemoryDB (Konsol)
- Menskalakan cluster MemoryDB (AWSCLI)
- Menskalakan cluster MemoryDB (MemoryDB API)

Menskalakan ke bawah klaster MemoryDB (Konsol)

Prosedur berikut menjelaskan cara menskalakan ke bawah klaster MemoryDB menggunakanAWS Management Console. Selama proses ini, klaster MemoryDB Anda akan terus melayani permintaan dengan waktu henti minimal.

Untuk menurunkan skala klaster MemoryDB (konsol)

1. Masuk keAWS Management Console dan buka konsol MemoryDB untuk Redis di https://console.aws.amazon.com/memorydb/.
2. Dari daftar klaster, pilih klaster pilihan Anda.
3. Pilih Tindakan, dan kemudian pilih Ubah.
4. Dalam dialog Modify Cluster:
   - Pilih tipe simpul yang ingin Anda skalakan dari daftar Tipe simpul. Untuk menskalakan ke bawah, pilih tipe simpul yang lebih kecil dari simpul yang ada. Perhatikan bahwa tidak semua tipe simpul yang tersedia untuk diskalakan ke bawah.
5. Pilih Save changes (Simpan perubahan).

   Status klaster berubah menjadi modifikasi. Ketika status berubah ke tersedia, pengubahan selesai dan Anda dapat mulai menggunakan klaster baru.

Menskalakan cluster MemoryDB (AWSCLI)

Prosedur berikut menjelaskan cara menskalakan ke bawah klaster MemoryDB menggunakanAWS CLI. Selama proses ini, klaster MemoryDB Anda akan terus melayani permintaan dengan waktu henti minimal.

Untuk menurunkan skala cluster MemoryDB (AWSCLI)

1. Tentukan tipe simpul yang dapat Anda skalakan ke bawah dengan menjalankan perintah AWS CLI `list-allowed-node-type-updates` dengan parameter berikut.

   Untuk Linux, macOS, atau Unix:

   ```
   aws memorydb list-allowed-node-type-updates \
        --cluster-name my-cluster-name
   ```

   Untuk Windows:

   ```
   aws memorydb list-allowed-node-type-updates ^
        --cluster-name my-cluster-name
   ```

   Output dari operasi di atas terlihat seperti berikut ini (format JSON).

   ```
   {
        "ScaleUpNodeTypes": [
            "db.r6g.2xlarge",
            "db.r6g.large"
        ],
        "ScaleDownNodeTypes": [
            "db.r6g.large"
        ],
   }
   ```

   Untuk informasi selengkapnya, lihat list-allowed-node-type-updates.

2. Ubah klaster Anda untuk menskalakan ke bawah ke tipe simpul baru yang lebih kecil menggunakan`update-cluster` perintah dan parameter berikut.

   - `--cluster-name`— Nama klaster Anda yang diskalakan ke bawah.
   - `--node-type`— Tipe simpul baru yang Anda inginkan untuk menskalakan klaster. Nilai ini harus berupa salah satu dari tipe simpul yang dihasilkan oleh`list-allowed-node-type-updates` perintah di langkah 1.

   Untuk Linux, macOS, atau Unix:

   ```
   aws memorydb update-cluster  \
   ```

```
    --cluster-name my-cluster \
    --node-type db.r6g.large
```

Untuk Windows:

```
aws memorydb update-cluster ^
    --cluster-name my-cluster ^
    --node-type db.r6g.large
```

Untuk informasi selengkapnya, lihat klaster pembaruan.

Menskalakan cluster MemoryDB (MemoryDB API)

Proses berikut menskalakan klaster simpul saat ini ke tipe simpul baru yang lebih kecil menggunakan API MemoryDB. Selama proses ini, klaster MemoryDB Anda akan terus melayani permintaan dengan waktu henti minimal.

Jumlah waktu yang dibutuhkan untuk menskalakan ke bawah ke tipe simpul yang lebih kecil bervariasi, bergantung pada tipe simpul dan jumlah data dalam klaster Anda saat ini.

Penskalaan ke bawah (MemoryDB API)

1. Tentukan tipe simpul yang dapat Anda turunkan skalanya menggunakan ListAllowedNodeTypeUpdatesAPI dengan parameter berikut:

   - `ClusterName`— nama cluster. Gunakan parameter ini untuk mendeskripsikan klaster tertentu daripada semua klaster.

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=ListAllowedNodeTypeUpdates
    &ClusterName=MyCluster
    &Version=2021-01-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210802T192317Z
    &X-Amz-Credential=<credential>
```

2. Skalakan klaster Anda saat ini ke bawah ke tipe simpul baru menggunakan UpdateClusterAPI dengan parameter berikut.

- `ClusterName`— nama cluster.
- `NodeType`— tipe simpul baru yang lebih kecil dari klaster ini. Nilai ini harus berupa salah satu dari tipe instans yang dihasilkan oleh tindakan `ListAllowedNodeTypeUpdates` di langkah 1.

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=UpdateCluster
    &NodeType=db.r6g.2xlarge
    &ClusterName=myReplGroup
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210801T220302Z
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Date=20210801T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

Mengonfigurasi parameter mesin menggunakan grup parameter

MemoryDB untuk Redis menggunakan parameter untuk mengontrol properti runtime node dan cluster 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 spesifik tipe node MemoryDB.

Topik

- Mengubah grup parameter
- Menghapus grup parameter
- 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 Anda menjalankan Redis versi 6, Anda hanya dapat menggunakan grup parameter, default atau kustom, dari keluarga memorydb_redis6.
- Saat Anda mengubah parameter klaster, perubahan tersebut akan segera diterapkan pada klaster tersebut. Hal ini berlaku terlepas dari apakah Anda mengubah grup parameter klaster itu sendiri atau nilai parameter di dalam grup parameter klaster.

Tingkat grup parameter

MemoryDB untuk tingkatan grup parameter Redis

Default Global

Grup parameter root tingkat atas untuk semua MemoryDB untuk pelanggan Redis di wilayah tersebut.

Grup parameter default global:

- Dicadangkan untuk MemoryDB dan tidak tersedia untuk pelanggan.

Default Pelanggan

Salinan grup parameter Global Default yang dibuat untuk digunakan pelanggan.

Grup parameter Default Pelanggan:

- Dibuat dan dimiliki oleh MemoryDB.
- Tersedia untuk pelanggan untuk digunakan sebagai grup parameter untuk setiap cluster yang menjalankan versi mesin yang didukung oleh grup parameter ini.
- Tidak dapat diedit oleh pelanggan.

Milik Pelanggan

Salinan grup parameter Default Pelanggan. Grup parameter Customer Owned dibuat setiap kali pelanggan membuat grup parameter.

Kelompok parameter yang dimiliki pelanggan:

- Dibuat dan dimiliki oleh pelanggan.
- Dapat ditetapkan untuk semua klaster pelanggan yang kompatibel.
- Dapat dimodifikasi oleh pelanggan untuk membuat grup parameter kustom.

  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 MemoryDB, API AWS CLI, atau MemoryDB.

Membuat grup parameter (Konsol)

Prosedur berikut menunjukkan cara membuat grup parameter menggunakan konsol MemoryDB.

Untuk membuat grup parameter menggunakan konsol MemoryDB

1. Masuk ke AWS Management Console dan buka konsol MemoryDB for Redis di https://console.aws.amazon.com/memorydb/.
2. Untuk melihat daftar semua grup parameter yang tersedia, di panel navigasi sebelah kiri pilih Grup Parameter.
3. Untuk membuat grup parameter, pilih Buat grup parameter.

   Halaman grup Create parameter muncul.
4. 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.
5. Di kotak Deskripsi, masukkan deskripsi untuk grup parameter yang baru.
6. Di kotak kompatibilitas versi Redis, pilih versi mesin yang sesuai dengan grup parameter ini.
7. Di Tag, secara opsional tambahkan tag untuk mencari dan memfilter grup parameter Anda atau melacak AWS biaya Anda.
8. Untuk membuat grup parameter, pilih Buat.

Untuk mengakhiri proses tanpa membuat grup parameter, pilih Batalkan.

9. 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-parameter-group` dengan parameter ini.

- `--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.
- `--family` — Keluarga mesin dan versi untuk grup parameter.
- `--description` — Deskripsi yang diberikan pengguna untuk grup parameter.

Example

Contoh berikut membuat grup parameter bernama MyRedis6x menggunakan keluarga memorydb_redis6 sebagai template.

Untuk Linux, macOS, atau Unix:

```
aws memorydb create-parameter-group \
    --parameter-group-name myRedis6x  \
    --family memorydb_redis6 \
    --description "My first parameter group"
```

Untuk Windows:

```
aws memorydb create-parameter-group ^
```

```
    --parameter-group-name myRedis6x  ^
    --family memorydb_redis6 ^
    --description "My first parameter group"
```

Output dari perintah ini akan terlihat seperti ini.

```
{
    "ParameterGroup": {
        "Name": "myRedis6x",
        "Family": "memorydb_redis6",
        "Description": "My first parameter group",
        "ARN": "arn:aws:memorydb:us-east-1:012345678912:parametergroup/myredis6x"
    }
}
```

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-parameter-group`.

Membuat grup parameter (MemoryDB API)

Untuk membuat grup parameter menggunakan API MemoryDB, gunakan `CreateParameterGroup` tindakan 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.
- `Family` — Keluarga mesin dan versi untuk grup parameter. Misalnya, `memorydb_redis6`.
- `Description` — Deskripsi yang diberikan pengguna untuk grup parameter.

Example

Contoh berikut membuat grup parameter bernama MyRedis6x menggunakan keluarga memorydb_redis6 sebagai template.

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=CreateParameterGroup
   &Family=memorydb_redis6
   &ParameterGroupName=myRedis6x
   &Description=My%20first%20parameter%20group
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20210802T192317Z
   &Version=2021-01-01
   &X-Amz-Credential=<credential>
```

Respons dari tindakan ini akan terlihat seperti ini.

```
<CreateParameterGroupResponse xmlns="http://memory-db.us-east-1.amazonaws.com/
doc/2021-01-01/">
  <CreateParameterGroupResult>
    <ParameterGroup>
      <Name>myRedis6x</Name>
      <Family>memorydb_redis6</Family>
      <Description>My first parameter group</Description>
      <ARN>arn:aws:memorydb:us-east-1:012345678912:parametergroup/myredis6x</ARN>
    </ParameterGroup>
  </CreateParameterGroupResult>
  <ResponseMetadata>
    <RequestId>d8465952-af48-11e0-8d36-859edca6f4b8</RequestId>
  </ResponseMetadata>
</CreateParameterGroupResponse>
```

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 `CreateParameterGroup`.

Membuat daftar grup parameter berdasarkan nama

Anda dapat membuat daftar grup parameter menggunakan konsol MemoryDB, API AWS CLI, atau MemoryDB.

Membuat daftar grup parameter berdasarkan nama (Konsol)

Prosedur berikut menunjukkan cara melihat daftar grup parameter menggunakan konsol MemoryDB.

Untuk daftar grup parameter menggunakan konsol MemoryDB

1. Masuk ke AWS Management Console dan buka konsol MemoryDB for Redis di https://console.aws.amazon.com/memorydb/.
2. Untuk melihat daftar semua grup parameter yang tersedia, pilih Grup Parameter di panel navigasi sebelah kiri.

Daftar grup parameter berdasarkan nama (AWS CLI)

Untuk menghasilkan daftar kelompok parameter menggunakan AWS CLI, gunakan perintah`describe-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-results` yang akan dicantumkan. Dalam kedua kasus, nama, keluarga, dan deskripsi grup parameter akan dicantumkan.

Example

Kode contoh berikut mencantumkan kelompok parameter MyRedis6x.

Untuk Linux, macOS, atau Unix:

```
aws memorydb describe-parameter-groups \
    --parameter-group-name myRedis6x
```

Untuk Windows:

```
aws memorydb describe-parameter-groups ^
    --parameter-group-name myRedis6x
```

Output dari perintah ini akan terlihat seperti ini, mencantumkan nama, keluarga, dan deskripsi untuk grup parameter.

```
{
    "ParameterGroups": [
     {
         "Name": "myRedis6x",
         "Family": "memorydb_redis6",
         "Description": "My first parameter group",
                "ARN": "arn:aws:memorydb:us-east-1:012345678912:parametergroup/
myredis6x"
     }
    ]
}
```

Example

Kode contoh berikut mencantumkan grup parameter MyRedis6x untuk grup parameter yang berjalan pada mesin Redis versi 5.0.6 dan seterusnya.

Untuk Linux, macOS, atau Unix:

```
aws memorydb describe-parameter-groups \
    --parameter-group-name myRedis6x
```

Untuk Windows:

```
aws memorydb describe-parameter-groups ^
    --parameter-group-name myRedis6x
```

Output dari perintah ini akan terlihat seperti ini, mencantumkan nama, keluarga, dan deskripsi untuk grup parameter.

```
{
    "ParameterGroups": [
     {
         "Name": "myRedis6x",
         "Family": "memorydb_redis6",
         "Description": "My first parameter group",
                "ARN": "arn:aws:memorydb:us-east-1:012345678912:parametergroup/
myredis6x"
     }
    ]
}
```

Example

Kode contoh berikut mencantumkan hingga 20 kelompok parameter.

```
aws memorydb describe-parameter-groups --max-results 20
```

Output JSON dari perintah ini akan terlihat seperti ini, mencantumkan nama, keluarga, dan deskripsi untuk setiap grup parameter.

```
{
    "ParameterGroups": [
        {
            "ParameterGroupName": "default.memorydb-redis6",
            "Family": "memorydb_redis6",
            "Description": "Default parameter group for memorydb_redis6",
            "ARN": "arn:aws:memorydb:us-east-1:012345678912:parametergroup/
default.memorydb-redis6"
        },
        ...
    ]
}
```

Untuk informasi selengkapnya, lihat describe-parameter-groups.

Daftar grup parameter berdasarkan nama (MemoryDB API)

Untuk menghasilkan daftar grup parameter menggunakan API MemoryDB, gunakan tindakan. `DescribeParameterGroups` Jika Anda memberikan nama grup parameter, hanya grup parameter tersebut yang akan dicantumkan. Jika Anda memberikan nama grup parameter, hanya grup parameter hingga `MaxResults` yang akan dicantumkan. Dalam kedua kasus, nama, keluarga, dan deskripsi grup parameter akan dicantumkan.

Example

Kode contoh berikut mencantumkan hingga 20 kelompok parameter.

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=DescribeParameterGroups
   &MaxResults=20
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20210802T192317Z
```

```
   &Version=2021-01-01
   &X-Amz-Credential=<credential>
```

Respons dari tindakan ini akan terlihat seperti ini, mencantumkan nama, keluarga, dan deskripsi dalam kasus memorydb_redis6, untuk setiap grup parameter.

```
<DescribeParameterGroupsResponse xmlns="http://memory-db.us-east-1.amazonaws.com/
doc/2021-01-01/">
  <DescribeParameterGroupsResult>
    <ParameterGroups>
      <ParameterGroup>
        <Name>myRedis6x</Name>
        <Family>memorydb_redis6</Family>
        <Description>My custom Redis 6 parameter group</Description>
        <ARN>arn:aws:memorydb:us-east-1:012345678912:parametergroup/myredis6x</ARN>
      </ParameterGroup>
       <ParameterGroup>
        <Name>default.memorydb-redis6</Name>
        <Family>memorydb_redis6</Family>
        <Description>Default parameter group for memorydb_redis6</Description>
        <ARN>arn:aws:memorydb:us-east-1:012345678912:parametergroup/default.memorydb-
redis6</ARN>
      </ParameterGroup>
    </ParameterGroups>
  </DescribeParameterGroupsResult>
  <ResponseMetadata>
    <RequestId>3540cc3d-af48-11e0-97f9-279771c4477e</RequestId>
  </ResponseMetadata>
</DescribeParameterGroupsResponse>
```

Example

Kode contoh berikut mencantumkan kelompok parameter MyRedis6x.

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=DescribeParameterGroups
   &ParameterGroupName=myRedis6x
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20210802T192317Z
   &Version=2021-01-01
   &X-Amz-Credential=<credential>
```

Respons dari tindakan ini akan terlihat seperti ini, mencantumkan nama, keluarga, dan deskripsi.

```
<DescribeParameterGroupsResponse xmlns="http://memory-db.us-east-1.amazonaws.com/
doc/2021-01-01/">
  <DescribeParameterGroupsResult>
    <ParameterGroups>
      <ParameterGroup>
        <Name>myRedis6x</Name>
        <Family>memorydb_redis6</Family>
        <Description>My custom Redis 6 parameter group</Description>
        <ARN>arn:aws:memorydb:us-east-1:012345678912:parametergroup/myredis6x</ARN>
      </ParameterGroup>
    </ParameterGroups>
  </DescribeParameterGroupsResult>
  <ResponseMetadata>
    <RequestId>3540cc3d-af48-11e0-97f9-279771c4477e</RequestId>
  </ResponseMetadata>
</DescribeParameterGroupsResponse>
```

Untuk informasi selengkapnya, lihat `DescribeParameterGroups`.

Membuat daftar nilai grup parameter

Anda dapat mencantumkan parameter dan nilainya untuk grup parameter menggunakan konsol MemoryDB, API AWS CLI, atau MemoryDB.

Membuat daftar nilai grup parameter (Konsol)

Prosedur berikut menunjukkan cara membuat daftar parameter dan nilainya untuk grup parameter menggunakan konsol MemoryDB.

Untuk membuat daftar parameter grup parameter dan nilainya menggunakan konsol MemoryDB

1. Masuk ke AWS Management Console dan buka konsol MemoryDB for Redis di https://console.aws.amazon.com/memorydb/.
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 nama (bukan kotak di sebelahnya) dari 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.

Daftar nilai kelompok parameter (AWS CLI)

Untuk membuat daftar parameter grup parameter dan nilainya menggunakan AWS CLI, gunakan perintah`describe-parameters`.

Example

Kode contoh berikut mencantumkan semua parameter dan nilainya untuk kelompok parameter MyRedis6x.

Untuk Linux, macOS, atau Unix:

```
aws memorydb describe-parameters \
    --parameter-group-name myRedis6x
```

Untuk Windows:

```
aws memorydb describe-parameters ^
    --parameter-group-name myRedis6x
```

Untuk informasi selengkapnya, lihat `describe-parameters`.

Daftar nilai grup parameter (MemoryDB API)

Untuk membuat daftar parameter grup parameter dan nilainya menggunakan API MemoryDB, gunakan tindakan. `DescribeParameters`

Untuk informasi selengkapnya, lihat `DescribeParameters`.

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 parameter menggunakan konsol MemoryDB. Anda akan menggunakan prosedur yang sama untuk mengubah nilai parameter apa pun.

Untuk mengubah nilai parameter menggunakan konsol MemoryDB

1. Masuk ke AWS Management Console dan buka konsol MemoryDB for Redis di https://console.aws.amazon.com/memorydb/.
2. Untuk melihat daftar semua grup parameter yang tersedia, pilih Grup Parameter di panel navigasi sebelah kiri.
3. Pilih grup parameter yang ingin Anda modifikasi dengan memilih tombol radio di sebelah kiri nama grup parameter.

   Pilih Tindakan dan kemudian Lihat detail. Atau, Anda juga dapat memilih nama grup parameter untuk pergi ke halaman detail.

4. Untuk memodifikasi parameter, pilih Edit. Semua parameter yang dapat diedit akan diaktifkan untuk diedit. Anda mungkin harus bergerak melintasi halaman untuk menemukan parameter yang ingin Anda ubah. Atau, Anda dapat mencari parameter berdasarkan nama, nilai, atau jenis di kotak pencarian.
5. Buat modifikasi parameter yang diperlukan.
6. Untuk menyimpan perubahan Anda, memilih Simpan perubahan.
7. Jika Anda mengubah nilai parameter di seluruh jumlah halaman, Anda dapat meninjau semua perubahan dengan memilih Pratinjau perubahan. Untuk mengonfirmasi perubahan, pilih Simpan perubahan. Untuk membuat lebih banyak modifikasi, pilih kembali.
8. Halaman detail Parameter juga memberi Anda opsi untuk mengatur ulang ke nilai default. Untuk mengatur ulang ke nilai default, pilih Reset ke default. Kotak centang akan muncul di sisi kiri semua parameter. Anda dapat memilih yang ingin Anda atur ulang dan memilih Lanjutkan untuk mengatur ulang untuk mengonfirmasi.

   Pilih konfirmasi untuk mengonfirmasi tindakan reset pada kotak dialog.
9. Halaman detail parameter memungkinkan Anda mengatur jumlah parameter yang ingin Anda lihat di setiap halaman. Gunakan roda gigi di sisi kanan untuk membuat perubahan itu. Anda juga dapat mengaktifkan/menonaktifkan kolom yang Anda inginkan di halaman detail. Perubahan ini berlangsung selama sesi konsol.

   Untuk menemukan nama parameter yang Anda ubah, lihat Parameter spesifik Redis.

Memodifikasi grup parameter (AWS CLI)

Untuk mengubah nilai parameter menggunakan AWS CLI, gunakan perintah`update-parameter-group`.

Untuk menemukan nama dan nilai yang diizinkan dari parameter yang ingin Anda ubah, lihat Parameter spesifik Redis

Untuk informasi lebih lanjut, lihat update-parameter-group.

Memodifikasi grup parameter (MemoryDB API)

Untuk mengubah nilai parameter grup parameter menggunakan API MemoryDB, gunakan tindakan. `UpdateParameterGroup`

Untuk menemukan nama dan nilai yang diizinkan dari parameter yang ingin Anda ubah, lihat Parameter spesifik Redis

Untuk informasi selengkapnya, lihat `UpdateParameterGroup`.

Menghapus grup parameter

Anda dapat menghapus grup parameter kustom menggunakan konsol MemoryDB, API AWS CLI, atau MemoryDB.

Anda tidak dapat menghapus grup parameter jika grup parameter ini dikaitkan dengan klaster. Anda juga tidak dapat menghapus grup parameter default.

Menghapus grup parameter (Konsol)

Prosedur berikut menunjukkan cara menghapus grup parameter menggunakan konsol MemoryDB.

Untuk menghapus grup parameter menggunakan konsol MemoryDB

1. Masuk ke AWS Management Console dan buka konsol MemoryDB for Redis di https://console.aws.amazon.com/memorydb/.
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 tombol radio di sebelah kiri nama grup parameter.

   Pilih Tindakan dan kemudian pilih Hapus.
4. Layar konfirmasi Hapus Grup Parameter akan muncul.
5. Untuk menghapus grup parameter, masukkan Hapus di kotak teks konfirmasi.

   Untuk mempertahankan grup parameter, pilih Batalkan.

Menghapus grup parameter (AWS CLI)

Untuk menghapus grup parameter menggunakan AWS CLI, gunakan perintah`delete-parameter-group`. Untuk grup parameter yang akan dihapus, grup parameter yang ditentukan berdasarkan `--parameter-group-name` tidak boleh memiliki klaster yang terkait dengannya, dan juga tidak dapat berupa grup parameter default.

Kode contoh berikut menghapus grup parameter MyRedis6x.

Example

Untuk Linux, macOS, atau Unix:

```
aws memorydb delete-parameter-group \
    --parameter-group-name myRedis6x
```

Untuk Windows:

```
aws memorydb delete-parameter-group ^
    --parameter-group-name myRedis6x
```

Untuk informasi lebih lanjut, lihat delete-parameter-group.

Menghapus grup parameter (MemoryDB API)

Untuk menghapus grup parameter menggunakan API MemoryDB, gunakan tindakan. `DeleteParameterGroup` Untuk grup parameter yang akan dihapus, grup parameter yang ditentukan berdasarkan `ParameterGroupName` tidak boleh memiliki klaster yang terkait dengannya, dan juga tidak dapat berupa grup parameter default.

Example

Kode contoh berikut menghapus grup parameter MyRedis6x.

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=DeleteParameterGroup
   &ParameterGroupName=myRedis6x
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20210802T192317Z
   &Version=2021-01-01
   &X-Amz-Credential=<credential>
```

Untuk informasi selengkapnya, lihat `DeleteParameterGroup`.

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

> **Note**
>
> MemoryDB telah memperkenalkan rilis pratinjau pencarian Vektor yang mencakup grup parameter baru yang tidak dapat diubah. `default.memorydb-redis7.search.preview` Grup parameter ini tersedia di konsol MemoryDB dan saat membuat vector-search-enabled cluster baru menggunakan perintah CLI create-cluster. Rilis pratinjau tersedia di AWS Wilayah berikut: US East (Virginia N.), US East (Ohio), US West (Oregon), Asia Pasifik (Tokyo), dan Eropa (Irlandia).

Keluarga kelompok parameter: memorydb_redis7

Parameter yang ditambahkan dalam Redis 7 adalah sebagai berikut.

Nama	Detail	Deskripsi
`latency-tracking`	Nilai yang diizinkan: `yes`, `no` Default: `no` Tipe: string	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
	Dapat diubah: Ya Perubahan berlaku: Segera di semua simpul dalam klaster.	
`hash-max-listpack-entries`	Nilai yang diizinkan: `0+` Default: `512` Tipe: integer Dapat diubah: Ya Perubahan berlaku: Segera di semua simpul dalam klaster.	Jumlah maksimum entri hash agar set data dikompresi.
`hash-max-listpack-value`	Nilai yang diizinkan: `0+` Default: `64` Tipe: integer Dapat diubah: Ya 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+` Default: `128` Tipe: integer Dapat diubah: Ya 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+` Default: `64` Tipe: integer Dapat diubah: Ya 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+` Default: `512` Tipe: integer 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+` Default: `64` Tipe: integer Dapat diubah: Ya 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+` Default: `128` Tipe: integer Dapat diubah: Ya 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+` Default: `64` Tipe: integer Dapat diubah: Ya Perubahan berlaku: Segera di semua simpul dalam klaster.	Gunakan `listpack` bukan `ziplist` untuk merepresentasikan pengenkodean hash kecil.

Redis 6 parameter

> **Note**
>
> Di mesin Redis versi 6.2, ketika keluarga node r6gd diperkenalkan untuk digunakan dengan Tingkatan data, only `noeviction`, `volatile-lru` dan kebijakan `allkeys-lru` max-memory didukung dengan tipe node r6gd.

Keluarga kelompok parameter: memorydb_redis6

Parameter yang ditambahkan dalam Redis 6 adalah sebagai berikut.

Nama	Detail	Deskripsi
`maxmemory-policy`	Jenis: STRING Nilai yang diizinkan: volatile-lru, allkeys-lru, volatile-lfu, allkeys-lfu, volatile-random, allkeys-random, volatile-ttl, noeviction Default: noeviction	Kebijakan pengosongan untuk kunci saat penggunaan memori maksimum tercapai. Untuk informasi selengkapnya, lihat Menggunakan Redis sebagai cache LRU Menggunakan Redis sebagai cache LRU.
`list-compress-depth`	Jenis: INTEGER Nilai yang diizinkan: 0- Default: 0	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: • 0: Menonaktifkan semua kompresi. • 1: Mulai mengompresi dengan simpul pertama masuk dari kepala dan ekor. [head]->simpul->simpul->...->simpul->[tail] Semua simpul kecuali jika [head] dan [tail] dikompresi.

Nama	Detail	Deskripsi
		• 2: Mulai mengompresi dengan simpul kedua masuk dari kepala dan ekor. [head]->[next]->simpul->simpul->...->simpul->[prev]->[tail] [head], [next], [prev], [tail] tidak dikompresi. Semua simpul lainnya dikompresi. • Dll.
`hll-sparse-max-bytes`	Jenis: INTEGER Nilai yang diizinkan: 1-16000 Default: 3000	HyperLogLog batas byte representasi jarang. Batas termasuk header 16 byte. Ketika HyperLogLog menggunakan representasi jarang melintasi batas ini, itu diubah menjadi representasi padat. Nilai yang lebih besar dari 16000 tidak disarankan karena pada titik tersebut dense representation lebih hemat memori. Kami merekomendasikan nilai sekitar 3000 untuk mendapatkan manfaat dari pengkodean hemat ruang tanpa memperlambat `PFADD` terlalu banyak, yaitu O (N) dengan pengkodean yang jarang. Nilai dapat dinaikkan menjadi ~ 10000 ketika CPU tidak menjadi perhatian, tetapi ruang adalah, dan kumpulan data terdiri dari banyak HyperLogLogs dengan kardinalitas dalam kisaran 0 - 15000.
`lfu-log-factor`	Jenis: INTEGER Nilai yang diizinkan: 1- Default: 10	Faktor log untuk menambah penghitung kunci untuk kebijakan penggusuran LFU.

Nama	Detail	Deskripsi
`lfu-decay-time`	Jenis: INTEGER Nilai yang diizinkan: 0- Default: 1	Jumlah waktu dalam hitungan menit untuk mengurangi penghitung kunci untuk kebijakan penggusuran LFU.
`active-defrag-max-scan-fields`	Jenis: INTEGER Nilai yang diizinkan: 1-1000000 Default: 1000	Jumlah maksimum bidang set/hash/zset/list yang akan diproses dari pemindaian kamus utama selama defragmentasi aktif.
`active-defrag-threshold-upper`	Jenis: INTEGER Nilai yang diizinkan: 1-100 Default: 100	Persentase maksimum fragmentasi yang mana kita menggunakan upaya maksimal.
`client-output-buffer-limit-pubsub-hard-limit`	Jenis: INTEGER Nilai yang diizinkan: 0- Default: 33554432	Untuk klien publish/subscribe Redis: Jika buffer output klien mencapai jumlah byte tertentu, klien akan terputus.
`client-output-buffer-limit-pubsub-soft-limit`	Jenis: INTEGER Nilai yang diizinkan: 0- Default: 8388608	Untuk klien terbitkan/berlangganan Redis: Jika buffer keluaran klien mencapai jumlah byte yang ditentukan, klien akan terputus, tetapi hanya jika kondisi ini berlanjut untuk `client-output-buffer-limit-pubsub-soft-seconds.`

Nama	Detail	Deskripsi
`client-output-buffer-limit-pubsub-soft-seconds`	Jenis: INTEGER Nilai yang diizinkan: 0- Default: 60	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.
`timeout`	Jenis: INTEGER Nilai yang diizinkan: 0,20- Default: 0	Jumlah detik waktu tunggu simpul sebelum batas waktunya habis Nilainya adalah: • 0 - jangan pernah memutuskan klien yang menganggur. • 1-19 - nilai tidak valid. • >=20 — jumlah detik node menunggu sebelum memutuskan sambungan klien idle.
`notify-keyspace-events`	Jenis: STRING Nilai yang diizinkan: NULL Default: NULL	Acara keyspace untuk Redis untuk memberi tahu klien Pub/Sub tentang. Secara default semua notifikasi dinonaktifkan.
`maxmemory-samples`	Jenis: INTEGER Nilai yang diizinkan: 1- Default: 3	Untuk least-recently-used `(LRU)` dan `time-to-live (TTL)` perhitungan, parameter ini mewakili ukuran sampel kunci untuk diperiksa . Secara default, Redis memilih 3 kunci dan menggunakan kunci yang paling lama tidak digunakan.

Nama	Detail	Deskripsi
`slowlog-max-len`	Jenis: INTEGER Nilai yang diizinkan: 0- Default: 128	Panjang maksimum Redis Slow Log. Tidak ada batasan untuk panjang ini. Ketahuilah bahwa itu akan menghabiskan memori. Anda dapat merebut kembali memori yang digunakan oleh log lambat dengan `SLOWLOG RESET`.
`activerehashing`	Jenis: STRING Nilai yang diizinkan: ya, tidak Default: yes	Tabel hash utama di-rehash sepuluh kali per detik; setiap operasi rehash mengonsumsi 1 milidetik waktu CPU. 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.
`client-output-buffer-limit-normal-hard-limit`	Jenis: INTEGER Nilai yang diizinkan: 0- Default: 0	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`	Jenis: INTEGER Nilai yang diizinkan: 0- Default: 0	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).
`client-output-buffer-limit-normal-soft-seconds`	Jenis: INTEGER Nilai yang diizinkan: 0- Default: 0	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).

Nama	Detail	Deskripsi
`tcp-keepalive`	Jenis: INTEGER Nilai yang diizinkan: 0- Default: 300	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.
`active-defrag-cycle-min`	Jenis: INTEGER Nilai yang diizinkan: 1-75 Default: 5	Upaya minimal untuk defrag dalam persentase CPU.
`stream-node-max-bytes`	Jenis: INTEGER Nilai yang diizinkan: 0- Default: 4096	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`	Jenis: INTEGER Nilai yang diizinkan: 0- 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 simpul tunggal sebelum beralih ke simpul baru saat menambahkan entri stream baru. Jika disetel ke 0, jumlah item di simpul pohon tidak terbatas.
`lazyfree-lazy-eviction`	Jenis: STRING Nilai yang diizinkan: ya, tidak Default: no	Lakukan penghapusan asinkron pada penggusuran.

Nama	Detail	Deskripsi
`active-defrag-ignore-bytes`	Jenis: INTEGER Nilai yang diizinkan: 1048576- Default: 104857600	Jumlah minimum sisa fragmentasi untuk memulai defrag aktif.
`lazyfree-lazy-expire`	Jenis: STRING Nilai yang diizinkan: ya, tidak Default: no	Lakukan penghapusan asinkron pada kunci kedaluwarsa.
`active-defrag-threshold-lower`	Jenis: INTEGER Nilai yang diizinkan: 1-100 Default: 10	Persentase minimum fragmentasi untuk memulai defrag aktif.
`active-defrag-cycle-max`	Jenis: INTEGER Nilai yang diizinkan: 1-75 Default: 75	Upaya maksimal untuk defrag dalam persentase CPU.
`lazyfree-lazy-server-del`	Jenis: STRING Nilai yang diizinkan: ya, tidak Default: no	Melakukan penghapusan asinkron untuk perintah yang memperbarui nilai.
`slowlog-log-slower-than`	Jenis: INTEGER Nilai yang diizinkan: 0- Default: 10000	Waktu eksekusi maksimum, dalam mikrodetik, melebihi agar perintah dapat dicatat oleh fitur Redis`Slow Log`. Perhatikan bahwa angka negatif menonaktifkan log lambat, sementara nilai nol memaksa pencatatan setiap perintah.

Nama	Detail	Deskripsi
`hash-max-ziplist-entries`	Jenis: INTEGER Nilai yang diizinkan: 0- Default: 512	Menentukan jumlah memori yang digunakan untuk hash. Hash dengan jumlah entri kurang dari yang ditentukan akan disimpan menggunakan pengenkodean khusus yang menghemat ruang.
`hash-max-ziplist-value`	Jenis: INTEGER Nilai yang diizinkan: 0- Default: 64	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.
`set-max-intset-entries`	Jenis: INTEGER Nilai yang diizinkan: 0- Default: 512	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.
`zset-max-ziplist-entries`	Jenis: INTEGER Nilai yang diizinkan: 0- Default: 128	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`	Jenis: INTEGER Nilai yang diizinkan: 0- Default: 64	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.

Nama	Detail	Deskripsi
`tracking-table-max-keys`	Jenis: INTEGER Nilai yang diizinkan: 1-100000000 Default: 1000000	Untuk membantu caching sisi klien, Redis mendukung pelacakan kunci mana yang diakses klien tertentu. 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.
`acllog-max-len`	Jenis: INTEGER Nilai yang diizinkan: 1-10000 Default: 128	Jumlah maksimum entri dalam Log ACL.

Nama	Detail	Deskripsi
`active-expire-effort`	Jenis: INTEGER Nilai yang diizinkan: 1-10 Default: 1	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. 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`	Jenis: STRING Nilai yang diizinkan: ya, tidak Default: no	Menentukan apakah perilaku default `DEL` perintah bertindak sama seperti`UNLINK`.

Nama	Detail	Deskripsi
activedefrag	Jenis: STRING Nilai yang diizinkan: ya, tidak Default: no	Diaktifkan defragmentasi memori aktif.
maxclients	Jenis: INTEGER Nilai yang diizinkan: 65000 Default: 65000	Jumlah maksimum klien yang dapat dihubungkan pada satu waktu. Tidak dapat dimodifikasi.
client-query-buffer-limit	Jenis: INTEGER Nilai yang diizinkan: 1048576-1073741824 Default: 1073741824	Ukuran maks buffer kueri klien tunggal. Perubahan terjadi segera.
proto-max-bulk-len	Jenis: INTEGER Nilai yang diizinkan: 1048576-536870912 Default: 536870912	Ukuran maks dari permintaan elemen tunggal. Perubahan terjadi segera.

Parameter spesifik tipe node MemoryDB

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 `maxmemory` untuk setiap jenis node. Nilai `maxmemory` adalah jumlah maksimum byte yang tersedia untuk Anda gunakan, untuk data, dan untuk penggunaan lainnya, pada simpul.

Jenis simpul	Maxmemory
db.r7g.large	14037181030
db.r7g.xlarge	28261849702
db.r7g.2xlarge	56711183565
db.r7g.4xlarge	113609865216
db.r7g.8xlarge	225000375228
db.r7g.12xlarge	341206346547
db.r7g.16xlarge	450000750456
db.r6gd.xlarge	28261849702
db.r6gd.2xlarge	56711183565
db.r6gd.4xlarge	113609865216
db.r6gd.8xlarge	225000375228
db.r6g.large	14037181030
db.r6g.xlarge	28261849702
db.r6g.2xlarge	56711183565
db.r6g.4xlarge	113609865216
db.r6g.8xlarge	225000375228
db.r6g.12xlarge	341206346547
db.r6g.16xlarge	450000750456
db.t4g.small	1471026299
db.t4g.medium	3317862236

> **Note**
>
> Semua jenis instans MemoryDB harus dibuat dalam VPC Amazon Virtual Private Cloud.

Tutorial: Mengonfigurasi fungsi Lambda untuk mengakses MemoryDB di VPC Amazon

Dalam tutorial ini Anda dapat mempelajari cara:

- Buat klaster MemoryDB di Amazon Virtual Private Cloud (Amazon VPC) default Anda di wilayah us-east-1.
- Buat fungsi Lambda untuk mengakses klaster . Saat membuat fungsi Lambda, Anda perlu menyediakan ID subnet di Amazon VPC dan grup keamanan VPC agar fungsi Lambda dapat mengakses sumber daya di VPC Anda. Sebagai ilustrasi dalam tutorial ini, fungsi Lambda menghasilkan UUID, menulisnya ke cluster, dan mengambilnya dari cluster..
- Panggil fungsi Lambda secara manual dan verifikasi bahwa ia mengakses cluster di VPC Anda.
- Bersihkan fungsi Lambda, klaster, dan peran IAM yang disiapkan untuk tutorial ini.

Topik

Langkah 1: Buat cluster

Untuk membuat cluster, ikuti langkah-langkah ini.

Topik

- Langkah 1.4: Buat Daftar Kontrol Akses (ACL)

Langkah 1.1: Buat cluster

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

```
aws memorydb create-cluster --cluster-name cluster-01 --engine-version 7.0 --acl-name
 open-access \
--description "MemoryDB IAM auth application" \
--node-type db.r6g.large
```

Perhatikan bahwa nilai bidang Status diatur ke `CREATING`. Diperlukan beberapa menit bagi MemoryDB untuk menyelesaikan pembuatan cluster Anda.

Langkah 1.2: Salin titik akhir cluster

Verifikasi bahwa MemoryDB telah selesai membuat cluster dengan perintah. `describe-clusters`

```
aws memorydb describe-clusters \
--cluster-name cluster-01
```

Salin Alamat Titik Akhir Cluster 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. Pastikan untuk mengganti account_id 123456789012 dalam kebijakan ini dengan account_id Anda.

   ```
   {
   "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. Pastikan untuk mengganti account_id 123456789012 dalam kebijakan ini dengan account_id Anda.

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

3. Buat peran IAM.

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

4. Buat kebijakan IAM.

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

5. Lampirkan kebijakan IAM di peran tersebut. Pastikan untuk mengganti account_id 123456789012 dalam kebijakan ini dengan account_id Anda.

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

Langkah 1.4: Buat Daftar Kontrol Akses (ACL)

1. Buat pengguna baru yang mendukung IAM.

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

2. Buat pengguna baru yang mendukung IAM.

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

3. Buat ACL dan lampirkan ke cluster.

```
aws memorydb create-acl \
  --acl-name iam-acl-01 \
  --user-names iam-user-01

aws memorydb update-cluster \
  --cluster-name cluster-01 \
  --acl-name iam-acl-01
```

Langkah 2: Buat fungsi Lambda

Untuk membuat fungsi Lambda, lakukan langkah-langkah ini.

Topik

Langkah 2.1: Buat paket deployment

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

Python

Contoh kode Python berikut membaca dan menulis item ke cluster MemoryDB Anda. Salin kode tersebut dan simpan ke dalam file bernama `app.py`. Pastikan untuk mengganti `cluster_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 MemoryDBIAMProvider(redis.CredentialProvider):
    def __init__(self, user, cluster_name, region="us-east-1"):
        self.user = user
        self.cluster_name = cluster_name
        self.region = region

        session = botocore.session.get_session()
        self.request_signer = RequestSigner(
            ServiceId("memorydb"),
            self.region,
            "memorydb",
            "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}

        url = urlunparse(
            ParseResult(
```

```
                scheme="https",
                netloc=self.cluster_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
        # MemoryDB 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
    cluster_name = "cluster-01" # replace with your cache name
    cluster_endpoint = "clustercfg.cluster-01.xxxxxx.memorydb.us-east-1.amazonaws.com"
 # replace with your cluster endpoint
    creds_provider = MemoryDBIAMProvider(user=username, cluster_name=cluster_name)
    redis_client = redis.Redis(host=cluster_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 cluster
    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 cluster and print
    # the results
    if decoded_result == uuid_in:
        print(f"Success: Inserted {uuid_in}. Fetched {decoded_result} from MemoryDB.")
    else:
        raise Exception(f"Bad value retrieved. Expected {uuid_in}, got
 {decoded_result}")

    return "Fetched value from MemoryDB"
```

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

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

```
mkdir package
```

Instal `redis-py` dan `cachetools` gunakan pip.

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

Buat file.zip yang berisi `redis-py` dan `cachetools` pustaka. Di Linux dan macOS, jalankan perintah berikut. Di Windows, gunakan utilitas zip pilihan Anda untuk membuat file.zip dengan `redis-py` dan `cachetools` perpustakaan 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 package
zip -r ../my_deployment_package.zip
```

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 "memorydb-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 (AccessMemoryDB) menggunakan perintah AWS CLI create-function.

Dari direktori proyek yang berisi berkas .zip paket penyebaran Anda, 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 AccessMemoryDB  \
--region us-east-1 \
--zip-file fileb://my_deployment_package.zip \
--role arn:aws:iam::123456789012:role/memorydb-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 AccessMemory (DB) menggunakan perintah AWS Lambda pemanggilan.

   ```
   aws lambda invoke \
   --function-name AccessMemoryDB  \
   --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 MemoryDB.
     ```

   - Tinjau hasilnya di AWS Lambda konsol.

Langkah 4: Bersihkan (Opsional)

Untuk membersihkannya, ambil langkah-langkah ini.

Topik

Langkah 4.1: Hapus fungsi Lambda

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

Langkah 4.2: Hapus cluster MemoryDB

Hapus klaster .

```
aws memorydb delete-cluster \
 --cluster-name cluster-01
```

Hapus pengguna dan ACL.

```
aws memorydb delete-user \
 --user-id iam-user-01
```

```
aws memorydb delete-acl \
 --acl-name iam-acl-01
```

Langkah 4.3: Hapus Peran dan kebijakan IAM

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

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

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

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

Pencarian vektor

Fitur ini dalam rilis pratinjau untuk MemoryDB untuk Redis dan dapat berubah.

Pencarian vektor untuk MemoryDB memperluas fungsionalitas MemoryDB. Pencarian vektor dapat digunakan bersama dengan fungsionalitas MemoryDB yang ada. Aplikasi yang tidak menggunakan pencarian vektor tidak terpengaruh oleh keberadaannya. Pratinjau pencarian vektor tersedia dalam versi MemoryDB 7.1 dan seterusnya di wilayah berikut: AS Timur (Virginia N. dan Ohio), AS Barat (Oregon), UE (Irlandia), dan Asia Pasifik (Tokyo).

Pencarian vektor untuk Amazon MemoryDB untuk Redis menyederhanakan arsitektur aplikasi Anda sambil memberikan pencarian vektor berkecepatan tinggi. Pencarian vektor untuk MemoryDB sangat ideal untuk kasus penggunaan di mana kinerja puncak dan skala adalah kriteria pemilihan yang paling penting. Anda dapat menggunakan data MemoryDB yang ada, atau Redis API, untuk membangun pembelajaran mesin dan kasus penggunaan AI generatif seperti generasi tambahan pengambilan, deteksi anomali, pengambilan dokumen, dan rekomendasi waktu nyata.

Topik

Ikhtisar pencarian vektor

Fitur ini dalam rilis pratinjau untuk MemoryDB untuk Redis dan dapat berubah.

Pencarian vektor dibangun di atas pembuatan, pemeliharaan, dan penggunaan indeks. Setiap operasi pencarian vektor menentukan indeks tunggal dan operasinya terbatas pada indeks itu, yaitu, operasi pada satu indeks tidak terpengaruh oleh operasi pada indeks lainnya. Kecuali untuk operasi

untuk membuat dan menghancurkan indeks, sejumlah operasi dapat dikeluarkan terhadap indeks apa pun kapan saja, yang berarti bahwa pada tingkat cluster, beberapa operasi terhadap beberapa indeks dapat berlangsung secara bersamaan.

Indeks individu diberi nama objek yang ada di namespace unik, yang terpisah dari ruang nama Redis lainnya: kunci, fungsi, dll. Setiap indeks secara konseptual mirip dengan tabel database konvensional karena terstruktur dalam dua dimensi: kolom dan baris. Setiap baris dalam tabel sesuai dengan Redis Key. Setiap kolom dalam indeks sesuai dengan anggota atau bagian dari kunci itu. Dalam dokumen ini, kunci istilah, baris dan catatan identik dan digunakan secara bergantian. Demikian pula istilah kolom, bidang, jalur dan anggota pada dasarnya identik dan juga digunakan secara bergantian.

Tidak ada perintah khusus untuk menambah, menghapus, atau memodifikasi data yang diindeks. Sebaliknya yang ada HASH atau JSON perintah yang memodifikasi kunci yang ada dalam indeks juga secara otomatis memperbarui indeks.

Topik

Indeks dan ruang kunci Redis

Indeks dibangun dan dipelihara di atas subset dari keyspace Redis. Beberapa indeks dapat memilih subset yang terpisah atau tumpang tindih dari keyspace Redis tanpa batasan. Ruang kunci untuk setiap indeks ditentukan oleh daftar awalan kunci yang disediakan saat indeks dibuat. Daftar awalan adalah opsional dan jika dihilangkan, seluruh ruang kunci Redis akan menjadi bagian dari indeks itu. Indeks juga diketik karena hanya mencakup kunci yang memiliki tipe yang cocok. Saat ini, hanya indeks JSON dan HASH yang didukung. Indeks HASH hanya mengindeks kunci HASH yang dicakup oleh daftar awalannya dan juga indeks JSON hanya mengindeks kunci JSON yang dicakup oleh daftar awalannya. Kunci dalam daftar awalan ruang kunci indeks yang tidak memiliki tipe yang ditentukan diabaikan dan tidak memengaruhi operasi pencarian.

Ketika perintah HASH atau JSON memodifikasi kunci yang berada dalam ruang kunci indeks indeks yang diperbarui. Proses ini melibatkan penggalian bidang yang dideklarasikan untuk setiap indeks

dan memperbarui indeks dengan nilai baru. Proses pembaruan dilakukan di utas latar belakang, yang berarti bahwa indeks pada akhirnya hanya konsisten dengan konten keyspace mereka. Dengan demikian sisipan atau pembaruan kunci tidak akan terlihat di hasil pencarian untuk waktu yang singkat. Selama periode beban sistem yang berat dan/atau mutasi data yang berat, penundaan visibilitas bisa menjadi lebih lama.

Pembuatan indeks adalah proses multi-langkah. Langkah pertama adalah menjalankan perintah FT.CREATE yang mendefinisikan indeks. Eksekusi yang berhasil membuat secara otomatis memulai langkah kedua - penimbunan ulang. Proses pengisian ulang berjalan di thread latar belakang dan memindai ruang kunci Redis mencari kunci yang berada dalam daftar awalan indeks baru. Setiap kunci yang ditemukan ditambahkan ke indeks. Akhirnya seluruh keyspace dipindai, menyelesaikan proses pembuatan indeks. Perhatikan bahwa saat proses pengisian ulang berjalan, mutasi kunci yang diindeks diizinkan, tidak ada batasan dan proses pengisian ulang indeks tidak akan selesai sampai semua kunci diindeks dengan benar. Operasi kueri yang dicoba saat indeks sedang menjalani pengisian ulang tidak diizinkan dan diakhiri dengan kesalahan. Penyelesaian proses backfilling dapat ditentukan dari output `FT.INFO` perintah untuk indeks tersebut ('backfill_status').

Jenis bidang indeks

Setiap bidang (kolom) indeks memiliki jenis tertentu yang dideklarasikan saat indeks dibuat dan lokasi di dalam kunci. Untuk kunci HASH lokasi adalah nama bidang dalam HASH. Untuk kunci JSON lokasi adalah deskripsi jalur JSON. Ketika kunci dimodifikasi, data yang terkait dengan bidang yang dideklarasikan diekstraksi, dikonversi ke tipe yang dideklarasikan dan disimpan dalam indeks. Jika data hilang atau tidak berhasil dikonversi ke tipe yang dideklarasikan, maka bidang tersebut dihilangkan dari indeks. Ada empat jenis bidang, seperti yang dijelaskan sebagai berikut:

- Bidang angka berisi satu nomor. Untuk bidang JSON, aturan numerik nomor JSON harus diikuti. Untuk HASH, bidang ini diharapkan berisi teks ASCII dari angka yang ditulis dalam format standar untuk nomor tetap atau floating point. Terlepas dari representasi dalam kunci, bidang ini dikonversi ke nomor floating point 64-bit untuk penyimpanan dalam indeks. Bidang angka dapat digunakan dengan operator pencarian rentang. Karena angka yang mendasarinya disimpan di floating point dengan batasan presisi, aturan umum tentang perbandingan numerik untuk nomor floating point berlaku.
- Bidang tag berisi nol atau lebih nilai tag yang dikodekan sebagai string UTF-8 tunggal. String diuraikan menjadi nilai tag menggunakan karakter pemisah (default adalah koma tetapi dapat diganti) dengan spasi putih depan dan belakang dihapus. Sejumlah nilai tag dapat terkandung dalam satu bidang tag. Bidang tag dapat digunakan untuk memfilter kueri untuk kesetaraan nilai tag dengan perbandingan case-sensitive atau case-insensitive.

- Bidang teks berisi gumpalan byte yang tidak perlu sesuai dengan UTF-8. Bidang teks dapat digunakan untuk menghias hasil kueri dengan nilai-nilai yang bermakna aplikasi. Misalnya URL atau isi dokumen, dll.
- Bidang vektor berisi vektor angka yang juga dikenal sebagai embedding. Bidang vektor mendukung K-pencarian tetangga terdekat (KNN) dari vektor berukuran tetap menggunakan algoritma dan metrik jarak tertentu. Untuk indeks HASH, bidang harus berisi seluruh vektor yang dikodekan dalam format biner (IEEE 754 endian kecil). Untuk kunci JSON, jalur harus mereferensikan array dengan ukuran yang benar yang diisi dengan angka. Perhatikan bahwa ketika array JSON digunakan sebagai bidang vektor, representasi internal array dalam kunci JSON diubah menjadi format yang diperlukan oleh algoritma yang dipilih, mengurangi konsumsi memori dan presisi. Operasi baca selanjutnya menggunakan perintah JSON akan menghasilkan nilai presisi yang berkurang.

Algoritma indeks vektor

Dua algoritma indeks vektor disediakan:

- Flat — Algoritma Flat adalah pemrosesan linier brute force dari setiap vektor dalam indeks, menghasilkan jawaban yang tepat dalam batas-batas ketepatan perhitungan jarak. Karena pemrosesan linier indeks, waktu berjalan untuk algoritma ini bisa sangat tinggi untuk indeks besar.
- HNSW (Hierarchical Navigable Small Worlds) — Algoritma HNSW adalah alternatif yang memberikan perkiraan jawaban yang benar dengan imbalan waktu eksekusi yang jauh lebih rendah. Algoritma dikendalikan oleh tiga parameter`M`, `EF_CONSTRUCTION` dan`EF_RUNTIME`. Dua parameter pertama ditentukan pada waktu pembuatan indeks dan tidak dapat diubah. `EF_RUNTIME`Parameter memiliki nilai default yang ditentukan pada pembuatan indeks, tetapi dapat diganti pada setiap operasi kueri individu sesudahnya. Ketiga parameter ini berinteraksi untuk menyeimbangkan memori dan konsumsi CPU selama operasi konsumsi dan kueri serta mengontrol kualitas perkiraan pencarian KNN yang tepat (dikenal sebagai rasio recall).

Kedua algoritma pencarian vektor (Flat dan HNSW) mendukung parameter opsional. `INITIAL_CAP` Ketika ditentukan, parameter ini mengalokasikan memori untuk indeks, sehingga mengurangi overhead manajemen memori dan peningkatan tingkat konsumsi vektor.

Algoritma pencarian vektor seperti HNSW mungkin tidak efisien menangani penghapusan atau penimpaan vektor yang dimasukkan sebelumnya. Penggunaan operasi ini dapat mengakibatkan

konsumsi memori indeks berlebih dan/atau penurunan kualitas penarikan. Pengindeksan ulang adalah salah satu metode untuk memulihkan penggunaan dan/atau ingatan memori yang optimal.

Ekspresi kueri pencarian vektor

Perintah FT.SEARCH dan FT.AGGREGATE memerlukan ekspresi kueri. Ekspresi ini adalah parameter string tunggal yang terdiri dari satu atau lebih operator. Setiap operator menggunakan satu bidang dalam indeks untuk mengidentifikasi subset kunci dalam indeks. Beberapa operator dapat digabungkan menggunakan penggabung boolean serta tanda kurung untuk lebih meningkatkan atau membatasi kumpulan kunci yang dikumpulkan (atau set hasil).

Wildcard

Operator wildcard, tanda bintang ('*'), cocok dengan semua kunci dalam indeks.

Rentang numerik

Operator rentang numerik memiliki sintaks berikut:

```
<range-search> ::= '@' <numeric-field-name> ':' '[' <bound> <bound> ']'
<bound>  ::= <number> | '(' <number>
<number> ::= <integer> | <fixed-point> | <floating-point> | 'Inf' | '-Inf' | '+Inf'
```

< numeric-field-name > harus berupa bidang tipe yang dideklarasikan`NUMERIC`. Secara default terikat bersifat inklusif tetapi tanda kurung terbuka terkemuka ['('] dapat digunakan untuk membuat eksklusif terikat. Pencarian rentang dapat dikonversi menjadi perbandingan relasional tunggal (<, <=, >, >=) dengan menggunakan`Inf`, `+Inf` atau `-Inf` sebagai salah satu batas. Terlepas dari format numerik yang ditentukan (integer, fixed-point, floating-point, infinity) jumlahnya diubah menjadi floating point 64-bit untuk melakukan perbandingan, mengurangi presisi yang sesuai.

Example Contoh-contoh

```
@numeric-field:[0 10]                    // 0   <= <value> <= 10
@numeric-field:[(0 10]                   // 0   <  <value> <= 10
@numeric-field:[0 (10]                   // 0   <= <value> <  10
@numeric-field:[(0 (10]                  // 0   <  <value> <  10
@numeric-field:[1.5 (Inf]                // 1.5 <= value
```

Bandingkan tag

Operator perbandingan tag memiliki sintaks berikut:

```
<tag-search> ::= '@' <tag-field-name> ':' '{' <tag> [ '|' <tag> ]* '}'
```

Jika salah satu tag di operator cocok dengan salah satu tag di bidang tag catatan, maka catatan tersebut disertakan dalam kumpulan hasil. Bidang yang dirancang oleh `<tag-field-name>` harus berupa bidang indeks yang dideklarasikan dengan tipe`TAG`. Contoh perbandingan tag adalah:

```
@tag-field:{ atag }
@tag-field: { tag1 | tag2 }
```

Kombinasi Boolean

Set hasil dari operator numerik atau tag dapat digabungkan menggunakan logika boolean: dan/atau. Tanda kurung dapat digunakan untuk mengelompokkan operator dan/atau mengubah urutan evaluasi. Sintaks operator logika boolean adalah:

```
<expression> ::= <phrase> | <phrase> '|' <expression> | '(' <expression> ')'
<phrase> ::= <term> | <term> <phrase>
<term> ::= <range-search> | <tag-search> | '*'
```

Beberapa istilah yang digabungkan menjadi frasa adalah “dan” -ed. Beberapa frasa yang dikombinasikan dengan pipa ('|') adalah “atau” -ed.

Pencarian vektor

Operator pencarian vektor melakukan pencarian tetangga K-terdekat dari indeks bidang vektor. Sintaks pencarian vektor adalah:

```
<vector-search> ::= <expression> '=>[KNN' <k> '@' <vector-field-name> '$' <parameter-
name> <modifiers> ']'
<modifiers> ::= [ 'EF_RUNTIME' <integer> ] [ 'AS' <distance-field-name>]
```

Pencarian vektor hanya diterapkan pada vektor yang memenuhi `<expression>` yang dapat berupa kombinasi dari operator yang didefinisikan di atas: wildcard, pencarian rentang, pencarian tag dan/atau kombinasi boolean daripadanya.

- `<k>`adalah bilangan bulat yang menentukan jumlah vektor tetangga terdekat yang akan dikembalikan.
- `<vector-field-name>`harus menentukan bidang tipe yang dideklarasikan`VECTOR`.

- `<parameter-name>`bidang menentukan salah satu entri untuk `PARAM` tabel perintah `FT.SEARCH` atau`FT.AGGREGATE`. Parameter ini adalah nilai vektor referensi untuk perhitungan jarak. Nilai vektor dikodekan ke dalam `PARAM` nilai dalam format biner IEEE 754 endian kecil (sama dikodekan seperti untuk bidang vektor HASH)
- Untuk indeks vektor tipe HNSW, `EF_RUNTIME` klausa opsional dapat digunakan untuk mengganti nilai default `EF_RUNTIME` parameter yang ditetapkan saat indeks dibuat.
- Opsional `<distance-field-name>` memberikan nama bidang untuk kumpulan hasil yang berisi jarak yang dihitung antara vektor referensi dan kunci yang terletak.

Perintah INFO

Pencarian vektor menambah perintah Redis INFO dengan beberapa bagian statistik dan penghitung tambahan. Permintaan untuk mengambil bagian `SEARCH` akan mengambil semua bagian berikut:

Bagian **`search_memory`**

Nama	Penjelasan
search_used_memory_bytes	Jumlah byte memori yang dikonsumsi di semua struktur data pencarian
search_used_memory_human	Versi yang dapat dibaca manusia di atas

Bagian **`search_index_stats`**

Nama	Penjelasan
search_number_of_indexes	Jumlah indeks yang dibuat
search_num_fulltext_indexes	Jumlah bidang non-vektor di semua indeks
search_num_vector_indexes	Jumlah bidang vektor di semua indeks
search_num_hash_indexes	Jumlah indeks pada kunci tipe HASH
search_num_json_indexes	Jumlah indeks pada kunci tipe JSON

Nama	Penjelasan
search_total_indexed_keys	Jumlah total kunci di semua indeks
search_total_indexed_vectors	Jumlah total vektor di semua indeks
search_total_indexed_hash_keys	Jumlah total kunci tipe HASH di semua indeks
search_total_indexed_json_keys	Jumlah total kunci tipe JSON di semua indeks
search_total_index_size	Byte yang digunakan oleh semua indeks
search_total_fulltext_index_size	Byte yang digunakan oleh struktur indeks non-vektor
search_total_vector_index_size	Byte yang digunakan oleh struktur indeks vektor
search_max_index_lag_ms	Penundaan konsumsi selama pembaruan batch konsumsi terakhir

Bagian **`search_ingestion`**

Nama	Penjelasan
search_background_indexing_status	Status konsumsi. `NO_ACTIVITY` berarti menganggur. Nilai lain menunjukkan ada kunci dalam proses dicerna.
search_ingestion_dijeda	Kecuali saat memulai ulang, ini harus selalu “tidak”.

Bagian **`search_backfill`**

> **Note**
>
> Beberapa bidang yang didokumentasikan di bagian ini hanya terlihat saat pengisian ulang sedang berlangsung.

Nama	Penjelasan
search_num_active_backfills	Jumlah kegiatan pengurukan saat ini
search_backfills_dijeda	Kecuali ketika kehabisan ingatan, ini harus selalu “tidak”.
search_highest_backfill_progress_percentage	% penyelesaian (0-100) dari isi ulang yang paling lengkap
search_lowest_backfill_progress_percentage	% completiton (0-100) dari isi ulang yang paling sedikit selesai

Bagian **`search_query`**

Nama	Penjelasan
search_num_active_queries	Jumlah `FT.SEARCH` dan `FT.AGGREGATE` perintah yang sedang berlangsung

Keamanan pencarian vektor

Redis ACL (Access Control Lists) mekanisme keamanan untuk kedua perintah dan akses data diperluas untuk mengontrol fasilitas pencarian. Kontrol ACL atas perintah pencarian individual didukung sepenuhnya. Kategori ACL baru,`@search`, disediakan dan banyak kategori yang ada (`@fast`,,, `@read` `@write`, dll.) Diperbarui untuk menyertakan perintah baru. Perintah pencarian tidak memodifikasi data kunci, artinya mesin ACL yang ada untuk akses tulis dipertahankan. Aturan akses

untuk operasi HASH dan JSON tidak dimodifikasi oleh kehadiran indeks; kontrol akses tingkat kunci normal masih diterapkan pada perintah tersebut.

Perintah pencarian dengan indeks juga memiliki akses mereka dikendalikan melalui Redis ACL. Pemeriksaan akses dilakukan pada tingkat seluruh indeks, bukan pada tingkat per-kunci. Ini berarti bahwa akses ke indeks diberikan kepada pengguna hanya jika pengguna tersebut memiliki izin untuk mengakses semua kunci yang mungkin dalam daftar awalan keyspace indeks tersebut. Dengan kata lain, konten sebenarnya dari indeks tidak mengontrol akses. Sebaliknya, itu adalah isi teoritis dari indeks seperti yang didefinisikan oleh daftar awalan yang digunakan untuk pemeriksaan keamanan. Hal ini dapat mudah untuk membuat situasi di mana pengguna telah membaca dan/atau menulis akses ke kunci tetapi tidak dapat mengakses indeks yang berisi kunci itu. Perhatikan bahwa hanya akses baca ke ruang kunci yang diperlukan untuk membuat atau menggunakan indeks - ada atau tidak adanya akses tulis tidak dipertimbangkan.

Untuk informasi selengkapnya tentang menggunakan ACL dengan MemoryDB, lihat Mengautentikasi pengguna dengan Daftar Kontrol Akses (ACL).

Fitur pencarian vektor dan batas

> Fitur ini dalam rilis pratinjau untuk MemoryDB untuk Redis dan dapat berubah.

Ketersediaan pencarian vektor

Konfigurasi MemoryDB berkemampuan pencarian vektor didukung pada tipe node R6g, R7g, dan T4G dan tersedia di AWS Wilayah berikut: US East (Virginia N.), US East (Ohio), US West (Oregon), Asia Pasifik (Tokyo), dan Eropa (Irlandia).

Pembatasan parametrik

Tabel berikut menunjukkan batas untuk berbagai item pencarian vektor di pratinjau:

Item	Nilai maksimum
Jumlah dimensi dalam vektor	32768
Jumlah indeks yang dapat dibuat	10

Item	Nilai maksimum
Jumlah bidang dalam indeks	50
Jumlah operasi pengisian ulang FT.CREATE indeks simultan	1
Klausa FT.SEARCH dan FT.AGGREGATE TIMEOUT (MilliSeconds)	60000
Jumlah tahapan pipeline dalam perintah FT.AGGREGATE	32
Jumlah bidang dalam klausa FT.AGGREGATE LOAD	1024
Jumlah bidang dalam klausa FT.AGGREGATE GROUPBY	16
Jumlah bidang dalam klausa FT.AGGREGATE SORTBY	16
Jumlah parameter dalam klausa FT.AGGREGATE PARAM	32
Parameter HNSW M	512
Parameter HNSW EF_CONSTRUCTION	4096
Parameter HNSW EF_RUNTIME	4096

Batas penskalaan

Pencarian vektor untuk MemoryDB saat ini terbatas pada pecahan tunggal dan penskalaan horizontal tidak didukung. Pencarian vektor mendukung penskalaan vertikal dan replika.

Pembatasan operasional

Indeks Persistensi dan Penimbunan Ulang

Pratinjau pencarian vektor mempertahankan definisi indeks, tetapi bukan kontennya. Jadi setiap permintaan operasional atau peristiwa yang menyebabkan node memulai atau memulai ulang mengharuskan semua indeks dibangun kembali dari definisi dan data kunci sumbernya. Proses rebuild dimulai secara otomatis setelah semua data dipulihkan — tidak ada tindakan pengguna yang diperlukan untuk memulai ini. Pembangunan kembali dilakukan sebagai operasi pengurukan segera setelah data dipulihkan. Ini secara fungsional setara dengan sistem yang secara otomatis menjalankan perintah FT.CREATE untuk setiap indeks yang ditentukan. Perhatikan bahwa node menjadi tersedia untuk operasi aplikasi segera setelah data dipulihkan tetapi kemungkinan sebelum pengurukan indeks selesai, yang berarti bahwa isi ulang akan kembali terlihat oleh aplikasi, misalnya, perintah pencarian menggunakan indeks backfilling dapat ditolak. Untuk informasi lebih lanjut tentang penimbunan ulang, lihatIkhtisar pencarian vektor.

Penyelesaian isi ulang indeks tidak disinkronkan antara primer dan replika. Kurangnya sinkronisasi ini secara tak terduga dapat terlihat oleh aplikasi dan oleh karena itu disarankan agar aplikasi memverifikasi penyelesaian pengisian ulang pada primer dan semua replika sebelum memulai operasi pencarian.

Impor/ekspor snapshot dan Migrasi Langsung

Kehadiran indeks pencarian dalam file RDB membatasi transportabilitas yang kompatibel dari data tersebut. Format indeks yang ditentukan oleh edisi pratinjau hanya dipahami oleh kluster edisi pratinjau lainnya. Dengan demikian file RDB dengan indeks pencarian hanya dapat ditransfer antara atau digunakan oleh kluster MemoryDB yang diaktifkan pratinjau.

Namun, file RDB yang tidak mengandung indeks tidak dibatasi dengan cara ini. Dengan demikian data dalam klaster pratinjau dapat diekspor ke kluster non-pratinjau dengan menghapus indeks sebelum ekspor.

Konsumsi memori

Implementasi indeks vektor saat ini mengkonsumsi kira-kira dua kali jumlah memori seperti yang akan dikonsumsi oleh implementasi Ketersediaan Umum.

Keluar dari Memori saat mengisi ulang

Mirip dengan operasi tulis Redis, isi ulang indeks dikenakan batasan. out-of-memory Jika memori Redis terisi saat pengisian ulang sedang berlangsung, semua isi ulang dijeda. Jika memori tersedia, proses pengurukan dilanjutkan. Dimungkinkan juga untuk menghapus dan mengindeks saat pengisian ulang dijeda karena kehabisan memori.

Transaksi

PerintahFT.CREATE,,FT.DROPINDEX, FT.ALIASADDFT.ALIASDEL, dan FT.ALIASUPDATE tidak dapat dieksekusi dalam konteks transaksional, yaitu, tidak dalam blok MULTI/EXEC atau dalam skrip LUA atau FUNGSI.

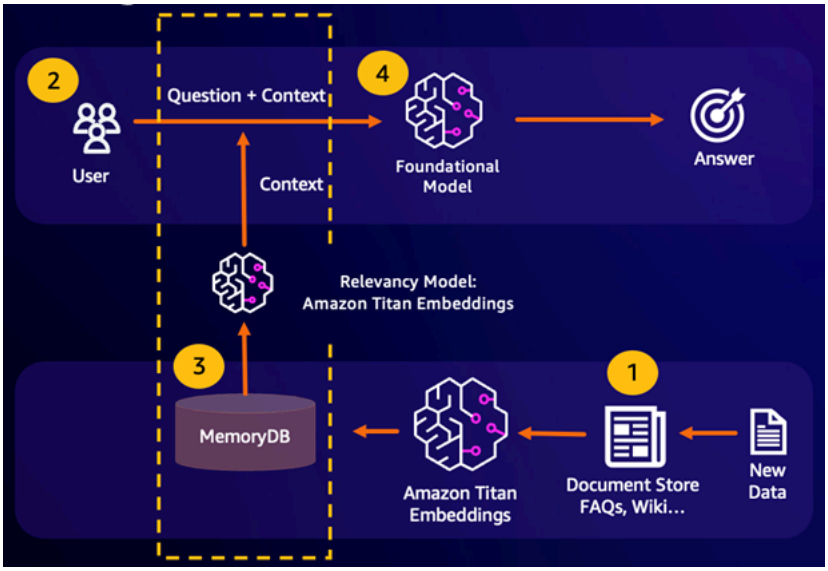
Kasus penggunaan

> Fitur ini dalam rilis pratinjau untuk MemoryDB untuk Redis dan dapat berubah.

Berikut ini adalah kasus penggunaan pencarian vektor.

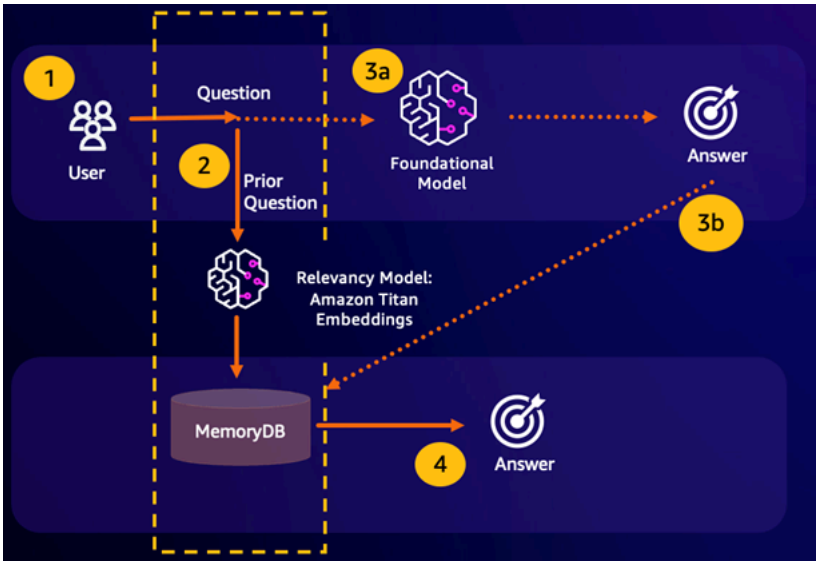
Retrieval Augmented Generation (RAG)

Retrieval Augmented Generation (RAG) memanfaatkan pencarian vektor untuk mengambil bagian yang relevan dari kumpulan data yang besar untuk menambah model bahasa besar (LLM). Secara khusus, encoder menyematkan konteks input dan kueri penelusuran ke dalam vektor, kemudian menggunakan perkiraan pencarian tetangga terdekat untuk menemukan bagian yang serupa secara semantik. Bagian yang diambil ini digabungkan dengan konteks asli untuk memberikan informasi tambahan yang relevan ke LLM untuk mengembalikan respons yang lebih akurat kepada pengguna.



Memori Penyangga Model Foundation (FM)

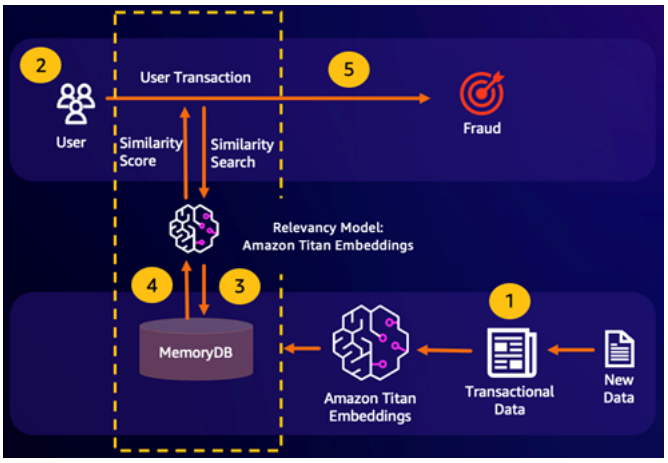
Memori buffer model Foundation (FM) adalah proses untuk mengurangi biaya komputasi dengan menyimpan hasil sebelumnya dari FM. Dengan menggunakan kembali hasil sebelumnya dari kesimpulan sebelumnya alih-alih mengkomputernya kembali, memori buffer FM mengurangi jumlah perhitungan yang diperlukan selama inferensi melalui FM. Memori buffer FM ini memungkinkan model bahasa besar merespons lebih cepat dengan biaya lebih rendah karena biaya layanan dari FM.



- Pencarian semantik — Jika kueri pelanggan semantik serupa berdasarkan skor kesamaan yang ditentukan dengan pertanyaan sebelumnya, memori buffer FM (MemoryDB) akan mengembalikan jawaban ke pertanyaan sebelumnya di langkah 4 dan tidak akan memanggil FM melalui langkah 3. Ini akan menghindari latensi model pondasi (FM) dan biaya yang dikeluarkan, memberikan pengalaman yang lebih cepat bagi pelanggan.
- Kehilangan pencarian semantik — Jika kueri pelanggan tidak semantik serupa berdasarkan skor kesamaan yang ditentukan dengan kueri sebelumnya, pelanggan akan menghubungi FM untuk memberikan respons kepada pelanggan pada langkah 3a. Respon yang dihasilkan dari FM kemudian akan disimpan sebagai vektor ke dalam MemoryDB untuk kueri future (langkah 3b) untuk meminimalkan biaya FM pada pertanyaan semantik serupa. Dalam alur ini, langkah 4 tidak akan dipanggil karena tidak ada pertanyaan semantik serupa untuk kueri asli.

Deteksi penipuan

Deteksi penipuan, suatu bentuk deteksi anomali, mewakili transaksi yang valid sebagai vektor sambil membandingkan representasi vektor dari transaksi baru bersih. Penipuan terdeteksi ketika transaksi baru bersih ini memiliki kesamaan rendah dengan vektor yang mewakili data transaksional yang valid. Hal ini memungkinkan penipuan dideteksi dengan memodelkan perilaku normal, daripada mencoba memprediksi setiap kemungkinan kejadian penipuan. MemoryDB memungkinkan organisasi untuk melakukan ini dalam periode throughput tinggi, dengan positif palsu minimal dan latensi milidetik satu digit.



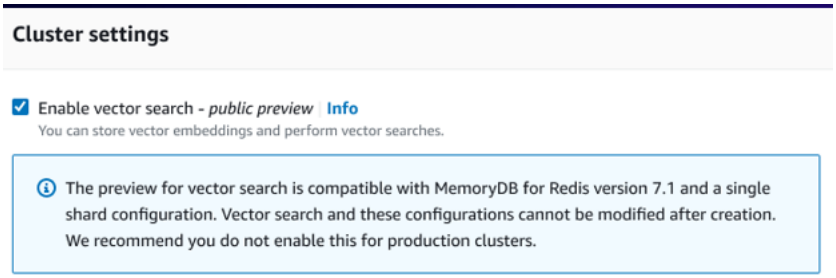
Kasus penggunaan lainnya

- Mesin rekomendasi dapat menemukan pengguna produk atau konten serupa dengan mewakili item sebagai vektor. Vektor dibuat dengan menganalisis atribut dan pola. Berdasarkan pola dan atribut pengguna, item baru yang tidak terlihat dapat direkomendasikan kepada pengguna dengan menemukan vektor paling mirip yang telah dinilai secara positif selaras dengan pengguna.
- Mesin pencari dokumen mewakili dokumen teks sebagai vektor angka yang padat, menangkap makna semantik. Pada waktu pencarian, mesin mengubah permintaan pencarian ke vektor dan menemukan dokumen dengan vektor yang paling mirip dengan kueri menggunakan perkiraan pencarian tetangga terdekat. Pendekatan kesamaan vektor ini memungkinkan pencocokan dokumen berdasarkan makna daripada hanya mencocokkan kata kunci.

Menggunakan AWS Management Console

Fitur ini dalam rilis pratinjau untuk MemoryDB untuk Redis dan dapat berubah.

Untuk membuat cluster yang diaktifkan untuk pencarian vektor di dalam konsol, Anda perlu mengaktifkan pencarian vektor di bawah pengaturan Cluster. Pencarian vektor tersedia untuk MemoryDB untuk Redis versi 7.1 dan konfigurasi pecahan tunggal.



Untuk informasi lebih lanjut tentang menggunakan pencarian vektor dengan AWS Management Console, lihatMembuat cluster (Konsol).

Menggunakan AWS Command Line Interface

Fitur ini dalam rilis pratinjau untuk MemoryDB untuk Redis dan dapat berubah.

Untuk membuat klaster MemoryDB yang diaktifkan pencarian vektor, Anda dapat menggunakan perintah create-cluster MemoryDB dengan meneruskan grup parameter yang tidak dapat diubah untuk mengaktifkan mode pratinjau `default.memorydb-redis7.search.preview` untuk kemampuan pencarian vektor.

```
aws memorydb create-cluster \
  --cluster-name <value> \
  --node-type <value> \
  --engine redis \
  --engine-version 7.1 \
  --num-shards 1 \
  --acl-name <value> \
  --parameter-group-name default.memorydb-redis7.search.preview
```

Perintah pencarian vektor

Berikut ini adalah daftar perintah yang didukung untuk pencarian vektor.

Topik

- FT.CREATE
- FT.SEARCH
- FT.AGREGAT
- FT.DROPINDEX
- FT.INFO
- KAKI. _DAFTAR
- FT.ALIASADD
- FT.ALIASDEL
- FT.ALIASUPDATE
- KAKI. _ALIASLIST
- FT.CONFIG MENDAPATKAN
- BANTUAN FT.CONFIG
- FT.CONFIG SET
- FT.PROFIL
- FT.JELASKAN
- FT.EXPLAINCLI

FT.CREATE

Membuat indeks dan memulai pengurukan indeks itu. Untuk informasi selengkapnya, lihat Ikhtisar pencarian vektor untuk detail tentang konstruksi indeks.

Sintaksis

```
FT.CREATE <index-name>
ON HASH | JSON
[PREFIX <count> <prefix1> [<prefix2>...]]
SCHEMA
(<field-identifier> [AS <alias>]
  NUMERIC
| TAG [SEPARATOR <sep>] [CASESENSITIVE]
| TEXT
| VECTOR [HNSW|FLAT] <attr_count> [<attribute_name> <attribute_value>])

)+
```

Skema

- Pengidentifikasi bidang:
  - Untuk kunci hash, pengenal bidang adalah nama bidang.
  - Untuk kunci JSON, pengenal bidang adalah jalur JSON.

  Untuk informasi selengkapnya, lihat Jenis bidang indeks.

- Jenis bidang:
  - TAG: Untuk informasi selengkapnya, lihat Tag.
  - NUMERIK: Bidang berisi angka.
  - TEXT: Bidang berisi gumpalan data.
  - VECTOR: bidang vektor yang mendukung pencarian vektor.
    - Algoritma — dapat berupa HNSW (Hierarchical Navigable Small World) atau FLAT (brute force).
    - `attr_count`— jumlah atribut yang akan diteruskan sebagai konfigurasi algoritma, ini termasuk nama dan nilai.
    - `{attribute_name} {attribute_value}`— pasangan kunci/nilai khusus algoritma yang menentukan konfigurasi indeks.

      Untuk algoritma FLAT, atribut adalah:

      Diperlukan:

      - DIM — Jumlah dimensi dalam vektor.
      - DISTANCE_METRIC — Bisa menjadi salah satu [L2 | IP | COSINE].
      - JENIS - Jenis vektor. Satu-satunya jenis yang didukung adalah`FLOAT32`.

      Opsional:

      - INITIAL_CAP — Kapasitas vektor awal dalam indeks yang mempengaruhi ukuran alokasi memori indeks.

      Untuk algoritma HNSW, atributnya adalah:

      Diperlukan:

      - JENIS - Jenis vektor. Satu-satunya jenis yang didukung adalah`FLOAT32`.

- DIM — Dimensi vektor, ditentukan sebagai bilangan bulat positif. Maksimal: 32768
- DISTANCE_METRIC — Bisa menjadi salah satu [L2 | IP | COSINE].

Opsional:

- INITIAL_CAP — Kapasitas vektor awal dalam indeks yang mempengaruhi ukuran alokasi memori indeks. Default ke 1024.
- M — Jumlah tepi keluar maksimum yang diizinkan untuk setiap node dalam grafik di setiap lapisan. pada lapisan nol jumlah maksimal tepi keluar adalah 2M. Defaultnya adalah 16 Maksimum adalah 512.
- EF_CONSTRUCTION — mengontrol jumlah vektor yang diperiksa selama konstruksi indeks. Nilai yang lebih tinggi untuk parameter ini akan meningkatkan rasio penarikan dengan mengorbankan waktu pembuatan indeks yang lebih lama. Nilai default adalah 200. Nilai maksimum adalah 4096.
- EF_RUNTIME — mengontrol jumlah vektor yang diperiksa selama operasi kueri. Nilai yang lebih tinggi untuk parameter ini dapat menghasilkan penarikan yang lebih baik dengan mengorbankan waktu kueri yang lebih lama. Nilai parameter ini dapat diganti berdasarkan per-query. Nilai default adalah 10. Nilai maksimum adalah 4096.

Nilai yang ditampilkan

Mengembalikan string pesan OK sederhana atau balasan kesalahan.

Contoh

> **Note**
>
> Contoh berikut menggunakan argumen asli redis-cli, seperti de-quoting dan de-escaping data, sebelum mengirimnya ke Redis. Untuk menggunakan klien bahasa pemrograman lainnya (Python, Ruby, C #, dll.), Ikuti aturan penanganan lingkungan tersebut untuk menangani string dan data biner. Untuk informasi selengkapnya tentang klien yang didukung, lihat Alat untuk Dibangun AWS

Example 1: Buat beberapa indeks

Buat indeks untuk vektor ukuran 2

```
FT.CREATE hash_idx1 ON HASH PREFIX 1 hash: SCHEMA vec AS VEC VECTOR HNSW 6 DIM 2 TYPE
 FLOAT32 DISTANCE_METRIC L2
OK
```

Buat indeks JSON 6 dimensi menggunakan algoritma HNSW:

```
FT.CREATE json_idx1 ON JSON PREFIX 1 json: SCHEMA $.vec AS VEC VECTOR HNSW 6 DIM 6 TYPE
 FLOAT32 DISTANCE_METRIC L2
OK
```

Example Contoh 2: Mengisi beberapa data

Perintah berikut diformat sehingga dapat dieksekusi sebagai argumen untuk program terminal redis-cli. Pengembang yang menggunakan klien bahasa pemrograman (seperti Python, Ruby, C #, dll.) Perlu mengikuti aturan penanganan lingkungan mereka untuk menangani string dan data biner.

Membuat beberapa data hash dan json:

```
HSET hash:0 vec "\x00\x00\x00\x00\x00\x00\x00\x00"
HSET hash:1 vec "\x00\x00\x00\x00\x00\x00\x80\xbf"
JSON.SET json:0 . '{"vec":[1,2,3,4,5,6]}'
JSON.SET json:1 . '{"vec":[10,20,30,40,50,60]}'
JSON.SET json:2 . '{"vec":[1.1,1.2,1.3,1.4,1.5,1.6]}'
```

Perhatikan hal berikut:

- Kunci data hash dan JSON memiliki awalan definisi indeksnya.
- Vektor berada pada jalur yang sesuai dari definisi indeks.
- Vektor hash dimasukkan sebagai data hex sedangkan data JSON dimasukkan sebagai angka.
- Vektor adalah panjang yang sesuai, entri vektor hash dua dimensi memiliki data hex senilai dua float, entri vektor json enam dimensi memiliki enam angka.

Example Contoh 3: Hapus dan buat ulang indeks

```
FT.DROPINDEX json_idx1
OK

FT.CREATE json_idx1 ON JSON PREFIX 1 json: SCHEMA $.vec AS VEC VECTOR FLAT 6 DIM 6 TYPE
 FLOAT32 DISTANCE_METRIC L2
```

```
OK
```

Perhatikan indeks JSON baru menggunakan `FLAT` algoritma bukan `HNSW` algoritma. Perhatikan juga bahwa itu akan mengindeks ulang data JSON yang ada:

```
FT.SEARCH json_idx1 "*=>[KNN 100 @VEC $query_vec]" PARAMS 2 query_vec
 "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
 DIALECT 2
1) (integer) 3
2) "json:2"
3) 1) "__VEC_score"
   2) "11.11"
   3) "$"
   4) "[{\"vec\":[1.1, 1.2, 1.3, 1.4, 1.5, 1.6]}]"
4) "json:0"
5) 1) "__VEC_score"
   2) "91"
   3) "$"
   4) "[{\"vec\":[1.0, 2.0, 3.0, 4.0, 5.0, 6.0]}]"
6) "json:1"
7) 1) "__VEC_score"
   2) "9100"
   3) "$"
   4) "[{\"vec\":[10.0, 20.0, 30.0, 40.0, 50.0, 60.0]}]"
```

FT.SEARCH

Menggunakan ekspresi kueri yang disediakan untuk menemukan kunci dalam indeks. Setelah ditemukan, hitungan dan/atau konten bidang yang diindeks dalam kunci tersebut dapat dikembalikan. Untuk informasi selengkapnya, lihat Ekspresi kueri penelusuran vektor.

Untuk membuat data untuk digunakan dalam contoh ini, lihat perintah FT.CREATE.

Sintaksis

```
FT.SEARCH <index-name> <query>
[RETURN <token_count> (<field-identifier> [AS <alias>])+]
[TIMEOUT timeout]
[PARAMS <count> <name> <value> [<name> <value>]]
[LIMIT <offset> <count>]
[COUNT]
```

- RETURN: Klausa ini mengidentifikasi bidang kunci mana yang dikembalikan. Klausa AS opsional pada setiap bidang mengesampingkan nama bidang dalam hasil. Hanya bidang yang telah dideklarasikan untuk indeks ini yang dapat ditentukan.
- LIMIT <offset><count>:: Klausa ini memberikan kemampuan pagination karena hanya kunci yang memenuhi nilai offset dan count yang dikembalikan. Jika klausa ini dihilangkan, defaultnya adalah “LIMIT 0 10", yaitu, hanya maksimal 10 kunci yang akan dikembalikan.
- PARAMS: Dua kali jumlah pasangan nilai kunci. Pasangan kunci/nilai Param dapat direferensikan dari dalam ekspresi kueri. Untuk informasi selengkapnya, lihat Ekspresi kueri penelusuran vektor.
- COUNT: Klausa ini menekan pengembalian isi kunci, hanya jumlah kunci yang dikembalikan. Ini adalah alias untuk “LIMIT 0 0".

Nilai yang ditampilkan

Mengembalikan array atau balasan kesalahan.

- Jika operasi selesai dengan sukses, ia mengembalikan array. Elemen pertama adalah jumlah total kunci yang cocok dengan query. Elemen yang tersisa adalah pasangan nama kunci dan daftar bidang. Daftar bidang adalah array lain yang terdiri dari pasangan nama bidang dan nilai.
- Jika indeks sedang dalam proses diisi kembali, perintah segera mengembalikan balasan kesalahan.
- Jika batas waktu tercapai, perintah mengembalikan balasan kesalahan.

Contoh: Lakukan beberapa pencarian

> **Note**
>
> Contoh berikut menggunakan argumen asli redis-cli, seperti de-quoting dan de-escaping data, sebelum mengirimnya ke Redis. Untuk menggunakan klien bahasa pemrograman lainnya (Python, Ruby, C #, dll.), Ikuti aturan penanganan lingkungan tersebut untuk menangani string dan data biner. Untuk informasi selengkapnya tentang klien yang didukung, lihat Alat untuk Dibangun AWS

Pencarian hash

```
FT.SEARCH hash_idx1 "*=>[KNN 2 @VEC $query_vec]" PARAMS 2 query_vec
 "\x00\x00\x00\x00\x00\x00\x00\x00" DIALECT 2
```

```
1) (integer) 2
2) "hash:0"
3) 1) "__VEC_score"
   2) "0"
   3) "vec"
   4) "\x00\x00\x00\x00\x00\x00\x00\x00"
4) "hash:1"
5) 1) "__VEC_score"
   2) "1"
   3) "vec"
   4) "\x00\x00\x00\x00\x00\x00\x80\xbf"
```

Ini menghasilkan dua hasil, diurutkan berdasarkan skornya, yaitu jarak dari vektor kueri (dimasukkan sebagai hex).

Pencarian JSON

```
FT.SEARCH json_idx1 "*=>[KNN 2 @VEC $query_vec]" PARAMS 2 query_vec
 "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
 DIALECT 2
1) (integer) 2
2) "json:2"
3) 1) "__VEC_score"
   2) "11.11"
   3) "$"
   4) "[{\"vec\":[1.1, 1.2, 1.3, 1.4, 1.5, 1.6]}]"
4) "json:0"
5) 1) "__VEC_score"
   2) "91"
   3) "$"
   4) "[{\"vec\":[1.0, 2.0, 3.0, 4.0, 5.0, 6.0]}]"
```

Ini menghasilkan dua hasil terdekat, diurutkan berdasarkan skornya, dan perhatikan bahwa nilai vektor JSON dikonversi menjadi float dan vektor kueri masih merupakan data vektor. Perhatikan juga bahwa karena KNN parameternya 2, hanya ada dua hasil. Nilai yang lebih besar akan mengembalikan lebih banyak hasil:

```
FT.SEARCH json_idx1 "*=>[KNN 100 @VEC $query_vec]" PARAMS 2 query_vec
 "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
 DIALECT 2
1) (integer) 3
2) "json:2"
```

```
3) 1) "__VEC_score"
   2) "11.11"
   3) "$"
   4) "[{\"vec\":[1.1, 1.2, 1.3, 1.4, 1.5, 1.6]}]"
4) "json:0"
5) 1) "__VEC_score"
   2) "91"
   3) "$"
   4) "[{\"vec\":[1.0, 2.0, 3.0, 4.0, 5.0, 6.0]}]"
6) "json:1"
7) 1) "__VEC_score"
   2) "9100"
   3) "$"
   4) "[{\"vec\":[10.0, 20.0, 30.0, 40.0, 50.0, 60.0]}]"
```

FT.AGREGAT

Sebuah superset dari perintah FT.SEARCH, memungkinkan pemrosesan tambahan substansif dari kunci yang dipilih oleh ekspresi kueri.

Sintaksis

```
FT.AGGREGATE index query
  [LOAD * | [count field [field ...]]]
  [TIMEOUT timeout]
  [PARAMS count name value [name value ...]]
  [FILTER expression]
  [LIMIT offset num]
  [GROUPBY count property [property ...] [REDUCE function count arg [arg ...] [AS name]
 [REDUCE function count arg [arg ...] [AS name] ...]] ...]]
  [SORTBY count [ property ASC | DESC [property ASC | DESC ...]] [MAX num]]
  [APPLY expression AS name]
```

- Klausa FILTER, LIMIT, GROUPBY, SORTBY, dan APPLY dapat diulang beberapa kali dalam urutan apa pun dan dicampur secara bebas. Mereka diterapkan dalam urutan yang ditentukan dengan output dari satu klausa yang memberi masukan dari klausa berikutnya.
- Dalam sintaks di atas, “properti” adalah bidang yang dideklarasikan dalam perintah FT.CREATE untuk indeks ini ATAU output dari klausa APPLY sebelumnya atau fungsi REDUCE.
- Klausa LOAD dibatasi untuk memuat bidang yang telah dideklarasikan dalam indeks. “LOAD*” akan memuat semua bidang yang dideklarasikan dalam indeks.

- Fungsi peredam berikut didukung: COUNT, COUNT_DISTINCTISH, SUM, MIN, MAX, AVG, STDDEV, QUANTILE, TOLIST, FIRST_VALUE, dan RANDOM_SAMPLE. Untuk informasi selengkapnya, lihat Agregasi
- BATAS <offset><count>: Mempertahankan catatan mulai <offset>dan berlanjut hingga<count>, semua catatan lainnya dibuang.
- PARAMS: Dua kali jumlah pasangan nilai kunci. Pasangan kunci/nilai Param dapat direferensikan dari dalam ekspresi kueri. Untuk informasi selengkapnya, lihat Ekspresi kueri penelusuran vektor.

Nilai yang ditampilkan

Mengembalikan array atau balasan kesalahan.

- Jika operasi selesai dengan sukses, ia mengembalikan array. Elemen pertama adalah bilangan bulat tanpa arti tertentu (harus diabaikan). Elemen yang tersisa adalah hasil output pada tahap terakhir. Setiap elemen adalah array nama bidang dan pasangan nilai.
- Jika indeks sedang dalam proses diisi kembali, perintah segera mengembalikan balasan kesalahan.
- Jika batas waktu tercapai, perintah mengembalikan balasan kesalahan.

FT.DROPINDEX

Jatuhkan indeks. Definisi indeks dan konten terkait dihapus. Kunci Redis tidak terpengaruh.

Sintaksis

```
FT.DROPINDEX <index-name>
```

Nilai yang ditampilkan

Mengembalikan string pesan OK sederhana atau balasan kesalahan.

FT.INFO

Sintaksis

```
FT.INFO <index-name>
```

Output dari halaman FT.INFO adalah array pasangan nilai kunci seperti yang dijelaskan dalam tabel berikut:

Kunci	Tipe nilai	Deskripsi
index_name	string	Nama indeks
creation_timestamp	integer	Stempel waktu pembuatan bergaya Unix
key_type	string	HASH atau JSON
key_prefixes	array string	Awalan kunci untuk indeks ini
ladang	array informasi lapangan	Bidang indeks ini
space_usage	integer	Byte memori yang digunakan oleh indeks ini
fullext_space_usage	integer	Byte memori yang digunakan oleh bidang non-vektor
vector_space_usage	integer	Byte memori yang digunakan oleh bidang vektor
num_docs	integer	Jumlah kunci yang saat ini terdapat dalam indeks
num_indexed_vectors	integer	Jumlah vektor yang saat ini terdapat dalam indeks
current_lag	integer	Penundaan konsumsi terbaru (MilliSeconds)
backfill_status	string	Salah satu dari: Selesai, InProgres, Dijeda atau Gagal

Tabel berikut menjelaskan informasi untuk setiap bidang:

Kunci	Tipe nilai	Deskripsi
pengenal	string	nama bidang
field_name	string	Nama anggota hash atau JSON Path
tipe	string	salah satu dari: Numerik, Tag, Teks atau Vektor
pilihan	string	abaikan

Jika bidangnya bertipe Vector, informasi tambahan akan hadir tergantung pada algoritma.

Untuk algoritma HNSW:

Kunci	Tipe nilai	Deskripsi
algoritma	string	HNSW
data_type	string	MENGAPUNG32
distance_metric	string	salah satu dari: L2, IP atau Cosine
initial_capacity	integer	Ukuran awal indeks bidang vektor
kapasitas saat ini	integer	Ukuran indeks bidang vektor saat ini
maximum_edges	integer	Parameter M saat pembuatan
ef_konstruksi	integer	Parameter EF_CONSTR UCTION saat pembuatan
ef_runtime	integer	Parameter EF_RUNTIME saat pembuatan

Untuk algoritma FLAT:

Kunci	Tipe nilai	Deskripsi
algoritma	string	DATAR
data_type	string	MENGAPUNG32
distance_metric	string	salah satu dari: L2, IP atau Cosine
initial_capacity	integer	Ukuran awal indeks bidang vektor
kapasitas saat ini	integer	Ukuran indeks bidang vektor saat ini

KAKI. _DAFTAR

Daftar semua indeks.

Sintaksis

```
FT._LIST
```

Nilai yang ditampilkan

Mengembalikan array nama indeks

FT.ALIASADD

Tambahkan alias untuk indeks. Nama alias baru dapat digunakan di mana saja yang diperlukan nama indeks.

Sintaksis

```
FT.ALIASADD <alias> <index-name>
```

Nilai yang ditampilkan

Mengembalikan string pesan OK sederhana atau balasan kesalahan.

FT.ALIASDEL

Hapus alias yang ada untuk indeks.

Sintaksis

```
FT.ALIASDEL <alias>
```

Nilai yang ditampilkan

Mengembalikan string pesan OK sederhana atau balasan kesalahan.

FT.ALIASUPDATE

Perbarui alias yang ada untuk menunjuk ke indeks fisik yang berbeda. Perintah ini hanya mempengaruhi referensi future ke alias. Operasi yang sedang berlangsung (FT.SEARCH, FT.AGGREGATE) tidak terpengaruh oleh perintah ini.

Sintaksis

```
FT.ALIASUPDATE <alias> <index>
```

Nilai yang ditampilkan

Mengembalikan string pesan OK sederhana atau balasan kesalahan.

KAKI. _ALIASLIST

Daftar alias indeks.

Sintaksis

```
FT._ALIASLIST
```

Nilai yang ditampilkan

Mengembalikan array ukuran jumlah alias saat ini. Setiap elemen dari array adalah pasangan alias-index.

FT.CONFIG MENDAPATKAN

Mengembalikan nilai parameter TIMEOUT.

Sintaksis

```
FT.CONFIG GET [* | <timeout>]
```

Nilai yang ditampilkan

Mengembalikan nilai parameter TIMEOUT.

BANTUAN FT.CONFIG

Ambil informasi tentang parameter TIMEOUT.

Sintaksis

```
FT.CONFIG HELP [* | <timeout>]
```

Nilai yang ditampilkan

Mengembalikan informasi pada parameter TIMEOUT

FT.CONFIG SET

Atur parameter TIMEOUT. Nilai defaultnya adalah 10.000 milidetik.

> **Note**
>
> Nama parameter yang dapat dikonfigurasi tidak peka huruf besar/kecil.

Sintaksis

```
FT.CONFIG SET <timeout> <value>
```

Nilai yang ditampilkan

Mengembalikan nilai pengaturan TIMEOUT.

FT.PROFIL

Jalankan kueri dan kembalikan informasi profil tentang kueri itu.

Sintaksis

```
FT.PROFILE

<index>
SEARCH | AGGREGATE
[LIMITED]
QUERY <query ....>
```

Nilai yang ditampilkan

Sebuah array dua elemen. Elemen pertama adalah hasil dari `FT.AGGREGATE` perintah `FT.SEARCH` atau yang diprofilkan. Elemen kedua adalah array informasi kinerja dan profil.

FT.JELASKAN

Mengurai kueri dan mengembalikan informasi tentang bagaimana kueri itu diurai.

Sintaksis

```
FT.EXPLAIN <index> <query>
```

Nilai yang ditampilkan

String yang berisi hasil yang diuraikan.

FT.EXPLAINCLI

Sama seperti perintah FT.EXPLY kecuali bahwa hasilnya ditampilkan dalam format berbeda yang lebih berguna dengan redis-cli.

Sintaksis

```
FT.EXPLAINCLI <index> <query>
```

Nilai yang ditampilkan

String yang berisi hasil yang diuraikan.

Keamanan di MemoryDB untuk Redis

Keamanan cloud di AWS merupakan prioritas tertinggi. Sebagai pelanggan AWS, Anda akan mendapatkan manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan organisasi yang paling sensitif terhadap keamanan.

Keamanan adalah tanggung jawab bersama antara AWS dan Anda. Model tanggung jawab bersama menjelaskan hal ini sebagai keamanan cloud dan keamanan dalam cloud:

- Keamanan cloud – AWS bertanggung jawab untuk melindungi infrastruktur yang menjalankan layanan-layanan AWS di dalam AWS Cloud. AWS juga memberikan Anda layanan yang dapat digunakan dengan aman. Auditor pihak ketiga melakukan pengujian dan verifikasi secara berkala terhadap efektivitas keamanan kami sebagai bagian dari Program Kepatuhan AWS. Untuk mempelajari tentang program kepatuhan yang berlaku di MemoryDB, lihat AWSLayanan dalam Cakupan melalui Program KepatuhanAWS .
- Keamanan dalam cloud – Tanggung jawab Anda ditentukan oleh layanan AWS yang Anda gunakan. Anda juga bertanggung jawab atas faktor lain, mencakup kepekaan data Anda, persyaratan perusahaan, serta peraturan perundangan yang berlaku

Dokumentasi ini membantu Anda memahami cara menerapkan model tanggung jawab bersama saat menggunakan MemoryDB untuk Redis. Dokumentasi ini juga menunjukkan kepada Anda cara mengonfigurasi MemoryDB untuk memenuhi tujuan keamanan dan kepatuhan. Anda juga mempelajari cara menggunakanAWS layanan lain yang membantu Anda memantau dan mengamankan sumber daya MemoryDB Anda.

Daftar Isi

Perlindungan data di MemoryDB untuk Redis

Model tanggung jawab bersama AWS berlaku pada perlindungan data di . Sebagaimana dijelaskan dalam model ini, AWS bertanggung jawab untuk melindungi infrastruktur global yang menjalankan semua AWS Cloud. Anda harus bertanggung jawab untuk memelihara kendali terhadap konten yang di-hosting pada infrastruktur ini. Anda juga bertanggung jawab atas tugas konfigurasi dan manajemen keamanan untuk Layanan AWS yang Anda gunakan. Untuk informasi selengkapnya tentang privasi data, lihat FAQ Privasi Data. Untuk informasi tentang perlindungan data di Eropa, silakan lihat postingan blog Model Tanggung Jawab Bersama AWS dan GDPR di Blog Keamanan AWS.

Untuk tujuan perlindungan data, sebaiknya Anda melindungi kredensial Akun AWS dan menyiapkan AWS IAM Identity Center atau AWS Identity and Access Management (IAM) untuk pengguna individu. Dengan cara seperti itu, 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:

- Gunakan autentikasi multi-faktor (MFA) pada setiap akun.
- Gunakan SSL/TLS untuk melakukan komunikasi dengan sumber daya AWS. Kami membutuhkan TLS 1.2 dan merekomendasikan TLS 1.3.
- Siapkan API dan log 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 sensitif yang disimpan di Amazon S3.
- Jika Anda memerlukan modul kriptografi tervalidasi FIPS 140-2 ketika mengakses AWS melalui antarmuka baris perintah atau API, gunakan titik akhir FIPS. Untuk informasi selengkapnya tentang titik akhir FIPS yang tersedia, silakan lihat Standar Pemrosesan Informasi Federal (FIPS) 140-2.

Sebaiknya Anda tidak memasukkan informasi rahasia atau sensitif, seperti alamat email pelanggan, ke dalam tanda atau bidang teks bebas seperti bidang Nama. Ini termasuk saat Anda bekerja dengan atau lainnya Layanan AWS menggunakan konsol, APIAWS CLI, atau AWS SDK. Data apa pun yang Anda masukkan ke dalam tanda atau bidang teks bebas yang digunakan untuk nama dapat digunakan untuk log penagihan atau diagnostik. Saat Anda memberikan URL ke server eksternal, sebaiknya Anda tidak menyertakan informasi kredensial di URL untuk memvalidasi permintaan Anda ke server tersebut.

Keamanan data di MemoryDB untuk Redis

Untuk membantu menjaga keamanan data Anda, MemoryDB for Redis dan Amazon EC2 menyediakan mekanisme untuk melindungi terhadap akses data Anda yang tidak sah di server.

MemoryDB juga menyediakan fitur enkripsi untuk data pada cluster:

- Enkripsi in-transit mengenkripsi data Anda setiap kali data bergerak dari satu tempat ke tempat lain, misalnya antara beberapa simpul di klaster Anda atau antara klaster dan aplikasi Anda.
- Enkripsi AT-rest mengenkripsi log transaksi dan data on-disk Anda selama operasi snapshot.

Anda juga dapat menggunakan Mengautentikasi pengguna dengan Daftar Kontrol Akses (ACL) untuk mengontrol akses pengguna ke cluster Anda.

Topik

Enkripsi At-Rest di MemoryDB

Untuk membantu menjaga keamanan data Anda, MemoryDB untuk Redis dan Amazon S3 menyediakan beberapa cara untuk membatasi akses ke data di dalam klaster Anda. Untuk informasi selengkapnya, lihat MemoryDB dan Amazon VPC dan Manajemen identitas dan akses di MemoryDB untuk Redis.

Enkripsi at-rest MemoryDB selalu diaktifkan untuk meningkatkan keamanan data dengan mengenkripsi data yang gigih. Ini mengenkripsi aspek-aspek berikut:

- Data dalam log transaksi
- Disk selama operasi sinkronisasi, snapshot, dan swap
- Snapshot yang disimpan di Amazon S3

MemoryDB menawarkan enkripsi default (dikelola layanan) saat istirahat, serta kemampuan untuk menggunakan kunci root pelanggan yang simetris Anda sendiri di AWSKey Management Service (KMS).

Data yang disimpan pada SSD (solid-state drive) dalam klaster yang diaktifkan dengan tiering data selalu dienkripsi secara default.

Untuk informasi tentang enkripsi in-transit, lihat Enkripsi dalam transit (TLS) di MemoryDB

Topik

- Menggunakan Customer Managed Keys dariAWS KMS
- Lihat juga

Menggunakan Customer Managed Keys dariAWS KMS

MemoryDB mendukung kunci root yang dikelola pelanggan yang simetris (kunci KMS) untuk enkripsi saat istirahat. Kunci KMS yang dikelola pelanggan adalah kunci enkripsi yang Anda buat, miliki, dan kelola di akun AWS Anda. Untuk informasi selengkapnya, lihat Kunci Root Pelanggan di Panduan Pengembang Layanan ManajemenAWS Kunci. Kunci harus dibuat diAWS KMS sebelum dapat digunakan dengan MemoryDB.

Untuk mempelajari cara membuat kunci root AWS KMS, lihat Buat Kunci di Panduan Developer AWS Key Management Service.

MemoryDB memungkinkan Anda untuk berintegrasi denganAWS KMS. Untuk informasi lain, lihat Menggunakan Grant di Panduan Developer AWS Key Management Service. Tidak diperlukan tindakan pelanggan untuk mengaktifkan integrasi MemoryDB denganAWS KMS.

Kuncikms:ViaService kondisi membatasi penggunaan kunciAWS KMS untuk permintaan dariAWS layanan tertentu. Untuk digunakankms:ViaService dengan MemoryDB, termasuk kedua ViaService nama dalam nilai kunci syarat:memorydb.amazon_region.amazonaws.com. Untuk informasi lebih lanjut, lihat KMS:ViaService.

Anda dapat menggunakan AWS CloudTrailuntuk melacak permintaan yang MemoryDB untuk Redis kirimkan keAWS Key Management Service atas nama Anda. Semua panggilan API yangAWS Key Management Service terkait dengan kunci yang dikelola pelanggan memiliki CloudTrail log yang sesuai. Anda juga dapat melihat grant yang dibuat oleh MemoryDB dengan memanggil API ListGrantsKMS.

Setelah klaster dienkripsi menggunakan kunci yang dikelola pelanggan, semua snapshot untuk klaster akan dienkripsi sebagai berikut:

- Snapshot harian otomatis dienkripsi menggunakan kunci yang dikelola pelanggan yang terkait dengan klaster.
- Snapshot akhir yang dibuat saat klaster dihapus, juga dienkripsi menggunakan kunci yang dikelola pelanggan yang terkait dengan klaster.
- Snapshot yang dibuat secara manual dienkripsi secara default untuk menggunakan kunci KMS yang terkait dengan klaster. Anda dapat mengesampingkan ini dengan memilih kunci dikelola pelanggan yang lain.
- Menyalin snapshot berakibat default pada menggunakan kunci yang dikelola pelanggan yang terkait dengan snapshot sumber. Anda dapat mengesampingkan ini dengan memilih kunci dikelola pelanggan yang lain.

> **Note**
>
> - Kunci yang dikelola pelanggan tidak dapat digunakan saat mengekspor snapshot ke bucket Amazon S3 pilihan Anda. Namun, semua snapshot yang diekspor ke Amazon S3 dienkripsi menggunakan enkripsi sisi Server. Anda dapat memilih untuk menyalin file snapshot 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 dikelola pelanggan untuk mengenkripsi snapshot yang dibuat secara manual yang tidak menggunakan kunci dikelola pelanggan untuk enkripsi. Dengan opsi ini, file snapshot yang disimpan di Amazon S3 akan dienkripsi menggunakan kunci KMS, meskipun data itu tidak dienkripsi pada klaster yang asli.

Memulihkan dari snapshot memungkinkan Anda memilih dari opsi enkripsi yang tersedia, mirip dengan pilihan enkripsi yang tersedia saat membuat klaster baru.

- Jika Anda menghapus kunci atau menonaktifkan kunci dan mencabut kunci yang digunakan dalam mengenkripsi klaster, maka klaster itu 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 yang dikelola pelanggan yang berbeda untuk membuat snapshot untuk tujuan pengarsipan.
- Rotasi kunci otomatis mempertahankan properti kunci rootAWS KMS Anda, sehingga rotasi tidak berpengaruh pada kemampuan Anda mengakses data MemoryDB Anda. Cluster MemoryDB terenkripsi tidak mendukung rotasi kunci manual, yang melibatkan pembuatan kunci root baru dan memperbarui referensi apa pun ke kunci lama. Untuk mempelajari lebih lanjut, lihat Memutar Kunci root pelanggan di Panduan DeveloperAWS Key Management Service.
- Mengenkripsi klaster MemoryDB menggunakan kunci KMS memerlukan satu grant per klaster. Grant ini digunakan sepanjang usia klaster. Selain itu, satu grant per snapshot digunakan selama pembuatan snapshot. Grant ini diistirahatkan setelah snapshot dibuat.
- Untuk informasi lain tentang grant dan batasAWS KMS, lihat Kuota di Panduan DeveloperAWS Key Management Service.

Lihat juga

- Enkripsi dalam transit (TLS) di MemoryDB
- MemoryDB dan Amazon VPC
- Manajemen identitas dan akses di MemoryDB untuk Redis

Enkripsi dalam transit (TLS) di MemoryDB

Untuk membantu menjaga keamanan data Anda, MemoryDB for Redis dan Amazon EC2 menyediakan mekanisme untuk melindungi terhadap akses data Anda yang tidak sah di server. Dengan menyediakan kemampuan enkripsi dalam perjalanan, MemoryDB memberi Anda alat yang dapat Anda gunakan untuk membantu melindungi data Anda saat berpindah dari satu lokasi ke lokasi lain. Misalnya, Anda dapat memindahkan data dari node utama ke node replika baca di dalam klaster, atau antara cluster dan aplikasi Anda.

Topik

Gambaran umum enkripsi bergerak

Enkripsi dalam transit MemoryDB for Redis adalah fitur yang meningkatkan keamanan data Anda pada titik yang paling rentan — ketika sedang dalam perjalanan dari satu lokasi ke lokasi lain.

Enkripsi dalam transit MemoryDB mengimplementasikan fitur-fitur berikut:

- Koneksi terenkripsi — baik koneksi server dan klien dienkripsi Transport Layer Security (TLS).
- Replikasi terenkripsi— data yang bergerak antara node primer dan node replika dienkripsi.
- Autentikasi server—Klien dapat mengautentikasi bahwa koneksinya dilakukan ke server yang benar.

Mulai 07/20/2023, TLS 1.2 adalah versi minimum yang didukung untuk cluster baru dan yang sudah ada. Gunakan tautan ini untuk mempelajari lebih lanjut tentang TLS 1.2 di AWS.

Untuk informasi lebih lanjut tentang menghubungkan ke cluster MemoryDB, lihat. Menghubungkan ke node MemoryDB menggunakan redis-cli

Lihat juga

- Enkripsi At-Rest di MemoryDB
- Mengautentikasi Pengguna dengan Daftar Kontrol Akses (ACL)
- MemoryDB dan Amazon VPC
- Manajemen identitas dan akses di MemoryDB untuk Redis

Mengautentikasi pengguna dengan Daftar Kontrol Akses (ACL)

Anda dapat mengautentikasi pengguna dengan Access Control List (ACL).

ACL memungkinkan Anda mengontrol akses klaster dengan mengelompokkan pengguna. Daftar kontrol Access ini dirancang sebagai cara untuk mengatur akses ke cluster.

Dengan ACL, Anda membuat pengguna dan menetapkan mereka izin tertentu dengan menggunakan string akses, seperti yang dijelaskan di bagian berikutnya. Anda menetapkan pengguna ke daftar kontrol Access selaras dengan peran tertentu (administrator, sumber daya manusia) yang kemudian disebarkan ke satu atau beberapa cluster MemoryDB. Dengan melakukan ini, Anda dapat menetapkan batas keamanan antara klien menggunakan cluster atau cluster MemoryDB yang sama dan mencegah klien mengakses data satu sama lain.

ACL dirancang untuk mendukung pengenalan Redis ACL di Redis 6. Saat Anda menggunakan ACL dengan cluster MemoryDB Anda, ada beberapa batasan:

- Anda tidak dapat menentukan kata sandi dalam string akses. Anda mengatur kata sandi dengan CreateUseratau UpdateUserpanggilan.
- Untuk hak pengguna, Anda meneruskan on dan `off` sebagai bagian dari string akses. Jika keduanya tidak ditentukan dalam string akses, pengguna ditugaskan `off` dan tidak memiliki hak akses ke cluster.
- Anda tidak dapat menggunakan perintah terlarang. Jika Anda menentukan perintah terlarang, pengecualian akan dilemparkan. Untuk daftar perintah tersebut, lihatPerintah Redis Terbatas.
- Anda tidak dapat menggunakan perintah `reset` sebagai bagian dari string akses. Anda menentukan kata sandi dengan parameter API, dan MemoryDB 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. MemoryDB mendukung `ACL LIST` perintah, tetapi tidak menyertakan dukungan untuk hash kata sandi seperti yang dilakukan Redis. Dengan MemoryDB, Anda dapat menggunakan DescribeUsersoperasi untuk mendapatkan informasi serupa, termasuk aturan yang terkandung dalam string akses. Namun, DescribeUserstidak mengambil kata sandi pengguna.

  Perintah read-only lainnya yang didukung oleh MemoryDB termasuk ACL WHOAMI, ACL USERS, dan ACL CAT. MemoryDB tidak mendukung perintah ACL berbasis tulis lainnya.

Menggunakan ACL dengan MemoryDB dijelaskan secara lebih rinci berikut.

Topik

Menentukan Izin Menggunakan String Akses

Untuk menentukan izin ke cluster MemoryDB, Anda membuat string akses dan menetapkannya ke pengguna, menggunakan salah satu atau. AWS CLI 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. Untuk dapat menjalankan perintah, pengguna harus memiliki akses ke perintah yang dijalankan dan semua kunci yang diakses oleh perintah itu. 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.
- &*— Akses diberikan ke semua saluran pubsub.
- +@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 Daftar Kontrol Akses dengan Konsol dan CLI.

Jika Anda memigrasikan beban kerja yang ada ke MemoryDB, Anda dapat mengambil string akses dengan meneleponACL LIST, tidak termasuk pengguna dan hash kata sandi apa pun.

Kemampuan pencarian vektor

> **Note**
>
> Fitur ini dalam rilis pratinjau untuk MemoryDB untuk Redis dan dapat berubah.

UntukPencarian vektor, semua perintah pencarian termasuk dalam `@search` kategori dan kategori yang ada `@read``@write`, `@fast` dan `@slow` diperbarui untuk menyertakan perintah pencarian. Jika pengguna tidak memiliki akses ke kategori, maka pengguna tidak memiliki akses ke perintah apa pun dalam kategori tersebut. Misalnya, jika pengguna tidak memiliki akses ke`@search`, maka pengguna tidak dapat menjalankan perintah terkait pencarian.

Tabel berikut menunjukkan pemetaan perintah pencarian ke kategori yang sesuai.

Perintah VSS	@read	@write	@fast	@slow
`FT.CREATE`		Y	Y	
`FT.DROPINDEX`		Y	Y	
`FT.LIST`	Y			Y

Perintah VSS	@read	@write	@fast	@slow
FT.INFO	Y		Y	
FT.SEARCH	Y			Y
FT.AGGREGATE	Y			Y
FT.PROFILE	Y			Y
FT.ALIASADD		Y	Y	
FT.ALIASDEL		Y	Y	
FT.ALIASUPDATE		Y	Y	
FT._ALIASLIST	Y			Y
FT.EXPLAIN	Y		Y	
FT.EXPLAINCLI	Y		Y	
FT.CONFIG	Y		Y	

Menerapkan ACL ke cluster untuk MemoryDB

Untuk menggunakan MemoryDB ACL, Anda mengambil langkah-langkah berikut:

1. Buat satu atau beberapa pengguna.
2. Buat ACL dan tambahkan pengguna ke daftar.
3. Tetapkan ACL ke cluster.

Langkah ini dijelaskan secara mendetail di bagian berikut.

Topik

- Membuat Pengguna dan Daftar Kontrol Akses dengan Konsol dan CLI
- Mengelola Daftar Kontrol Akses dengan Konsol dan CLI
- Menetapkan daftar kontrol Access ke cluster

Membuat Pengguna dan Daftar Kontrol Akses dengan Konsol dan CLI

Informasi pengguna untuk pengguna ACL adalah nama pengguna, dan secara opsional kata sandi dan string akses. String akses menyediakan tingkat izin pada kunci dan perintah. Nama ini unik untuk pengguna dan itulah yang diteruskan ke mesin.

Pastikan bahwa izin pengguna yang Anda berikan masuk akal dengan tujuan ACL yang dimaksudkan. Misalnya, jika Anda membuat ACL yang dipanggil`Administrators`, setiap pengguna yang Anda tambahkan ke grup itu harus memiliki string aksesnya disetel ke akses penuh ke kunci dan perintah. Untuk pengguna di `e-commerce` ACL, Anda dapat mengatur string akses mereka ke akses hanya-baca.

MemoryDB secara otomatis mengkonfigurasi pengguna default per akun dengan nama pengguna. `"default"` Ini tidak akan dikaitkan dengan cluster apa pun kecuali eksplisit ditambahkan ke ACL. 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.

ACL “akses terbuka” yang tidak dapat diubah akan dibuat untuk setiap akun yang berisi pengguna default. Ini adalah satu-satunya ACL pengguna default dapat menjadi anggota. Saat Anda membuat cluster, Anda harus memilih ACL untuk dikaitkan dengan cluster. Meskipun Anda memiliki opsi untuk menerapkan ACL “akses terbuka” dengan pengguna default, kami sangat menyarankan untuk membuat ACL dengan pengguna yang memiliki izin terbatas pada kebutuhan bisnis mereka.

Cluster yang tidak mengaktifkan TLS harus menggunakan ACL “akses terbuka” untuk memberikan otentikasi terbuka.

ACL dapat dibuat tanpa pengguna. ACL kosong tidak akan memiliki akses ke cluster dan hanya dapat dikaitkan dengan cluster yang mendukung TLS.

Saat membuat pengguna, Anda dapat menetapkan hingga dua kata sandi. Saat Anda memodifikasi kata sandi, koneksi apa pun yang ada ke cluster dipertahankan.

Secara khusus, perhatikan kendala kata sandi pengguna ini saat menggunakan ACL untuk MemoryDB:

- Kata sandi harus berupa 16–128 karakter yang dapat dicetak.
- Karakter nonalfanumerik berikut tidak diperbolehkan: `, "" / @`.

Mengelola Pengguna dengan Konsol dan CLI

Membuat pengguna (Konsol)

Untuk membuat pengguna di konsol

1. Masuk ke AWS Management Console dan buka konsol MemoryDB for Redis di https://console.aws.amazon.com/memorydb/.
2. Di panel navigasi kiri, pilih Pengguna.
3. Pilih Buat pengguna
4. Pada halaman Buat pengguna, masukkan Nama.

   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.
5. Di bawah Kata Sandi, Anda dapat memasukkan hingga dua kata sandi.
6. Di bawah Access string, masukkan string akses. String akses menetapkan tingkat izin untuk kunci dan perintah apa saja yang diperbolehkan kepada pengguna.
7. Untuk Tag, Anda dapat menerapkan tag secara opsional untuk mencari dan memfilter pengguna Anda atau melacak AWS biaya Anda.
8. Pilih Buat.

Membuat pengguna menggunakan AWS CLI

Untuk membuat pengguna dengan menggunakan CLI

- Gunakan perintah create-user untuk membuat pengguna.

  Untuk Linux, macOS, atau Unix:

```
aws memorydb create-user \
  --user-name user-name-1 \
  --access-string "~objects:* ~items:* ~public:*" \
  --authentication-mode \
        Passwords="abc",Type=password
```

  Untuk Windows:

```
aws memorydb create-user ^
  --user-name user-name-1 ^
  --access-string "~objects:* ~items:* ~public:*" ^
  --authentication-mode \
        Passwords="abc",Type=password
```

Memodifikasi pengguna (Konsol)

Untuk memodifikasi pengguna di konsol

1. Masuk ke AWS Management Console dan buka konsol MemoryDB for Redis di https://console.aws.amazon.com/memorydb/.
2. Di panel navigasi kiri, pilih Pengguna.
3. Pilih tombol radio di sebelah pengguna yang ingin Anda modifikasi dan kemudian pilih Tindakan -> Ubah
4. Jika Anda ingin mengubah kata sandi, pilih tombol radio Ubah kata sandi. Perhatikan bahwa jika Anda memiliki dua kata sandi, Anda harus memasukkan keduanya saat memodifikasi salah satunya.
5. Jika Anda memperbarui string akses, masukkan yang baru.
6. Pilih Ubah.

Memodifikasi pengguna menggunakan AWS CLI

Untuk mengubah pengguna menggunakan CLI

1. Gunakan perintah update-user untuk memodifikasi pengguna.
2. Ketika pengguna dimodifikasi, daftar kontrol Access yang terkait dengan pengguna diperbarui, bersama dengan cluster apa pun yang terkait dengan ACL. Semua koneksi yang ada akan dipertahankan. Berikut ini adalah beberapa contohnya.

   Untuk Linux, macOS, atau Unix:

   ```
   aws memorydb update-user \
     --user-name user-name-1 \
     --access-string "~objects:* ~items:* ~public:*"
   ```

   Untuk Windows:

   ```
   aws memorydb update-user ^
     --user-name user-name-1 ^
     --access-string "~objects:* ~items:* ~public:*"
   ```

Melihat detail pengguna (Konsol)

Untuk melihat detail pengguna di konsol

1. Masuk ke AWS Management Console dan buka konsol MemoryDB for Redis di https://console.aws.amazon.com/memorydb/.
2. Di panel navigasi kiri, pilih Pengguna.
3. Pilih pengguna di bawah Nama pengguna atau gunakan kotak pencarian untuk menemukan pengguna.
4. Di bawah Pengaturan pengguna, Anda dapat meninjau string akses pengguna, jumlah kata sandi, status, dan Nama Sumber Daya Amazon (ARN).
5. Di bawah Daftar kontrol akses (ACL) Anda dapat meninjau ACL milik pengguna.
6. Di bawah Tag Anda dapat meninjau tag apa pun yang terkait dengan pengguna.

Melihat detail pengguna menggunakan AWS CLI

Gunakan perintah deskripsikan pengguna untuk melihat detail pengguna.

```
aws memorydb describe-users \
  --user-name my-user-name
```

Menghapus pengguna (Konsol)

Untuk menghapus pengguna di konsol

1. Masuk ke AWS Management Console dan buka konsol MemoryDB for Redis di https://console.aws.amazon.com/memorydb/.
2. Di panel navigasi kiri, pilih Pengguna.
3. Pilih tombol radio di sebelah pengguna yang ingin Anda modifikasi dan kemudian pilih Tindakan -> Hapus
4. Untuk mengonfirmasi, masukkan `delete` di kotak teks konfirmasi dan kemudian pilih Hapus.
5. Untuk membatalkan, pilih Batalkan.

Menghapus pengguna menggunakan AWS CLI

Untuk menghapus pengguna dengan menggunakan CLI

- Gunakan perintah delete-user untuk menghapus pengguna.

  Akun dihapus dan dihapus dari daftar kontrol Access yang menjadi miliknya. Berikut adalah contohnya.

  Untuk Linux, macOS, atau Unix:

  ```
  aws memorydb delete-user \
    --user-name user-name-2
  ```

  Untuk Windows:

  ```
  aws memorydb delete-user ^
    --user-name user-name-2
  ```

Mengelola Daftar Kontrol Akses dengan Konsol dan CLI

Anda dapat membuat daftar kontrol Access untuk mengatur dan mengontrol akses pengguna ke satu atau beberapa cluster, seperti yang ditunjukkan berikut.

Gunakan prosedur berikut untuk mengelola daftar kontrol Access menggunakan konsol.

Membuat Daftar Kontrol Akses (ACL) (Konsol)

Untuk membuat daftar kontrol Access menggunakan konsol

1. Masuk ke AWS Management Console dan buka konsol MemoryDB for Redis di https://console.aws.amazon.com/memorydb/.
2. Di panel navigasi kiri, pilih Daftar kontrol akses (ACL).
3. Pilih Buat ACL.
4. Pada halaman Buat daftar kontrol akses (ACL), masukkan nama ACL.

   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.
5. Di bawah Pengguna yang dipilih melakukan salah satu hal berikut:

   a. Buat pengguna baru dengan memilih Buat pengguna

   b. Tambahkan pengguna dengan memilih Kelola lalu pilih pengguna dari dialog Kelola pengguna lalu pilih Pilih.
6. Untuk Tag, Anda dapat menerapkan tag secara opsional untuk mencari dan memfilter ACL Anda atau melacak biaya Anda AWS .
7. Pilih Buat.

Membuat Access Control List (ACL) menggunakan AWS CLI

Gunakan prosedur berikut untuk membuat daftar kontrol Akses menggunakan CLI.

Untuk membuat ACL baru dan menambahkan pengguna dengan menggunakan CLI

- Gunakan perintah create-acl untuk membuat ACL.

  Untuk Linux, macOS, atau Unix:

```
aws memorydb create-acl \
  --acl-name "new-acl-1" \
  --user-names "user-name-1" "user-name-2"
```

  Untuk Windows:

```
aws memorydb create-acl ^
  --acl-name "new-acl-1" ^
  --user-names "user-name-1" "user-name-2"
```

Memodifikasi Daftar Kontrol Akses (ACL) (konsol)

Untuk mengubah daftar kontrol Access menggunakan konsol

1. Masuk ke AWS Management Console dan buka konsol MemoryDB for Redis di https://console.aws.amazon.com/memorydb/.
2. Di panel navigasi kiri, pilih Daftar kontrol akses (ACL).
3. Pilih ACL yang ingin Anda ubah dan kemudian pilih Modify
4. Pada halaman Ubah, di bawah Pengguna yang dipilih lakukan salah satu hal berikut:
   a. Buat pengguna baru dengan memilih Buat pengguna untuk ditambahkan ke ACL.
   b. Menambah atau menghapus pengguna dengan memilih Kelola lalu memilih atau menghapus pilihan pengguna dari dialog Kelola pengguna lalu memilih Pilih.
5. Pada halaman Buat daftar kontrol akses (ACL), masukkan nama ACL.

   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.

6. Di bawah Pengguna yang dipilih melakukan salah satu hal berikut:

   a. Buat pengguna baru dengan memilih Buat pengguna

   b. Tambahkan pengguna dengan memilih Kelola lalu pilih pengguna dari dialog Kelola pengguna lalu pilih Pilih.

7. Pilih Ubah untuk menyimpan perubahan Anda atau Batalkan untuk membuangnya.

Memodifikasi Access Control List (ACL) menggunakan AWS CLI

Untuk memodifikasi ACL dengan menambahkan pengguna baru atau menghapus anggota saat ini dengan menggunakan CLI

- Gunakan perintah update-acl untuk memodfikasi ACL.

  Untuk Linux, macOS, atau Unix:

```
aws memorydb update-acl --acl-name new-acl-1 \
--user-names-to-add user-name-3 \
--user-names-to-remove user-name-2
```

  Untuk Windows:

```
aws memorydb update-acl --acl-name new-acl-1 ^
--user-names-to-add user-name-3 ^
--user-names-to-remove user-name-2
```

> **Note**
>
> Setiap koneksi terbuka milik pengguna yang dihapus dari ACL diakhiri dengan perintah ini.

Melihat detail Daftar Kontrol Akses (ACL) (Konsol)

Untuk melihat detail ACL di konsol

1. Masuk ke AWS Management Console dan buka konsol MemoryDB for Redis di https://console.aws.amazon.com/memorydb/.
2. Di panel navigasi kiri, pilih Daftar kontrol akses (ACL).

3. Pilih ACL di bawah nama ACL atau gunakan kotak pencarian untuk menemukan ACL.
4. Di bawah Pengguna, Anda dapat meninjau daftar pengguna yang terkait dengan ACL.
5. Di bawah Kluster terkait, Anda dapat meninjau klaster tempat ACL berada.
6. Di bawah Tag Anda dapat meninjau tag apa pun yang terkait dengan ACL.

Melihat Daftar Kontrol Akses (ACL) menggunakan AWS CLI

Gunakan perintah describe-acls untuk melihat detail ACL.

```
aws memorydb describe-acls \
  --acl-name test-group
```

Menghapus Daftar Kontrol Akses (ACL) (konsol)

Untuk menghapus daftar kontrol Access menggunakan konsol

1. Masuk ke AWS Management Console dan buka konsol MemoryDB for Redis di https://console.aws.amazon.com/memorydb/.
2. Di panel navigasi kiri, pilih Daftar kontrol akses (ACL).
3. Pilih ACL yang ingin Anda ubah dan kemudian pilih Hapus
4. Pada halaman Hapus, masukkan `delete` di kotak konfirmasi dan pilih Hapus atau Batal untuk menghindari penghapusan ACL.

ACL itu sendiri, bukan pengguna yang termasuk dalam grup, dihapus.

Menghapus Access Control List (ACL) menggunakan AWS CLI

Untuk menghapus ACL dengan menggunakan CLI

- Gunakan perintah delete-acl untuk menghapus ACL.

  Untuk Linux, macOS, atau Unix:

  ```
  aws memorydb delete-acl /
     --acl-name
  ```

  Untuk Windows:

```
aws memorydb delete-acl ^
   --acl-name
```

Contoh sebelumnya menghasilkan respons berikut.

```
aws memorydb delete-acl --acl-name "new-acl-1"
{
    "ACLName": "new-acl-1",
    "Status": "deleting",
    "EngineVersion": "6.2",
    "UserNames": [
        "user-name-1",
        "user-name-3"
    ],
    "clusters": [],
    "ARN":"arn:aws:memorydb:us-east-1:493071037918:acl/new-acl-1"
}
```

Menetapkan daftar kontrol Access ke cluster

Setelah Anda membuat ACL dan menambahkan pengguna, langkah terakhir dalam mengimplementasikan ACL adalah menetapkan ACL ke cluster.

Menetapkan daftar kontrol Access ke cluster Menggunakan Konsol

Untuk menambahkan ACL ke cluster menggunakan AWS Management Console, lihatMembuat cluster MemoryDB.

Menetapkan daftar kontrol Access ke cluster Menggunakan AWS CLI

AWS CLI Operasi berikut membuat cluster dengan enkripsi dalam transit (TLS) diaktifkan dan acl-name parameter dengan nilai*my-acl-name*. Ganti grup subnet `subnet-group` dengan grup subnet yang ada.

Parameter Kunci

- **`--engine-version`**Harus 6.2.
- **`--tls-enabled`**— Digunakan untuk otentikasi dan untuk mengaitkan ACL.
- **`--acl-name`**— Nilai ini menyediakan daftar kontrol Access yang terdiri dari pengguna dengan izin akses tertentu untuk cluster.

Untuk Linux, macOS, atau Unix:

```
aws memorydb create-cluster \
    --cluster-name "new-cluster" \
    --description "new-cluster" \
    --engine-version "6.2" \
    --node-type db.r6g.large \
    --tls-enabled \
    --acl-name "new-acl-1" \
    --subnet-group-name "subnet-group"
```

Untuk Windows:

```
aws memorydb create-cluster ^
    --cluster-name "new-cluster" ^
    --cluster-description "new-cluster" ^
    --engine-version "6.2" ^
    --node-type db.r6g.large ^
    --tls-enabled ^
    --acl-name "new-acl-1" ^
    --subnet-group-name "subnet-group"
```

AWS CLI Operasi berikut memodifikasi cluster dengan enkripsi dalam transit (TLS) diaktifkan dan acl-name parameter dengan nilai. `new-acl-2`

Untuk Linux, macOS, atau Unix:

```
aws memorydb update-cluster \
    --cluster-name cluster-1 \
    --acl-name "new-acl-2"
```

Untuk Windows:

```
aws memorydb update-cluster ^
    --cluster-name cluster-1 ^
    --acl-name "new-acl-2"
```

Mengtentikasi IAM

Topik

Gambaran Umum

Dengan IAM Authentication Anda dapat mengotentikasi koneksi ke MemoryDB menggunakan identitas AWS IAM, ketika cluster Anda dikonfigurasi untuk menggunakan Redis versi 7 atau di atas. Ini memungkinkan Anda untuk memperkuat model keamanan Anda dan menyederhanakan banyak tugas keamanan administratif. Dengan IAM Authentication Anda dapat mengkonfigurasi kontrol akses berbutir halus untuk setiap klaster MemoryDB individu dan pengguna MemoryDB dan mengikuti prinsip-prinsip izin hak istimewa. Otentikasi IAM untuk MemoryDB Redis bekerja dengan menyediakan token otentikasi IAM berumur pendek bukan kata sandi pengguna MemoryDB berumur panjang di Redis atau perintah. `AUTH HELL0` Untuk informasi lebih lanjut tentang token otentikasi 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 mereka untuk membatasi akses Redis lebih lanjut. Anda juga dapat memberikan akses ke pengguna dari penyedia Identitas federasi mereka langsung ke cluster MemoryDB.

Untuk menggunakan AWS IAM dengan MemoryDB, Anda harus terlebih dahulu membuat pengguna MemoryDB dengan mode otentikasi diatur ke IAM, maka Anda dapat membuat atau menggunakan kembali identitas IAM. Identitas IAM membutuhkan kebijakan terkait untuk memberikan `memorydb:Connect` tindakan ke cluster MemoryDB dan pengguna MemoryDB. Setelah dikonfigurasi, Anda dapat membuat token otentikasi IAM menggunakan AWS kredensi pengguna atau peran IAM. Akhirnya Anda perlu memberikan token otentikasi IAM berumur pendek sebagai kata sandi di klien Redis Anda saat menghubungkan ke simpul cluster MemoryDB Anda. Klien Redis dengan dukungan untuk penyedia kredenial dapat secara otomatis menghasilkan kredensil sementara secara otomatis untuk setiap koneksi baru. MemoryDB akan melakukan otentikasi IAM untuk permintaan koneksi pengguna MemoryDB IAM-enabled dan akan memvalidasi permintaan koneksi dengan IAM.

Keterbatasan:

Saat menggunakan IAM, batasan berikut berlaku:

- IAM tersedia saat menggunakan mesin Redis versi 7.0 atau lebih tinggi.
- Token IAM berlaku selama 15 menit. Untuk koneksi berumur panjang, sebaiknya gunakan klien Redis yang mendukung antarmuka penyedia kredenial.
- Koneksi IAM yang diautentikasi ke MemoryDB akan secara otomatis terputus setelah 12 jam. Koneksi dapat diperpanjang selama 12 jam dengan mengirim `AUTH` atau `HELLO` perintah dengan token otentikasi IAM baru.
- Otentikasi IAM tidak didukung dalam `MULTI EXEC` perintah.
- Saat ini, IAM tidak mendukung kunci konteks ketentuan global. Untuk informasi lebih lanjut tentang kunci konteks syarat AWS global dalam Panduan Pengguna IAM.

Pengaturan

Untuk mengatur otentikasi IAM:

1. Membuat klaster

```
aws memorydb create-cluster \
    --cluster-name cluster-01 \
    --description "MemoryDB IAM auth application"
    --node-type db.r6g.large \
    --engine-version 7.0 \
    --acl-name open-access
```

2. Buat dokumen kebijakan kepercayaan IAM, seperti yang ditunjukkan di bawah ini, untuk peran Anda yang memungkinkan akun Anda untuk 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" : [
        "memorydb:connect"
      ],
      "Resource" : [
        "arn:aws:memorydb:us-east-1:123456789012:cluster/cluster-01",
        "arn:aws:memorydb:us-east-1:123456789012:user/iam-user-01"
      ]
    }
  ]
}
```

4. Buatlah IAM role.

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

5. Buat kebijakan IAM.

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

6. Lampirkan kebijakan IAM pada peran tersebut.

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

7. Buat pengguna baru.

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

8. Buat ACL dan lampirkan pengguna.

```
aws memorydb create-acl \
  --acl-name iam-acl-01 \
  --user-names iam-user-01

aws memorydb update-cluster \
  --cluster-name cluster-01 \
  --acl-name iam-acl-01
```

Terhubung

Connect dengan token sebagai kata sandi

Anda harus terlebih dahulu membuat token otentikasi IAM berumur pendek menggunakan permintaan AWSSigv4 yang telah ditandatangani sebelumnya. Setelah itu Anda memberikan token otentikasi IAM sebagai kata sandi saat menghubungkan ke cluster MemoryDB, seperti yang ditunjukkan pada contoh di bawah ini.

```
String userName = "insert user name"
String clusterName = "insert cluster name"
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 MemoryDB
 Redis.
IAMAuthTokenRequest iamAuthTokenRequest = new IAMAuthTokenRequest(userName,
 clusterName, region);
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(userName, iamAuthToken)
```

```
    .build();

// Create a new Lettuce Redis client
RedisClusterClient client = RedisClusterClient.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 ACTION_NAME = "connect";
    private static final String SERVICE_NAME = "memorydb";
    private static final long TOKEN_EXPIRY_SECONDS = 900;

    private final String userName;
    private final String clusterName;
    private final String region;

    public IAMAuthTokenRequest(String userName, String clusterName, String region) {
        this.userName = userName;
        this.clusterName = clusterName;
        this.region = region;
    }

    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(userName));
        return request;
    }

    private URI getRequestUri() {
        return URI.create(String.format("%s%s/", REQUEST_PROTOCOL, clusterName));
    }

    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());
    }
}
```

Connect dengan penyedia kredensi

Kode di bawah ini menunjukkan cara mengotentikasi dengan MemoryDB menggunakan penyedia kredensi otentikasi IAM.

```
String userName = "insert user name"
String clusterName = "insert cluster name"
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 MemoryDB Redis.
IAMAuthTokenRequest iamAuthTokenRequest = new IAMAuthTokenRequest(userName,
 clusterName, region);

// Create a Redis credentials provider using IAM credentials.
RedisCredentialsProvider redisCredentialsProvider = new
 RedisIAMAuthCredentialsProvider(
    userName, 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 cluster client
RedisClusterClient client = RedisClusterClient.create(redisURI);
client.connect();
```

Di bawah ini adalah contoh klien cluster Lettuce Redis yang membungkus IAM AuthTokenRequest di penyedia kredensil untuk menghasilkan kredensil sementara secara otomatis bila diperlukan.

```
public class RedisIAMAuthCredentialsProvider implements RedisCredentialsProvider {
    private static final long TOKEN_EXPIRY_SECONDS = 900;

    private final AWSCredentialsProvider awsCredentialsProvider;
    private final String userName;
    private final IAMAuthTokenRequest iamAuthTokenRequest;
    private final Supplier<String> iamAuthTokenSupplier;

    public RedisIAMAuthCredentialsProvider(String userName,
        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(userName, iamAuthTokenSupplier.get()));
    }

    private String getIamAuthToken() {
        return
iamAuthTokenRequest.toSignedRequestUri(awsCredentialsProvider.getCredentials());
    }
```

Manajemen identitas dan akses di MemoryDB untuk Redis

AWS Identity and Access Management (IAM) adalah Layanan AWS yang membantu administrator mengontrol akses ke AWS sumber daya dengan aman. Administrator IAM mengontrol siapa yang dapat diautentikasi (masuk) dan diberi wewenang (memiliki izin) untuk menggunakan sumber daya MemoryDB. IAM adalah Layanan AWS yang dapat Anda gunakan tanpa biaya tambahan.

Topik

Audiens

Bagaimana Anda menggunakan AWS Identity and Access Management (IAM) berbeda, tergantung pada pekerjaan yang Anda lakukan di MemoryDB.

Pengguna layanan - Jika Anda menggunakan layanan MemoryDB untuk melakukan pekerjaan Anda, maka administrator Anda memberi Anda kredensi dan izin yang Anda butuhkan. Saat Anda menggunakan lebih banyak fitur MemoryDB untuk melakukan pekerjaan Anda, Anda mungkin memerlukan izin tambahan. Memahami cara akses dikelola dapat membantu Anda meminta izin yang

tepat dari administrator Anda. Jika Anda tidak dapat mengakses fitur di MemoryDB, lihat. Pemecahan masalah MemoryDB untuk identitas dan akses Redis

Administrator layanan - Jika Anda bertanggung jawab atas sumber daya MemoryDB di perusahaan Anda, Anda mungkin memiliki akses penuh ke MemoryDB. Tugas Anda adalah menentukan fitur dan sumber daya MemoryDB mana yang harus diakses pengguna layanan Anda. Kemudian, Anda harus mengirimkan permintaan kepada administrator IAM Anda untuk mengubah izin pengguna layanan Anda. Tinjau informasi di halaman ini untuk memahami konsep Basic IAM. Untuk mempelajari lebih lanjut tentang bagaimana perusahaan Anda dapat menggunakan IAM dengan MemoryDB, lihat. Bagaimana MemoryDB untuk Redis bekerja dengan IAM

Administrator IAM - Jika Anda seorang administrator IAM, Anda mungkin ingin mempelajari detail tentang bagaimana Anda dapat menulis kebijakan untuk mengelola akses ke MemoryDB. Untuk melihat contoh kebijakan berbasis identitas MemoryDB yang dapat Anda gunakan di IAM, lihat. Contoh kebijakan berbasis identitas untuk MemoryDB untuk Redis

Mengautentikasi dengan identitas

Otentikasi adalah cara Anda masuk AWS menggunakan kredensyal identitas Anda. Anda harus diautentikasi (masuk ke AWS) sebagai Pengguna root akun AWS, sebagai pengguna IAM, atau dengan mengasumsikan peran IAM.

Anda dapat masuk AWS sebagai identitas federasi dengan menggunakan kredensyal yang disediakan melalui sumber identitas. AWS IAM Identity Center Pengguna (IAM Identity Center), autentikasi masuk tunggal perusahaan Anda, dan kredensi Google atau Facebook Anda adalah contoh identitas federasi. Saat Anda masuk sebagai identitas gabungan, administrator Anda sebelumnya menyiapkan federasi identitas menggunakan peran IAM. Ketika Anda mengakses AWS dengan menggunakan federasi, Anda secara tidak langsung mengambil peran.

Bergantung pada jenis pengguna Anda, Anda dapat masuk ke AWS Management Console atau portal AWS akses. Untuk informasi selengkapnya tentang masuk AWS, lihat Cara masuk ke Panduan AWS Sign-In Pengguna Anda Akun AWS.

Jika Anda mengakses AWS secara terprogram, AWS sediakan kit pengembangan perangkat lunak (SDK) dan antarmuka baris perintah (CLI) untuk menandatangani permintaan Anda secara kriptografis dengan menggunakan kredensil Anda. Jika Anda tidak menggunakan AWS alat, Anda harus menandatangani permintaan sendiri. Untuk informasi selengkapnya tentang penggunaan metode yang disarankan untuk menandatangani permintaan sendiri, lihat Menandatangani permintaan AWS API di Panduan Pengguna IAM.

Apa pun metode autentikasi yang digunakan, Anda mungkin diminta untuk menyediakan informasi keamanan tambahan. Misalnya, AWS merekomendasikan agar Anda menggunakan otentikasi 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) di AWS dalam Panduan Pengguna IAM.

Akun AWS pengguna root

Saat Anda membuat Akun AWS, Anda mulai dengan satu identitas masuk yang memiliki akses lengkap ke semua Layanan AWS dan sumber daya di akun. Identitas ini disebut pengguna Akun AWS root dan diakses dengan masuk dengan alamat email dan kata sandi yang Anda gunakan untuk membuat akun. Kami sangat menyarankan agar Anda tidak menggunakan pengguna root untuk tugas sehari-hari Anda. 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, mewajibkan pengguna manusia, termasuk pengguna yang memerlukan akses administrator, untuk menggunakan federasi dengan penyedia identitas untuk mengakses Layanan AWS dengan menggunakan kredensi sementara.

Identitas federasi adalah pengguna dari direktori pengguna perusahaan Anda, penyedia identitas web, direktori Pusat Identitas AWS Directory Service, atau pengguna mana pun yang mengakses Layanan AWS dengan menggunakan kredensil yang disediakan melalui sumber identitas. Ketika identitas federasi mengakses Akun AWS, mereka mengambil peran, dan peran memberikan kredensyal sementara.

Untuk manajemen 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 aplikasi Akun AWS dan aplikasi Anda. Untuk informasi tentang Pusat Identitas IAM, lihat Apa itu Pusat Identitas IAM? dalam Panduan Pengguna AWS IAM Identity Center .

Pengguna dan grup IAM

Pengguna IAM adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus untuk satu orang atau aplikasi. Jika memungkinkan, sebaiknya andalkan kredensial sementara daripada membuat pengguna IAM yang memiliki kredensial jangka panjang seperti kata sandi dan kunci

akses. Namun, jika Anda memiliki kasus penggunaan khusus yang memerlukan kredensial jangka panjang dengan pengguna IAM, kami sarankan Anda merotasi kunci akses. Untuk informasi selengkapnya, lihat Rotasikan kunci akses secara rutin untuk kasus penggunaan yang memerlukan kredensial jangka panjang dalam Panduan Pengguna IAM.

Grup IAM adalah identitas yang menentukan kumpulan pengguna IAM. Anda tidak dapat masuk sebagai grup. Anda dapat menggunakan grup untuk menentukan izin untuk beberapa pengguna sekaligus. Grup membuat izin lebih mudah dikelola untuk sekelompok besar pengguna. Misalnya, Anda dapat memiliki grup yang bernama IAMAdmins dan memberikan izin kepada 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 adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus. Peran ini mirip dengan pengguna IAM, tetapi tidak terkait dengan orang tertentu. Anda dapat mengambil peran IAM untuk sementara AWS Management Console dengan beralih peran. Anda dapat mengambil peran dengan memanggil operasi AWS CLI atau AWS API atau dengan 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 gabungan – Untuk menetapkan izin ke sebuah identitas gabungan, Anda dapat membuat peran dan menentukan izin untuk peran tersebut. Saat identitas terfederasi mengautentikasi, identitas tersebut akan dikaitkan dengan peran dan diberi izin yang ditentukan oleh peran tersebut. Untuk informasi tentang peran-peran untuk federasi, lihat Membuat peran untuk Penyedia Identitas pihak ketiga dalam Panduan Pengguna IAM. Jika Anda menggunakan Pusat Identitas IAM, Anda perlu mengonfigurasi set izin. Untuk mengontrol apa yang dapat diakses identitas Anda setelah identitas tersebut diautentikasi, Pusat Identitas IAM mengorelasikan izin yang diatur ke peran dalam IAM. Untuk informasi tentang rangkaian izin, lihat Rangkaian 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 (pengguna utama tepercaya) dengan akun berbeda untuk mengakses sumber daya yang ada di akun Anda. Peran adalah cara utama untuk memberikan akses lintas akun. Namun, dengan beberapa Layanan AWS, Anda dapat melampirkan kebijakan langsung ke sumber daya (alih-alih menggunakan peran sebagai proxy). Untuk mempelajari perbedaan antara kebijakan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Bagaimana peran IAM berbeda dari kebijakan berbasis sumber daya dalam Panduan Pengguna IAM.
- Akses lintas layanan — Beberapa Layanan AWS menggunakan fitur lain Layanan AWS. Sebagai contoh, ketika Anda melakukan panggilan dalam 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 teruskan (FAS) — Saat Anda menggunakan pengguna atau peran IAM untuk melakukan tindakan AWS, Anda dianggap sebagai prinsipal. Saat Anda menggunakan beberapa layanan, Anda mungkin melakukan tindakan yang kemudian memulai tindakan lain di layanan yang berbeda. FAS menggunakan izin dari pemanggilan utama Layanan AWS, dikombinasikan dengan permintaan Layanan AWS untuk membuat permintaan ke layanan hilir. Permintaan FAS hanya dibuat ketika layanan menerima permintaan yang memerlukan interaksi dengan orang lain Layanan AWS atau sumber daya untuk menyelesaikannya. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan saat membuat permintaan FAS, lihat Teruskan sesi akses.
  - Peran layanan – Peran layanan adalah peran IAM yang diambil oleh layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, memodifikasi, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat Membuat peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan pengguna IAM.
  - Peran terkait layanan — Peran terkait layanan adalah jenis peran layanan yang ditautkan ke. Layanan AWS Layanan dapat menggunakan peran untuk melakukan tindakan atas nama Anda. Peran terkait layanan muncul di Anda Akun AWS dan dimiliki oleh layanan. 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 kredensi sementara untuk aplikasi yang berjalan pada instans EC2 dan membuat atau permintaan API. AWS CLI AWS Cara ini lebih dianjurkan daripada menyimpan kunci akses dalam instans EC2. Untuk menetapkan AWS peran ke instans EC2 dan membuatnya tersedia untuk semua aplikasinya, Anda membuat profil instance yang dilampirkan ke instance. 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 di 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 akses menggunakan kebijakan

Anda mengontrol akses AWS dengan membuat kebijakan dan melampirkannya ke AWS identitas atau sumber daya. Kebijakan adalah objek AWS yang, ketika dikaitkan dengan identitas atau sumber daya, menentukan izinnya. AWS mengevaluasi kebijakan ini ketika prinsipal (pengguna, pengguna root, atau sesi peran) membuat permintaan. Izin dalam kebijakan dapat menentukan permintaan yang diizinkan atau ditolak. Sebagian besar kebijakan disimpan AWS sebagai dokumen JSON. Untuk informasi selengkapnya tentang struktur dan konten dokumen kebijakan JSON, lihat Gambaran umum kebijakan JSON dalam Panduan Pengguna IAM.

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Artinya, pengguna utama manakah yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Secara default, pengguna dan peran tidak memiliki izin. Untuk memberikan izin kepada pengguna untuk melakukan tindakan pada 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 operasi. Sebagai contoh, anggap saja Anda memiliki kebijakan yang mengizinkan tindakan `iam:GetRole`. Pengguna dengan kebijakan tersebut bisa mendapatkan informasi peran dari AWS Management Console, API AWS CLI, atau AWS API.

Kebijakan berbasis identitas

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke sebuah identitas, seperti pengguna IAM, grup pengguna IAM, atau peran IAM. Kebijakan ini mengontrol jenis tindakan yang dapat dilakukan pengguna dan peran, di sumber daya mana, dan dengan ketentuan 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 dilampirkan ke beberapa pengguna, grup, dan peran dalam. Akun AWS Kebijakan AWS terkelola mencakup kebijakan terkelola 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 berbasis sumber daya adalah kebijakan kepercayaan 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 yang dilampiri kebijakan tersebut, kebijakan ini menentukan jenis tindakan yang dapat dilakukan oleh pengguna utama tertentu di sumber daya tersebut dan apa ketentuannya. Anda harus menentukan pengguna utama dalam kebijakan berbasis sumber daya. Prinsipal dapat mencakup akun, pengguna, peran, pengguna federasi, atau. Layanan AWS

Kebijakan berbasis sumber daya merupakan kebijakan inline yang terletak di layanan tersebut. Anda tidak dapat menggunakan kebijakan AWS terkelola dari IAM dalam kebijakan berbasis sumber daya.

Daftar kontrol akses (ACL)

Daftar kontrol akses (ACL) mengendalikan pengguna utama mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACL sama dengan kebijakan berbasis sumber daya, meskipun tidak menggunakan format dokumen kebijakan JSON.

Amazon S3, AWS WAF, dan Amazon VPC adalah contoh layanan yang mendukung ACL. Untuk mempelajari ACL selengkapnya, silakan lihat Gambaran umum daftar kontrol akses (ACL) di Panduan Developer Layanan Penyimpanan Ringkas Amazon.

Tipe kebijakan lain

AWS mendukung jenis kebijakan tambahan yang kurang umum. Tipe-tipe kebijakan ini dapat mengatur izin maksimum yang diberikan kepada Anda berdasarkan tipe kebijakan yang lebih umum.

- Batasan izin – Batasan izin adalah fitur lanjutan di mana Anda menetapkan izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas kepada entitas IAM (pengguna atau peran IAM). Anda dapat menetapkan batasan izin untuk suatu entitas. Izin yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas milik entitas dan batasan izinnya. Kebijakan berbasis sumber

daya yang menentukan pengguna atau peran dalam bidang `Principal` tidak dibatasi oleh batasan izin. Penolakan secara eksplisit terhadap salah satu kebijakan ini akan mengesampingkan izin tersebut. 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 organisasi atau unit organisasi (OU) di. AWS Organizations AWS Organizations adalah layanan untuk mengelompokkan dan mengelola secara terpusat beberapa Akun AWS yang dimiliki bisnis Anda. Jika Anda mengaktifkan semua fitur di sebuah organisasi, maka Anda dapat menerapkan kebijakan kontrol layanan (SCP) ke salah satu atau semua akun Anda. SCP membatasi izin untuk entitas di akun anggota, termasuk masing-masing. 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 teruskan sebagai parameter saat Anda membuat sesi sementara secara terprogram untuk peran atau pengguna gabungan. Izin sesi yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas pengguna atau peran dan kebijakan sesi. Izin juga bisa datang dari kebijakan berbasis sumber daya. Penolakan eksplisit dalam salah satu kebijakan ini membatalkan izin. Untuk informasi selengkapnya, lihat Kebijakan sesi dalam Panduan Pengguna IAM.

Beberapa jenis kebijakan

Ketika beberapa jenis kebijakan berlaku untuk sebuah permintaan, izin yang dihasilkan lebih rumit untuk dipahami. Untuk mempelajari cara AWS menentukan apakah akan mengizinkan permintaan saat beberapa jenis kebijakan terlibat, lihat Logika evaluasi kebijakan di Panduan Pengguna IAM.

Bagaimana MemoryDB untuk Redis bekerja dengan IAM

Sebelum Anda menggunakan IAM untuk mengelola akses ke MemoryDB, pelajari fitur IAM apa yang tersedia untuk digunakan dengan MemoryDB.

Fitur IAM yang dapat Anda gunakan dengan MemoryDB untuk Redis

Fitur IAM	Dukungan MemoryDB
Kebijakan berbasis identitas	Ya
Kebijakan berbasis sumber daya	Tidak

Fitur IAM	Dukungan MemoryDB
Tindakan kebijakan	Ya
Sumber daya kebijakan	Ya
Kunci persyaratan kebijakan	Tidak
ACL	Ya
ABAC (tanda dalam kebijakan)	Ya
Kredensial sementara	Ya
Izin pengguna utama	Ya
Peran layanan	Ya
Peran terkait layanan	Ya

Untuk mendapatkan tampilan tingkat tinggi tentang cara kerja MemoryDB dan AWS layanan lainnya dengan sebagian besar fitur IAM, lihat AWS layanan yang bekerja dengan IAM di Panduan Pengguna IAM.

Kebijakan berbasis identitas untuk MemoryDB

Mendukung kebijakan berbasis identitas	Ya

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke sebuah identitas, seperti pengguna IAM, grup pengguna IAM, atau peran IAM. Kebijakan ini mengontrol jenis tindakan yang dapat dilakukan pengguna dan peran, di sumber daya mana, dan dengan ketentuan apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Membuat kebijakan IAM dalam Panduan Pengguna IAM.

Dengan kebijakan berbasis identitas IAM, Anda dapat menentukan tindakan dan sumber daya yang diizinkan atau ditolak, serta ketentuan terkait jenis tindakan yang diizinkan atau ditolak. Anda tidak dapat menentukan pengguna utama dalam kebijakan berbasis identitas karena kebijakan ini berlaku untuk pengguna atau peran yang dilampiri kebijakan. Untuk mempelajari semua elemen yang dapat

digunakan dalam kebijakan JSON, lihat Referensi elemen kebijakan JSON IAM dalam Panduan Pengguna IAM.

Contoh kebijakan berbasis identitas untuk MemoryDB

Untuk melihat contoh kebijakan berbasis identitas MemoryDB, lihat. Contoh kebijakan berbasis identitas untuk MemoryDB untuk Redis

Kebijakan berbasis sumber daya dalam MemoryDB

Mendukung kebijakan berbasis sumber daya	Tidak

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan kepercayaan 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 yang dilampiri kebijakan tersebut, kebijakan ini menentukan jenis tindakan yang dapat dilakukan oleh pengguna utama tertentu di sumber daya tersebut dan apa ketentuannya. Anda harus menentukan pengguna utama dalam kebijakan berbasis sumber daya. Prinsipal dapat mencakup akun, pengguna, peran, pengguna federasi, atau. Layanan AWS

Untuk mengaktifkan akses lintas akun, Anda dapat menentukan seluruh akun atau entitas IAM di akun lain sebagai pengguna utama dalam kebijakan berbasis sumber daya. Menambahkan pengguna utama lintas akun ke kebijakan berbasis sumber daya bagian dari membangun hubungan kepercayaan. Ketika prinsipal dan sumber daya berbeda Akun AWS, administrator IAM di akun tepercaya juga harus memberikan izin entitas utama (pengguna atau peran) untuk mengakses sumber daya. Izin diberikan dengan melampirkan kebijakan berbasis identitas ke entitas tersebut. Namun, jika kebijakan berbasis sumber daya memberikan akses kepada pengguna utama dalam akun yang sama, kebijakan berbasis identitas lainnya tidak diperlukan. Untuk informasi selengkapnya, lihat Perbedaan peran IAM dengan kebijakan berbasis sumber daya di Panduan Pengguna IAM.

Tindakan kebijakan untuk MemoryDB

Mendukung tindakan kebijakan	Ya

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Yaitu, pengguna utama mana yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Elemen `Action` dari kebijakan JSON menjelaskan tindakan yang dapat Anda gunakan untuk mengizinkan atau menolak akses dalam sebuah kebijakan. Tindakan kebijakan biasanya memiliki nama yang sama dengan operasi AWS API terkait. Ada beberapa pengecualian, misalnya tindakan hanya izin yang tidak memiliki operasi API yang cocok. Ada juga beberapa operasi yang memerlukan beberapa tindakan dalam suatu kebijakan. Tindakan tambahan ini disebut tindakan dependen.

Menyertakan tindakan dalam suatu kebijakan untuk memberikan izin melakukan operasi terkait.

Untuk melihat daftar tindakan MemoryDB, lihat Tindakan yang Ditetapkan oleh MemoryDB untuk Redis di Referensi Otorisasi Layanan.

Tindakan kebijakan di MemoryDB menggunakan awalan berikut sebelum tindakan:

```
MemoryDB
```

Untuk menetapkan secara spesifik beberapa tindakan dalam satu pernyataan, pisahkan tindakan-tindakan tersebut dengan koma.

```
"Action": [
      "MemoryDB:action1",
      "MemoryDB:action2"
         ]
```

Anda juga dapat menentukan beberapa tindakan menggunakan wildcard (*). Sebagai contoh, untuk menentukan semua tindakan yang dimulai dengan kata `Describe`, sertakan tindakan berikut:

```
"Action": "MemoryDB:Describe*"
```

Untuk melihat contoh kebijakan berbasis identitas MemoryDB, lihat. Contoh kebijakan berbasis identitas untuk MemoryDB untuk Redis

Sumber daya kebijakan untuk MemoryDB

Mendukung sumber daya kebijakan	Ya

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Yaitu, pengguna utama mana yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Elemen kebijakan JSON `Resource` menentukan objek atau beberapa 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 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 untuk semua sumber daya.

```
"Resource": "*"
```

Untuk melihat daftar jenis sumber daya MemoryDB dan ARNnya, lihat Sumber Daya yang Ditetapkan oleh MemoryDB untuk Redis di Referensi Otorisasi Layanan. Untuk mempelajari tindakan mana yang dapat Anda tentukan ARN dari setiap sumber daya, lihat Tindakan yang Ditentukan oleh MemoryDB untuk Redis.

Untuk melihat contoh kebijakan berbasis identitas MemoryDB, lihat. Contoh kebijakan berbasis identitas untuk MemoryDB untuk Redis

Kunci kondisi kebijakan untuk MemoryDB

Mendukung kunci kondisi kebijakan spesifik layanan	Tidak

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Yaitu, pengguna utama mana yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Elemen `Condition` (atau blok `Condition`) memungkinkan Anda menentukan kondisi di mana suatu pernyataan akan diterapkan. Elemen `Condition` bersifat opsional. Anda dapat membuat ekspresi kondisional yang menggunakan operator kondisi, misalnya sama dengan atau kurang dari, untuk mencocokkan kondisi dalam kebijakan dengan nilai-nilai yang diminta.

Jika Anda menentukan beberapa elemen `Condition` dalam satu pernyataan, atau beberapa kunci dalam satu elemen `Condition`, AWS akan mengevaluasinya dengan menggunakan operasi `AND` logis. Jika Anda menentukan beberapa nilai untuk satu kunci kondisi, AWS mengevaluasi kondisi menggunakan `OR` operasi logis. Semua kondisi harus dipenuhi sebelum izin pernyataan diberikan.

Anda juga dapat menggunakan variabel placeholder saat menentukan kondisi. Misalnya, Anda dapat memberikan izin kepada pengguna IAM untuk mengakses sumber daya hanya jika izin tersebut mempunyai tag yang sesuai dengan nama pengguna IAM mereka. Untuk informasi selengkapnya, lihat Elemen kebijakan IAM: variabel dan tag di Panduan Pengguna IAM.

AWS mendukung kunci kondisi global dan kunci kondisi khusus layanan. Untuk melihat semua kunci kondisi AWS global, lihat kunci konteks kondisi AWS global di Panduan Pengguna IAM.

Untuk melihat contoh kebijakan berbasis identitas MemoryDB, lihat. Contoh kebijakan berbasis identitas untuk MemoryDB untuk Redis

Daftar kontrol akses (ACL) di MemoryDB

Mendukung ACL	Ya

Daftar kontrol akses (ACL) mengontrol pengguna utama (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACL sama dengan kebijakan berbasis sumber daya, meskipun tidak menggunakan format dokumen kebijakan JSON.

Kontrol akses berbasis atribut (ABAC) dengan MemoryDB

Mendukung ABAC (tanda dalam kebijakan)	Ya

Kontrol akses berbasis atribut (ABAC) adalah strategi otorisasi yang menentukan izin berdasarkan atribut. Dalam AWS, atribut ini disebut tag. Anda dapat melampirkan tag ke entitas IAM (pengguna atau peran) dan ke banyak AWS sumber daya. Pemberian tanda ke entitas dan sumber daya adalah langkah pertama dari ABAC. Kemudian, rancanglah kebijakan ABAC untuk mengizinkan operasi saat 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 di mana pengelolaan kebijakan menjadi rumit.

Untuk mengendalikan akses berdasarkan tag, berikan informasi tentang tag di elemen kondisi 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 hanya untuk beberapa jenis sumber daya, nilainya adalah Parsial.

Untuk informasi selengkapnya tentang ABAC, lihat Apa itu ABAC? di Panduan Pengguna IAM. Untuk melihat tutorial terkait langkah-langkah penyiapan ABAC, lihat Menggunakan kontrol akses berbasis atribut (ABAC) di Panduan Pengguna IAM.

Menggunakan kredensyal Sementara dengan MemoryDB

Mendukung kredensial sementara	Ya

Beberapa Layanan AWS tidak berfungsi saat Anda masuk menggunakan kredensyal sementara. Untuk informasi tambahan, termasuk yang Layanan AWS bekerja dengan kredensi sementara, lihat Layanan AWS yang bekerja dengan IAM di Panduan Pengguna IAM.

Anda menggunakan kredensyal sementara jika Anda masuk AWS Management Console menggunakan metode apa pun kecuali nama pengguna dan kata sandi. Misalnya, ketika Anda mengakses AWS menggunakan tautan masuk tunggal (SSO) perusahaan Anda, proses tersebut secara otomatis membuat kredensil sementara. Anda juga akan membuat kredensial sementara secara otomatis saat masuk ke konsol sebagai pengguna dan kemudian beralih peran. Untuk informasi selengkapnya tentang cara beralih peran, lihat Beralih peran (konsol) di Panduan Pengguna IAM.

Anda dapat membuat kredensyal sementara secara manual menggunakan API AWS CLI atau AWS . Anda kemudian dapat menggunakan kredensyal sementara tersebut untuk mengakses. AWS AWS merekomendasikan agar Anda secara dinamis menghasilkan kredensyal sementara alih-alih menggunakan kunci akses jangka panjang. Untuk informasi selengkapnya, lihat Kredensial keamanan sementara di IAM.

Izin utama lintas layanan untuk MemoryDB

Mendukung sesi akses maju (FAS)	Ya

Saat Anda menggunakan pengguna atau peran IAM untuk melakukan tindakan AWS, Anda dianggap sebagai prinsipal. Saat Anda menggunakan beberapa layanan, Anda mungkin melakukan tindakan yang kemudian memulai tindakan lain di layanan yang berbeda. FAS menggunakan izin dari pemanggilan utama Layanan AWS, dikombinasikan dengan permintaan Layanan AWS untuk membuat permintaan ke layanan hilir. Permintaan FAS hanya dibuat ketika layanan menerima permintaan yang memerlukan interaksi dengan orang lain Layanan AWS atau sumber daya untuk menyelesaikannya. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan saat membuat permintaan FAS, lihat Teruskan sesi akses.

Peran layanan untuk MemoryDB

Mendukung peran layanan	Ya

Peran layanan adalah peran IAM yang diambil oleh layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, mengubah, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat Membuat peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan pengguna IAM.

> **Warning**
>
> Mengubah izin untuk peran layanan dapat merusak fungsionalitas MemoryDB. Edit peran layanan hanya ketika MemoryDB memberikan panduan untuk melakukannya.

Peran terkait layanan untuk MemoryDB

Mendukung peran terkait layanan	Ya

Peran terkait layanan adalah jenis peran layanan yang ditautkan ke. Layanan AWS Layanan dapat menggunakan peran untuk melakukan tindakan atas nama Anda. Peran terkait layanan muncul di Anda Akun AWS dan dimiliki oleh layanan. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.

Untuk detail tentang pembuatan atau pengelolaan peran terkait layanan, lihat Layanan AWS yang berfungsi dengan IAM. Temukan sebuah layanan dalam tabel yang memiliki `Yes` di kolom Peran

terkait layanan. Pilih tautan Ya untuk melihat dokumentasi peran terkait layanan untuk layanan tersebut.

Contoh kebijakan berbasis identitas untuk MemoryDB untuk Redis

Secara default, pengguna dan peran tidak memiliki izin untuk membuat atau memodifikasi sumber daya MemoryDB. Mereka juga tidak dapat melakukan tugas dengan menggunakan AWS Management Console, AWS Command Line Interface (AWS CLI), atau AWS API. Untuk memberikan izin kepada pengguna untuk melakukan tindakan pada sumber daya yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM. Administrator kemudian dapat menambahkan kebijakan IAM ke peran, dan pengguna dapat menjalankan peran.

Untuk mempelajari cara membuat kebijakan berbasis identitas IAM menggunakan contoh dokumen kebijakan JSON ini, lihat Membuat kebijakan IAM dalam Panduan Pengguna IAM.

Untuk detail tentang tindakan dan jenis sumber daya yang ditentukan oleh MemoryDB, termasuk format ARN untuk setiap jenis sumber daya, lihat Tindakan, Sumber Daya, dan Kunci Kondisi untuk MemoryDB for Redis di Referensi Otorisasi Layanan.

Topik

- Praktik terbaik kebijakan
- Menggunakan konsol MemoryDB
- Izinkan pengguna melihat izin mereka sendiri

Praktik terbaik kebijakan

Kebijakan berbasis identitas menentukan apakah seseorang dapat membuat, mengakses, atau menghapus sumber daya MemoryDB di akun Anda. Tindakan ini dikenai biaya untuk Akun AWS Anda. Ketika Anda membuat atau mengedit kebijakan berbasis identitas, ikuti panduan dan rekomendasi ini:

- Mulailah dengan kebijakan AWS terkelola dan beralih ke izin hak istimewa paling sedikit — Untuk mulai memberikan izin kepada pengguna dan beban kerja Anda, gunakan kebijakan AWS terkelola yang memberikan izin untuk banyak kasus penggunaan umum. Mereka tersedia di Anda Akun AWS. Kami menyarankan Anda mengurangi izin lebih lanjut dengan menentukan kebijakan yang dikelola AWS pelanggan yang khusus untuk kasus penggunaan Anda. Untuk informasi selengkapnya, lihat kebijakan yang dikelola AWS atau kebijakan yang dikelola AWS untuk fungsi pekerjaan 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 melakukan ini dengan menentukan tindakan yang dapat diambil pada sumber daya tertentu dalam kondisi tertentu, juga dikenal sebagai izin hak akses paling rendah. Untuk informasi selengkapnya tentang cara menggunakan IAM untuk menerapkan izin, lihat Kebijakan dan izin di IAM di Panduan Pengguna IAM.
- Gunakan kondisi dalam kebijakan IAM untuk membatasi akses lebih lanjut – Anda dapat menambahkan kondisi ke kebijakan Anda untuk membatasi akses ke tindakan dan sumber daya. Sebagai contoh, Anda dapat menulis kondisi kebijakan untuk menentukan bahwa semua permintaan harus dikirim menggunakan SSL. Anda juga dapat menggunakan ketentuan untuk memberikan akses ke tindakan layanan jika digunakan melalui yang spesifik Layanan AWS, seperti AWS CloudFormation. Untuk informasi selengkapnya, lihat Elemen kebijakan JSON IAM: Syarat di Panduan Pengguna IAM.
- Menggunakan IAM Access Analyzer untuk memvalidasi kebijakan IAM Anda guna memastikan izin yang aman dan berfungsi – 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 Analizer Akses IAM di Panduan Pengguna IAM.
- Memerlukan otentikasi multi-faktor (MFA) - Jika Anda memiliki skenario yang mengharuskan pengguna IAM atau pengguna root di Anda, Akun AWS aktifkan MFA untuk keamanan tambahan. Untuk mewajibkan MFA saat operasi API dipanggil, tambahkan kondisi MFA pada kebijakan Anda. Untuk informasi selengkapnya, lihat Mengonfigurasi akses API yang dilindungi MFA di Panduan Pengguna IAM.

Untuk informasi selengkapnya tentang praktik terbaik dalam IAM, lihat Praktik terbaik keamanan di IAM di Panduan Pengguna IAM.

Menggunakan konsol MemoryDB

Untuk mengakses konsol MemoryDB untuk Redis, Anda harus memiliki set izin minimum. Izin ini harus memungkinkan Anda untuk daftar dan melihat rincian tentang sumber daya MemoryDB di Anda. Akun AWS 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 mengizinkan izin konsol minimum untuk pengguna yang melakukan panggilan hanya ke AWS CLI atau AWS API. Sebaliknya, izinkan akses hanya ke tindakan yang cocok dengan operasi API yang coba dilakukan.

Untuk memastikan bahwa pengguna dan peran masih dapat menggunakan konsol MemoryDB, lampirkan juga MemoryDB `ConsoleAccess` atau kebijakan `ReadOnly` AWS terkelola ke entitas. Untuk informasi selengkapnya, lihat Menambahkan izin ke 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 ini di konsol atau menggunakan API atau secara terprogram. AWS CLI AWS

```
{
    "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 MemoryDB untuk identitas dan akses Redis

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah umum yang mungkin Anda temui saat bekerja dengan MemoryDB dan IAM.

Topik

- Saya tidak berwenang untuk melakukan tindakan di MemoryDB
- Saya tidak berwenang untuk melakukan iam: PassRole
- Saya ingin mengizinkan orang di luar AWS akun saya untuk mengakses sumber daya MemoryDB saya

Saya tidak berwenang untuk melakukan tindakan di MemoryDB

Jika AWS Management Console memberitahu Anda bahwa Anda tidak berwenang untuk melakukan tindakan, maka Anda harus menghubungi administrator Anda untuk bantuan. Administrator adalah orang yang memberikan nama pengguna dan kata sandi kepada Anda.

Contoh kesalahan berikut terjadi ketika pengguna `mateojackson` mencoba menggunakan konsol untuk melihat detail tentang suatu sumber daya *`my-example-widget`* fiktif, tetapi tidak memiliki izin `MemoryDB:`*`GetWidget`* fiktif.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 MemoryDB:GetWidget on resource: my-example-widget
```

Dalam hal ini, Mateo meminta administratornya untuk memperbarui kebijakan miliknya agar dia dapat mengakses sumber daya *`my-example-widget`* dengan menggunakan tindakan `MemoryDB:`*`GetWidget`*.

Saya tidak berwenang untuk melakukan iam: PassRole

Jika Anda menerima kesalahan bahwa Anda tidak berwenang untuk melakukan `iam:PassRole` tindakan, kebijakan Anda harus diperbarui agar Anda dapat meneruskan peran ke MemoryDB.

Beberapa Layanan AWS memungkinkan Anda untuk meneruskan peran yang ada ke layanan tersebut alih-alih membuat peran layanan baru atau peran terkait layanan. Untuk melakukannya, Anda harus memiliki izin untuk meneruskan peran ke layanan.

Contoh kesalahan berikut terjadi ketika pengguna IAM bernama `marymajor` mencoba menggunakan konsol untuk melakukan tindakan di MemoryDB. 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 memerlukan bantuan, hubungi AWS administrator Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Saya ingin mengizinkan orang di luar AWS akun saya untuk mengakses sumber daya MemoryDB saya

Anda dapat membuat peran yang dapat digunakan pengguna di akun lain atau pengguna di luar organisasi Anda untuk mengakses sumber daya Anda. Anda dapat menentukan siapa saja 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 pengguna akses ke sumber daya Anda.

Untuk mempelajari selengkapnya, periksa hal berikut:

- Untuk mempelajari apakah MemoryDB mendukung fitur-fitur ini, lihat. Bagaimana MemoryDB untuk Redis bekerja dengan IAM
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda di seluruh sumber daya Akun AWS yang Anda miliki, lihat Menyediakan akses ke pengguna IAM di pengguna lain Akun AWS yang Anda miliki di Panduan Pengguna IAM.

- Untuk mempelajari cara menyediakan akses ke sumber daya Anda kepada pihak ketiga Akun AWS, lihat Menyediakan akses yang Akun AWS dimiliki oleh pihak ketiga dalam Panduan Pengguna IAM.
- Untuk mempelajari cara memberikan akses melalui federasi identitas, lihat Memberikan akses kepada pengguna eksternal yang sah (federasi identitas) dalam Panduan Pengguna IAM.
- Untuk mempelajari perbedaan antara penggunaan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Perbedaan antara peran IAM dan kebijakan berbasis sumber daya di Panduan Pengguna IAM.

Kontrol akses

Anda dapat memiliki kredensi yang valid untuk mengautentikasi permintaan Anda, tetapi kecuali Anda memiliki izin, Anda tidak dapat membuat atau mengakses MemoryDB untuk sumber daya Redis. Misalnya, Anda harus memiliki izin untuk membuat cluster MemoryDB.

Bagian berikut menjelaskan cara mengelola izin untuk MemoryDB untuk Redis. Anda disarankan untuk membaca gambaran umum terlebih dahulu.

- Ikhtisar mengelola izin akses ke sumber daya MemoryDB Anda
- Menggunakan kebijakan berbasis identitas (kebijakan IAM) untuk MemoryDB for Redis

Ikhtisar mengelola izin akses ke sumber daya MemoryDB Anda

Setiap AWS sumber daya dimiliki oleh AWS akun, 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, MemoryDB untuk Redis juga mendukung melampirkan kebijakan izin ke 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) 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.

Topik

- MemoryDB untuk sumber daya dan operasi Redis
- Memahami kepemilikan sumber daya
- Mengelola akses ke sumber daya
- Menggunakan kebijakan berbasis identitas (kebijakan IAM) untuk MemoryDB for Redis

- Izin tingkat sumber daya
- Menggunakan Peran Tertaut Layanan untuk Amazon MemoryDB untuk Redis
- AWSkebijakan terkelola untuk MemoryDB untuk Redis
- Izin API MemoryDB: Referensi tindakan, sumber daya, dan kondisi

MemoryDB untuk sumber daya dan operasi Redis

Dalam MemoryDB untuk Redis, sumber daya utama adalah cluster.

Sumber daya ini memiliki Amazon Resource Name (ARN) yang unik dan terkait dengannya, seperti ditunjukkan berikut ini.

> **Note**
>
> Agar izin di tingkat sumber daya menjadi efektif, nama sumber daya pada string ARN harus huruf kecil.

Jenis sumber daya	Format ARN
Pengguna	*arn:aws:memorydb: us-timur-1:123456789012: pengguna/pengguna1*
Daftar Kontrol Akses (ACL)	*arn:aws:memorydb: us-timur-1:123456789012: acl/myacl*
Klaster	*arn:aws:memorydb: us-timur-1:123456789012: cluster/my-cluster*
Snapshot	*arn:aws:memorydb: us-timur-1:123456789012: snapshot/snapshot saya*

Jenis sumber daya	Format ARN
Grup parameter	*arn:aws:memorydb: us-timur-1:123456789012: parametergroup/* my-parameter-group
Grup subnet	*arn:aws:memorydb: us-timur-1:123456789012: subnetgroup/* my-subnet-group

MemoryDB menyediakan satu set operasi untuk bekerja dengan sumber daya MemoryDB. Untuk daftar operasi yang tersedia, lihat MemoryDB for Redis Actions.

Memahami kepemilikan sumber daya

Pemilik sumber daya adalah AWS akun yang membuat sumber daya. Artinya, pemilik sumber daya adalah AWS akun entitas utama yang mengotentikasi permintaan yang membuat sumber daya. Entitas utama dapat berupa akun root, pengguna IAM, atau peran IAM. Contoh berikut menggambarkan cara kerjanya:

- Misalkan Anda menggunakan kredensyal akun root AWS akun Anda untuk membuat cluster. Dalam hal ini, AWS akun Anda adalah pemilik sumber daya. Di MemoryDB, sumber daya adalah cluster.
- Misalkan Anda membuat pengguna IAM di AWS akun Anda dan memberikan izin untuk membuat klaster ke pengguna tersebut. Dalam hal ini, pengguna dapat membuat cluster. Namun, AWS akun Anda, tempat pengguna berada, memiliki sumber daya cluster.
- Misalkan Anda membuat peran IAM di AWS akun Anda dengan izin untuk membuat klaster. Dalam hal ini, siapa pun yang dapat mengambil peran dapat membuat cluster. AWS Akun Anda, tempat peran tersebut berada, memiliki sumber daya cluster.

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 MemoryDB untuk Redis. 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 adalah untuk pengguna tersebut untuk membuat sumber daya MemoryDB, seperti cluster, 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 AWS akun lain (misalnya, Akun B) atau AWS layanan 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 kepada setiap pengguna di Akun B. Melakukan hal ini memungkinkan pengguna di Akun B untuk membuat atau mengakses sumber daya di Akun A. Dalam beberapa kasus, Anda mungkin ingin memberikan izin AWS layanan 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 ini adalah contoh kebijakan yang memungkinkan pengguna untuk melakukan `DescribeClusters` tindakan untuk AWS akun Anda. MemoryDB 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": "DescribeClusters",
      "Effect": "Allow",
      "Action": [
         "memorydb:DescribeClusters"],
      "Resource": resource-arn
      }
   ]
}
```

Untuk informasi selengkapnya tentang penggunaan kebijakan berbasis identitas dengan MemoryDB, lihat. Menggunakan kebijakan berbasis identitas (kebijakan IAM) untuk MemoryDB for Redis 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 MemoryDB untuk Redis (lihatMemoryDB untuk sumber daya dan operasi Redis), layanan mendefinisikan sekumpulan operasi API (lihat Tindakan). Untuk memberikan izin untuk operasi API ini, MemoryDB mendefinisikan serangkaian tindakan yang dapat Anda tentukan dalam kebijakan. Misalnya, untuk sumber daya cluster MemoryDB, tindakan berikut didefinisikan:`CreateCluster`,`DeleteCluster`, dan. `DescribeClusters` 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 MemoryDB untuk sumber daya dan operasi Redis.
- Tindakan – Anda menggunakan kata kunci tindakan untuk mengidentifikasi operasi sumber daya yang ingin Anda izinkan atau tolak. Misalnya, tergantung pada yang ditentukan`Effect`, `memorydb:CreateCluster` izin memungkinkan atau menolak izin pengguna untuk melakukan MemoryDB untuk operasi Redis. `CreateCluster`
- 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 menampilkan semua tindakan MemoryDB for Redis API, lihat. Izin API MemoryDB: 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.

Menggunakan kebijakan berbasis identitas (kebijakan IAM) untuk MemoryDB for Redis

Topik ini memberikan contoh kebijakan berbasis identitas di mana administrator akun dapat melampirkan kebijakan izin ke identitas IAM (yaitu, pengguna, grup, dan peran).

> **Important**
>
> Kami menyarankan Anda terlebih dahulu membaca topik yang menjelaskan konsep dasar dan opsi untuk mengelola akses ke MemoryDB untuk sumber daya Redis. Untuk informasi selengkapnya, lihat Ikhtisar mengelola izin akses ke sumber daya MemoryDB Anda.

Bagian dalam topik ini mencakup hal berikut:

- Izin yang diperlukan untuk menggunakan MemoryDB untuk konsol Redis
- AWSKebijakan (yang telah ditentukan sebelumnya) untuk MemoryDB untuk Redis
- Contoh kebijakan yang dikelola pelanggan

Berikut adalah contoh kebijakan izin.

```
{
   "Version": "2012-10-17",
   "Statement": [{
       "Sid": "AllowClusterPermissions",
       "Effect": "Allow",
       "Action": [
          "memorydb:CreateCluster",
          "memorydb:DescribeClusters",
          "memorydb:UpdateCluster"],
       "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 MemoryDB untuk tindakan Redis (`memorydb:CreateCluster`,`memorydb:DescribeClusters`, dan`memorydb:UpdateCluster`) pada klaster mana pun yang dimiliki oleh akun.
- 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 menunjukkan semua tindakan MemoryDB for Redis API dan sumber daya yang diterapkan, lihat. Izin API MemoryDB: Referensi tindakan, sumber daya, dan kondisi

Izin yang diperlukan untuk menggunakan MemoryDB untuk konsol Redis

Tabel referensi izin mencantumkan MemoryDB untuk operasi API Redis dan menunjukkan izin yang diperlukan untuk setiap operasi. Untuk informasi selengkapnya tentang operasi API MemoryDB, lihat. Izin API MemoryDB: Referensi tindakan, sumber daya, dan kondisi

Untuk menggunakan konsol MemoryDB for Redis, pertama-tama berikan izin untuk tindakan tambahan seperti yang ditunjukkan dalam kebijakan izin berikut.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "MinPermsForMemDBConsole",
        "Effect": "Allow",
        "Action": [
            "memorydb:Describe*",
            "memorydb:List*",
            "ec2:DescribeAvailabilityZones",
            "ec2:DescribeVpcs",
            "ec2:DescribeAccountAttributes",
            "ec2:DescribeSecurityGroups",
            "cloudwatch:GetMetricStatistics",
            "cloudwatch:DescribeAlarms",
            "s3:ListAllMyBuckets",
```

```
            "sns:ListTopics",
            "sns:ListSubscriptions" ],
        "Resource": "*"
        }
    ]
}
```

Konsol MemoryDB membutuhkan izin tambahan ini karena alasan berikut:

- Izin untuk tindakan MemoryDB memungkinkan konsol untuk menampilkan sumber daya MemoryDB di akun.
- Konsol memerlukan izin untuk `ec2` tindakan untuk menanyakan Amazon EC2 sehingga dapat menampilkan Availability Zone, VPC, grup keamanan, dan atribut akun.
- Izin untuk `cloudwatch` tindakan memungkinkan konsol mengambil CloudWatch metrik dan alarm Amazon, dan menampilkannya di konsol.
- Izin untuk `sns` tindakan memungkinkan konsol mengambil topik dan langganan Amazon Simple Notification Service (Amazon SNS), dan menampilkannya di konsol.

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: Izinkan pengguna memanggil IAM API CreateServiceLinkedRole). Atau Anda seharusnya telah membuat peran terkait layanan MemoryDB.

Bila dikombinasikan dengan izin minimum yang diperlukan untuk menggunakan konsol MemoryDB for Redis, contoh kebijakan di bagian ini memberikan izin tambahan. Contohnya juga relevan dengan AWS SDK dan. AWS CLI Untuk informasi selengkapnya tentang izin apa yang diperlukan untuk menggunakan konsol MemoryDB, lihat. Izin yang diperlukan untuk menggunakan MemoryDB untuk konsol Redis

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 MemoryDB yang tampak sederhana dapat memerlukan

tindakan lain untuk mendukungnya saat Anda menggunakan konsol MemoryDB. Misalnya, `memorydb:CreateCluster` memberikan izin untuk membuat cluster MemoryDB. Namun, untuk melakukan operasi ini, konsol MemoryDB menggunakan sejumlah `Describe` dan `List` tindakan untuk mengisi daftar konsol.

Contoh-contoh

- Contoh 1: Izinkan akses hanya-baca pengguna ke sumber daya MemoryDB
- Contoh 2: Izinkan pengguna untuk melakukan tugas administrator sistem MemoryDB umum
- Contoh 3: Izinkan pengguna mengakses semua tindakan API MemoryDB
- Contoh 4: Izinkan pengguna memanggil IAM API CreateServiceLinkedRole

Contoh 1: Izinkan akses hanya-baca pengguna ke sumber daya MemoryDB

Kebijakan berikut memberikan izin untuk tindakan MemoryDB yang memungkinkan pengguna mencantumkan sumber daya. Biasanya, Anda menyematkan jenis kebijakan izin ini untuk grup manajer.

```
{
   "Version": "2012-10-17",
   "Statement":[{
      "Sid": "MemDBUnrestricted",
      "Effect":"Allow",
      "Action": [
          "memorydb:Describe*",
          "memorydb:List*"],
      "Resource":"*"
      }
   ]
}
```

Contoh 2: Izinkan pengguna untuk melakukan tugas administrator sistem MemoryDB umum

Tugas administrator sistem umum termasuk memodifikasi cluster, parameter, dan kelompok parameter. Administrator sistem mungkin juga ingin mendapatkan informasi tentang peristiwa MemoryDB. Kebijakan berikut memberikan izin pengguna untuk melakukan tindakan MemoryDB untuk tugas administrator sistem umum ini. Biasanya, Anda menyematkan jenis kebijakan izin ini untuk grup administrator sistem.

```
{
   "Version": "2012-10-17",
   "Statement":[{
      "Sid": "MDBAllowSpecific",
      "Effect":"Allow",
      "Action":[
          "memorydb:UpdateCluster",
          "memorydb:DescribeClusters",
          "memorydb:DescribeEvents",
          "memorydb:UpdateParameterGroup",
          "memorydb:DescribeParameterGroups",
          "memorydb:DescribeParameters",
          "memorydb:ResetParameterGroup",],
      "Resource":"*"
      }
   ]
}
```

Contoh 3: Izinkan pengguna mengakses semua tindakan API MemoryDB

Kebijakan berikut memungkinkan pengguna untuk mengakses semua tindakan MemoryDB. Sebaiknya Anda memberikan jenis kebijakan izin ini hanya untuk pengguna administrator.

```
{
   "Version": "2012-10-17",
   "Statement":[{
      "Sid": "MDBAllowAll",
      "Effect":"Allow",
      "Action":[
          "memorydb:*" ],
      "Resource":"*"
      }
   ]
}
```

Contoh 4: Izinkan pengguna memanggil IAM API CreateServiceLinkedRole

Kebijakan berikut mengizinkan pengguna untuk memanggil API `CreateServiceLinkedRole` IAM. Kami menyarankan Anda memberikan jenis kebijakan izin ini kepada pengguna yang memanggil operasi MemoryDB mutatif.

```
{
```

```
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"CreateSLRAllows",
      "Effect":"Allow",
      "Action":[
        "iam:CreateServiceLinkedRole"
      ],
      "Resource":"*",
      "Condition":{
        "StringLike":{
          "iam:AWS ServiceName":"memorydb.amazonaws.com"
        }
      }
    }
  ]
}
```

Izin tingkat sumber daya

Anda dapat membatasi cakupan izin dengan menentukan sumber daya dalam kebijakan IAM. Banyak tindakan AWS CLI API mendukung jenis sumber daya yang bervariasi tergantung pada perilaku tindakan. 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.

Format ARN Sumber Daya MemoryDB

> **Note**
>
> Agar izin di tingkat sumber daya menjadi efektif, nama sumber daya pada string ARN harus huruf kecil.

- *Pengguna - arn:aws:memorydb: us-timur- 1:123456789012: pengguna/pengguna1*
- *ACL – arn:aws:memorydb: us-timur- 1:123456789012: acl/my-acl*

- *Cluster – arn:aws:memorydb: us-timur- 1:123456789012: cluster/my-cluster*
- *Snapshot – arn:aws:memorydb: us-timur- 1:123456789012: snapshot/snapshot saya*
- *Kelompok parameter – arn:aws:memorydb: us-east- 1:123456789012: parametergroup/* my-parameter-group
- *Grup subnet - arn:aws:memorydb: us-timur- 1:123456789012: subnetgroup/*my-subnet-group

Contoh-contoh

- Contoh 1: Izinkan pengguna akses penuh ke jenis sumber daya MemoryDB tertentu
- Contoh 2: Tolak akses pengguna ke cluster.

Contoh 1: Izinkan pengguna akses penuh ke jenis sumber daya MemoryDB tertentu

Kebijakan berikut secara eksplisit memungkinkan akses `account-id` penuh yang ditentukan ke semua sumber daya grup subnet tipe, grup keamanan, dan klaster.

```
{
        "Sid": "Example1",
        "Effect": "Allow",
        "Action": "memorydb:*",
        "Resource": [
             "arn:aws:memorydb:us-east-1:account-id:subnetgroup/*",
             "arn:aws:memorydb:us-east-1:account-id:securitygroup/*",
             "arn:aws:memorydb:us-east-1:account-id:cluster/*"
        ]
}
```

Contoh 2: Tolak akses pengguna ke cluster.

Contoh berikut secara eksplisit menyangkal `account-id` akses yang ditentukan ke cluster tertentu.

```
{
        "Sid": "Example2",
        "Effect": "Deny",
        "Action": "memorydb:*",
        "Resource": [
                "arn:aws:memorydb:us-east-1:account-id:cluster/name"
        ]
```

```
}
```

Menggunakan Peran Tertaut Layanan untuk Amazon MemoryDB untuk Redis

Amazon MemoryDB for Redis menggunakan peran terkait layanan AWS Identity and Access Management (IAM). Peran terkait layanan adalah jenis peran IAM unik yang ditautkan langsung ke AWS layanan, seperti Amazon MemoryDB untuk Redis. Amazon MemoryDB untuk peran terkait layanan Redis telah ditentukan sebelumnya oleh Amazon MemoryDB untuk Redis. Mereka menyertakan semua izin yang diperlukan layanan untuk memanggil layanan AWS atas nama klaster Anda.

Peran terkait layanan membuat pengaturan Amazon MemoryDB untuk Redis lebih mudah karena Anda tidak perlu menambahkan izin yang diperlukan secara manual. Peran sudah ada dalam AWS akun Anda tetapi ditautkan ke Amazon MemoryDB untuk kasus penggunaan Redis dan memiliki izin yang telah ditentukan sebelumnya. Hanya Amazon MemoryDB for Redis yang dapat mengambil peran ini, dan hanya peran ini yang dapat menggunakan kebijakan izin yang telah ditentukan sebelumnya. Anda dapat menghapus peran hanya setelah terlebih dahulu menghapus sumber daya terkaitnya. Ini melindungi Amazon MemoryDB untuk sumber daya Redis karena Anda tidak dapat secara tidak sengaja menghapus izin yang diperlukan untuk mengakses sumber daya.

Untuk informasi tentang layanan lain yang mendukung peran terkait layanan, silakan lihat Layanan AWS yang bisa digunakan dengan IAM dan carilah layanan yang memiliki opsi Ya di kolom Peran Terkait Layanan. Pilih Ya dengan sebuah tautan untuk melihat dokumentasi peran terkait layanan untuk layanan tersebut.

Daftar Isi

- Membersihkan Peran Tertaut Layanan
- Menghapus Peran Tertaut Layanan (Konsol IAM)
- Menghapus Peran Tertaut Layanan (IAM CLI)
- Menghapus Peran Terkait Layanan (API IAM)

Izin Peran Tertaut Layanan untuk Amazon MemoryDB untuk Redis

Amazon MemoryDB for Redis menggunakan peran terkait layanan bernama AWSServiceRoleForMemoryDB— Kebijakan ini memungkinkan MemoryDB mengelola AWS sumber daya atas nama Anda sebagaimana diperlukan untuk mengelola klaster Anda.

Kebijakan izin peran AWSServiceRoleForMemoryDB terkait layanan memungkinkan Amazon MemoryDB for Redis menyelesaikan tindakan berikut pada sumber daya yang ditentukan:

```
{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "ec2:CreateTags"
                ],
                "Resource": "arn:aws:ec2:*:*:network-interface/*",
                "Condition": {
                    "StringEquals": {
                        "ec2:CreateAction": "CreateNetworkInterface"
                    },
                    "ForAllValues:StringEquals": {
                        "aws:TagKeys": [
                            "AmazonMemoryDBManaged"
                        ]
                    }
                }
            },
            {
                "Effect": "Allow",
                "Action": [
                    "ec2:CreateNetworkInterface"
                ],
                "Resource": [
                    "arn:aws:ec2:*:*:network-interface/*",
```

```
                "arn:aws:ec2:*:*:subnet/*",
                "arn:aws:ec2:*:*:security-group/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DeleteNetworkInterface",
                "ec2:ModifyNetworkInterfaceAttribute"
            ],
            "Resource": "arn:aws:ec2:*:*:network-interface/*",
            "Condition": {
                "StringEquals": {
                    "ec2:ResourceTag/AmazonMemoryDBManaged": "true"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DeleteNetworkInterface",
                "ec2:ModifyNetworkInterfaceAttribute"
            ],
            "Resource": "arn:aws:ec2:*:*:security-group/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
```

```
                        "cloudwatch:namespace": "AWS/MemoryDB"
                    }
                }
            }
        ]
    }
```

Untuk informasi selengkapnya, lihat AWSkebijakan terkelola: MemoryDBServiceRolePolicy.

Untuk mengizinkan entitas IAM membuat peran terkait AWSServiceRoleForMemoryDB layanan

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/memorydb.amazonaws.com/
AWSServiceRoleForMemoryDB*",
    "Condition": {"StringLike": {"iam:AWS ServiceName": "memorydb.amazonaws.com"}}
}
```

Untuk mengizinkan entitas IAM menghapus peran terkait AWSServiceRoleForMemoryDB layanan

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/memorydb.amazonaws.com/
AWSServiceRoleForMemoryDB*",
    "Condition": {"StringLike": {"iam:AWS ServiceName": "memorydb.amazonaws.com"}}
}
```

Atau, Anda dapat menggunakan kebijakan AWS terkelola untuk menyediakan akses penuh ke Amazon MemoryDB for Redis.

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 kiri konsol IAM, pilih Peran. Kemudian pilih Buat peran baru.
3. Di bagian Pilih tipe entitas tepercaya, pilih Layanan AWS .
4. Di bawah Atau pilih layanan untuk melihat kasus penggunaannya, pilih MemoryDB.
5. Pilih Selanjutnya: Izin.
6. Di bagian Nama kebijakan, perhatikan bahwa `MemoryDBServiceRolePolicy` 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 terkait 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 memorydb.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 `memorydb.amazonaws.com`.

Mengedit Deskripsi Peran Tertaut Layanan untuk Amazon MemoryDB untuk Redis

Amazon MemoryDB untuk Redis tidak memungkinkan Anda mengedit peran terkait layanan. AWSServiceRoleForMemoryDB Setelah membuat peran terkait layanan, Anda tidak dapat mengubah nama peran karena berbagai entitas mungkin merujuk peran tersebut. Namun, Anda dapat menyunting Deskripsi 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 kiri 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 terkait layanan.

Untuk mengubah deskripsi peran tertaut layanan (CLI)

1. (Opsional) Untuk melihat deskripsi saat ini untuk peran, gunakan AWS CLI untuk operasi `get-role` IAM.

   Example

   ```
   $ aws iam get-role --role-name AWSServiceRoleForMemoryDB
   ```

   Gunakan nama peran, bukan ARN, untuk mengacu ke peran itu 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 terkait layanan, gunakan operasi AWS CLI untuk IAM. update-role-description

   Untuk Linux, macOS, atau Unix:

```
$ aws iam update-role-description \
    --role-name AWSServiceRoleForMemoryDB \
    --description "new description"
```

   Untuk Windows:

```
$ aws iam update-role-description ^
    --role-name AWSServiceRoleForMemoryDB ^
    --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=AWSServiceRoleForMemoryDB
   &Version=2010-05-08
   &AUTHPARAMS
```

2. Untuk memperbarui deskripsi peran, gunakan operasi API IAM UpdateRoleDescription.

   Example

```
https://iam.amazonaws.com/
   ?Action=UpdateRoleDescription
   &RoleName=AWSServiceRoleForMemoryDB
   &Version=2010-05-08
   &Description="New description"
```

Menghapus Peran Tertaut Layanan untuk Amazon MemoryDB untuk Redis

Jika tidak perlu lagi menggunakan fitur atau layanan yang memerlukan peran tertaut layanan, sebaiknya Anda menghapus peran tersebut. Dengan begitu, Anda tidak perlu lagi memantau atau memelihara entitas yang tidak digunakan. Namun, Anda harus membersihkan peran terkait layanan sebelum dapat menghapusnya.

Amazon MemoryDB for Redis tidak menghapus peran terkait layanan untuk Anda.

Membersihkan Peran Tertaut Layanan

Sebelum Anda dapat menggunakan IAM untuk menghapus peran terkait layanan, konfirmasikan terlebih dahulu bahwa peran tersebut tidak memiliki sumber daya (cluster) yang terkait dengannya.

Untuk memastikan peran terkait 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 kiri konsol IAM, pilih Peran. Kemudian pilih nama (bukan kotak centang) AWSServiceRoleForMemoryDB peran.
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 Amazon MemoryDB untuk sumber daya Redis yang memerlukan AWSServiceRoleForMemoryDB (konsol)

- Untuk menghapus klaster, lihat referensi berikut:
  - Menggunakan AWS Management Console
  - Menggunakan AWS CLI
  - Menggunakan MemoryDB API

Menghapus Peran Tertaut Layanan (Konsol IAM)

Anda dapat menggunakan konsol IAM untuk menghapus sebuah peran terkait layanan.

Untuk menghapus peran terkait layanan (konsol)

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.

2. Di panel navigasi kiri 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 halaman konfirmasi, tinjau data layanan yang terakhir diakses, yang menunjukkan kapan masing-masing peran yang dipilih terakhir mengakses AWS layanan. 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 terkait 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 terkait 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 IAM API untuk menghapus peran terkait 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.

AWSkebijakan terkelola untuk MemoryDB untuk Redis

Untuk menambahkan izin ke para pengguna, grup, dan peran, akan lebih mudah menggunakan kebijakan terkelola AWS dibandingkan dengan menulis kebijakan sendiri. Dibutuhkan waktu dan keahlian untuk membuat kebijakan terkelola pelanggan IAM yang hanya menyediakan izin sesuai kebutuhan tim Anda. Untuk mulai dengan cepat, Anda dapat menggunakan kebijakan-kebijakan terkelola AWS kami. Kebijakan-kebijakan ini mencakup kasus penggunaan umum dan tersedia

di akun AWS Anda. Untuk informasi lebih lanjut tentang kebijakan-kebijakan terkelola AWS, lihat kebijakan terkelola AWS di Panduan Pengguna IAM.

Layanan AWS mempertahankan dan memperbarui kebijakan-kebijakan terkelola AWS. Anda tidak dapat mengubah izin yang ada dalam kebijakan-kebijakan yang dikelola AWS. Layanan terkadang menambahkan izin tambahan ke kebijakan yang dikelola AWS untuk mendukung fitur-fitur baru. Jenis pembaruan ini akan memengaruhi semua identitas (pengguna, grup, dan peran) di mana kebijakan tersebut dilampirkan. Layanan kemungkinan besar akan memperbarui kebijakan yang dikelola AWS saat ada fitur baru yang diluncurkan atau saat ada operasi baru yang tersedia. Layanan tidak menghapus izin yang ada di kebijakan yang dikelola AWS, sehingga pembaruan-pembaruan yang terjadi pada kebijakan tidak akan membuat izin yang ada rusak.

Selain itu, AWS mendukung kebijakan-kebijakan terkelola untuk fungsi tugas yang mencakup beberapa layanan. Misalnya, kebijakan ReadOnlyAccessAWSterkelola memberikan akses hanya baca ke semuaAWS layanan dan sumber daya. Saat layanan meluncurkan fitur baru, AWS menambahkan izin hanya-baca untuk operasi dan sumber daya yang baru. Untuk melihat daftar dan deskripsi dari kebijakan-kebijakan fungsi tugas, lihat kebijakan terkelola AWS untuk fungsi tugas di Panduan Pengguna IAM.

AWSkebijakan terkelola: MemoryDBServiceRolePolicy

Anda tidak dapat melampirkan kebijakan yangServiceRolePolicy AWS dikelola MemoryDB ke identitas di akun Anda. Kebijakan ini adalah bagian dari peran yang terhubung layananAWS MemoryDB. Peran ini memungkinkan layanan untuk mengelola antarmuka jaringan dan grup keamanan di akun Anda.

MemoryDB menggunakan izin dalam kebijakan ini untuk mengelola grup keamanan EC2 dan antarmuka jaringan. Hal ini diperlukan untuk mengelola cluster MemoryDB.

Rincian izin

Kebijakan ini mencakup izin berikut.

```
{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "ec2:CreateTags"
                ],
                "Resource": "arn:aws:ec2:*:*:network-interface/*",
                "Condition": {
                    "StringEquals": {
                        "ec2:CreateAction": "CreateNetworkInterface"
                    },
                    "ForAllValues:StringEquals": {
                        "aws:TagKeys": [
                            "AmazonMemoryDBManaged"
                        ]
                    }
                }
            },
            {
                "Effect": "Allow",
                "Action": [
                    "ec2:CreateNetworkInterface"
                ],
                "Resource": [
                    "arn:aws:ec2:*:*:network-interface/*",
                    "arn:aws:ec2:*:*:subnet/*",
                    "arn:aws:ec2:*:*:security-group/*"
                ]
            },
            {
                "Effect": "Allow",
                "Action": [
                    "ec2:DeleteNetworkInterface",
                    "ec2:ModifyNetworkInterfaceAttribute"
                ],
                "Resource": "arn:aws:ec2:*:*:network-interface/*",
                "Condition": {
                    "StringEquals": {
                        "ec2:ResourceTag/AmazonMemoryDBManaged": "true"
                    }
                }
```

```
            },
            {
                "Effect": "Allow",
                "Action": [
                    "ec2:DeleteNetworkInterface",
                    "ec2:ModifyNetworkInterfaceAttribute"
                ],
                "Resource": "arn:aws:ec2:*:*:security-group/*"
            },
            {
                "Effect": "Allow",
                "Action": [
                    "ec2:DescribeSecurityGroups",
                    "ec2:DescribeNetworkInterfaces",
                    "ec2:DescribeAvailabilityZones",
                    "ec2:DescribeSubnets",
                    "ec2:DescribeVpcs"
                ],
                "Resource": "*"
            },
            {
                "Effect": "Allow",
                "Action": [
                    "cloudwatch:PutMetricData"
                ],
                "Resource": "*",
                "Condition": {
                    "StringEquals": {
                        "cloudwatch:namespace": "AWS/MemoryDB"
                    }
                }
            }
        ]
    }
```

AWSKebijakan (yang telah ditentukan sebelumnya) untuk MemoryDB untuk Redis

AWS menangani banyak kasus penggunaan umum dengan menyediakan kebijakan IAM mandiri yang dibuat dan dikelola oleh AWS. Kebijakan terkelola memberikan izin yang diperlukan untuk kasus penggunaan umum sehingga Anda tidak perlu menyelidiki izin apa yang diperlukan. Untuk informasi selengkapnya, lihat Kebijakan Terkelola AWS dalam Panduan Pengguna IAM.

KebijakanAWS terkelola berikut, yang dapat Anda lampirkan ke pengguna di akun Anda, khusus untuk MemoryDB:

AmazonMemoryDBReadOnlyAccess

Anda dapat melampirkan kebijakan `AmazonMemoryDBReadOnlyAccess` ke identitas-identitas IAM Anda. Kebijakan ini memberikan izin administratif yang memungkinkan akses hanya baca ke semua sumber daya MemoryDB.

AmazonMemoryDBReadOnlyAccess - Memberikan akses baca saja ke MemoryDB untuk sumber daya Redis.

```
{
 "Version": "2012-10-17",
 "Statement": [{
  "Effect": "Allow",
  "Action": [
   "memorydb:Describe*",
   "memorydb:List*"
  ],
  "Resource": "*"
 }]
}
```

AmazonMemoryDBFullAccess

Anda dapat melampirkan kebijakan `AmazonMemoryDBFullAccess` ke identitas-identitas IAM Anda. Kebijakan ini memberikan izin administratif yang memungkinkan akses penuh ke semua sumber daya MemoryDB.

AmazonMemoryDBFullAccess - Memberikan akses penuh ke MemoryDB untuk sumber daya Redis.

```
{
 "Version": "2012-10-17",
 "Statement": [{
   "Effect": "Allow",
   "Action": "memorydb:*",
   "Resource": "*"
  },
  {
   "Effect": "Allow",
```

```
   "Action": "iam:CreateServiceLinkedRole",
   "Resource": "arn:aws:iam::*:role/aws-service-role/memorydb.amazonaws.com/
AWSServiceRoleForMemoryDB",
   "Condition": {
    "StringLike": {
     "iam:AWSServiceName": "memorydb.amazonaws.com"
    }
   }
  }
 ]
}
```

Anda juga dapat membuat kebijakan IAM khusus untuk mengizinkan izin tindakan API Redis. Anda dapat melampirkan kebijakan kustom ini ke pengguna IAM atau grup yang memerlukan izin tersebut.

Pembaruan MemoryDB untuk kebijakanAWS terkelola

Lihat detail tentang pembaruan ke kebijakanAWS terkelola untuk MemoryDB karena layanan ini mulai melacak perubahan ini. Untuk peringatan otomatis tentang perubahan pada halaman ini, berlangganan umpan RSS di halaman Riwayat Dokumen MemoryDB.

Perubahan	Deskripsi	Tanggal
AmazonMemoryDBFullAccess- Menambahkan kebijakan	MemoryDB menambahkan izin baru untuk menjelaskan dan daftar sumber daya yang didukung. Izin ini diperlukan untuk MemoryDB untuk query semua sumber daya yang didukung dalam account.	10/07/2021
AmazonMemoryDBReadOnlyAccess- Menambahkan kebijakan	MemoryDB menambahkan izin baru untuk menjelaskan dan daftar sumber daya yang didukung. Izin ini diperluka n untuk MemoryDB untuk membuat aplikasi berbasis	10/07/2021

Perubahan	Deskripsi	Tanggal
	akun dengan query semua sumber daya yang didukung dalam account.	
MemoryDB mulai melacak perubahan	Peluncuran layanan	08/19/2021

Izin API MemoryDB: Referensi tindakan, sumber daya, dan kondisi

Saat Anda mengatur kontrol akses dan menulis kebijakan izin untuk dilampirkan ke kebijakan IAM (baik berbasis identitas atau berbasis sumber daya), gunakan tabel berikut sebagai referensi. Tabel mencantumkan setiap MemoryDB untuk operasi API Redis dan tindakan terkait yang dapat Anda berikan izin untuk melakukan tindakan. 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 (*).

> **Note**
>
> Untuk menentukan tindakan, gunakan awalan `memorydb:` diikuti dengan nama operasi API (misalnya, `memorydb:DescribeClusters`).

Pencatatan log dan pemantauan

Pemantauan adalah bagian penting dari menjaga keandalan, ketersediaan, dan kinerja MemoryDB untuk Redis dan solusi Anda yang lain. AWS AWS menyediakan alat pemantauan berikut untuk menonton MemoryDB, melaporkan ketika ada sesuatu yang salah, dan mengambil tindakan otomatis bila perlu:

- Amazon CloudWatch memantau AWS sumber daya Anda dan aplikasi yang Anda jalankan AWS secara real time. Anda dapat mengumpulkan dan melacak metrik, membuat dasbor yang disesuaikan, dan mengatur alarm yang memberi tahu Anda atau mengambil tindakan saat metrik tertentu mencapai ambang batas yang ditentukan. Misalnya, Anda dapat CloudWatch melacak penggunaan CPU atau metrik lain dari instans Amazon EC2 Anda dan secara otomatis meluncurkan instans baru bila diperlukan. Untuk informasi selengkapnya, lihat Panduan CloudWatch Pengguna Amazon.
- Amazon CloudWatch Logs memungkinkan Anda memantau, menyimpan, dan mengakses file log Anda dari instans Amazon EC2, CloudTrail, dan sumber lainnya. CloudWatch Log dapat memantau informasi dalam file log dan memberi tahu Anda ketika ambang batas tertentu terpenuhi. Anda juga dapat mengarsipkan data log dalam penyimpanan yang sangat durabel. Untuk informasi selengkapnya, lihat Panduan Pengguna Amazon CloudWatch Logs.

- AWS CloudTrailmenangkap panggilan API dan peristiwa terkait yang dibuat oleh atau atas nama AWS akun Anda dan mengirimkan file log ke bucket Amazon S3 yang Anda tentukan. Anda dapat mengidentifikasi pengguna dan akun mana yang dipanggil AWS, alamat IP sumber dari mana panggilan dilakukan, dan kapan panggilan terjadi. Untuk informasi selengkapnya, silakan lihat Panduan Pengguna AWS CloudTrail.

Memonitor MemoryDB untuk Redis dengan Amazon CloudWatch

Anda dapat memantau MemoryDB untuk Redis menggunakan CloudWatch, yang mengumpulkan data mentah dan memprosesnya menjadi metrik yang dapat dibaca, mendekati real-time. Statistik ini disimpan untuk jangka waktu 15 bulan, sehingga Anda dapat mengakses informasi historis dan mendapatkan perspektif yang lebih baik tentang performa aplikasi atau layanan web Anda. Anda juga dapat mengatur alarm yang memperhatikan ambang batas tertentu dan mengirim notifikasi atau mengambil tindakan saat ambang batas tersebut terpenuhi. Untuk informasi selengkapnya, lihat Panduan CloudWatch Pengguna Amazon.

Bagian berikut mencantumkan metrik dan dimensi untuk MemoryDB.

Topik

- Metrik Tingkat Host
- Metrik untuk MemoryDB
- Metrik Apa yang Harus Saya Pantau?
- Memilih Statistik dan Periode Metrik
- CloudWatch Metrik pemantauan

Metrik Tingkat Host

`AWS/MemoryDB`Namespace menyertakan metrik tingkat host berikut untuk masing-masing node.

Lihat Juga

- Metrik untuk MemoryDB

Metrik	Deskripsi	Unit
`CPUUtilization`	Persentase pemanfaatan CPU untuk keseluruhan host. Karena Redis bersifat thread tunggal, sebaiknya Anda memantau metrik `EngineCPUUtilization` untuk simpul dengan jumlah vCPU 4 atau lebih.	Persen
`FreeableMemory`	Jumlah memori kosong yang tersedia di host. Ini berasal dari RAM, buffer, dan bahwa OS melaporkan sebagai freeable.	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 masuk dari segi jumlah paket pada satu instans tunggal.	Hitungan
`NetworkPacketsOut`	Jumlah paket yang dikirimkan di semua antarmuka jaringan oleh instans. Metrik ini mengidentifikasi volume lalu lintas keluar dari segi jumlah paket pada satu instans tunggal.	Hitungan
`NetworkBandwidthInAllowanceExceeded`	Jumlah paket yang di-shaping karena bandwidth agregat masuk melebihi nilai maksimum untuk instans.	Hitungan
`NetworkConntrackAllowanceExceeded`	Jumlah paket yang di-shaping karena pelacakan koneksi melebihi nilai maksimum untuk instans dan koneksi baru tidak dapat dibuat. Hal ini dapat mengakibatkan hilangnya paket untuk lalu lintas ke atau dari instans.	Hitungan

Metrik	Deskripsi	Unit
NetworkBandwidthOutAllowanceExceeded	Jumlah paket yang di-shaping karena bandwidth agregat keluar melebihi nilai maksimum untuk instans.	Hitungan
NetworkPacketsPerSecondAllowanceExceeded	Jumlah paket yang di-shaping karena paket dua arah per detik melebihi nilai maksimum untuk instans.	Hitungan
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.	Hitungan
NetworkMaxPacketsOut	Semburan maksimum paket yang ditransmisikan dalam setiap menit.	Hitungan
SwapUsage	Jumlah swap yang digunakan oleh host.	Byte

Metrik untuk MemoryDB

Namespace AWS/MemoryDB mencakup metrik Redis berikut.

Dengan pengecualian ReplicationLag dan EngineCPUUtilization, metrik ini berasal dari perintah info Redis. Setiap metrik dihitung pada tingkat simpul.

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
`BytesUsedForMemoryDB`	Jumlah total byte yang dialokasikan oleh MemoryDB untuk semua tujuan, termasuk dataset, buffer, dan sebagainya.	Byte
	`Dimension: Tier=SSD` untuk cluster menggunakanTingkatan data: Jumlah total byte yang digunakan oleh SSD.	Byte
	`Dimension: Tier=Memory` untuk cluster menggunakanTingkatan data: Jumlah total byte yang digunakan oleh memori. Ini adalah nilai statistik `used_memory` di Redis INFO.	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
`CommandAuthorizationFailures`	Jumlah seluruh upaya pengguna yang gagal untuk menjalankan perintah karena tidak	Hitungan

Metrik	Deskripsi	Unit
	memiliki izin untuk memanggil perintah itu. Anda dapat menemukan informasi selengkap nya tentang setiap kegagalan autentikasi menggunakan perintah ACL LOG. Sebaiknya atur peringatan untuk hal ini guna mendeteksi upaya akses yang tidak sah.	
`CurrConnections`	Jumlah koneksi klien, tidak termasuk koneksi dari replika baca. MemoryDB 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 menjumlah kan 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.	Hitungan
`DatabaseMemoryUsag ePercentage`	Persentase memori yang tersedia untuk klaster yang sedang digunakan. Ini dihitung dengan menggunakan `used_memory/maxmemory` dari INFO Redis.	Persen
`DB0AverageTTL`	Memaparkan `avg_ttl` dari DBO, dari statistik `keyspace` dari perintah INFO Redis.	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. **Note** Pada host MemoryDB, proses latar belakang memantau host untuk memberikan pengalaman database 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.	Hitungan
`IsPrimary`	Menunjukkan apakah node adalah simpul utama dari pecahan saat ini. Metrik ini dapat bernilai 0 (bukan primer) atau 1 (primer).	Jumlah
`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 peringata n untuk hal ini guna mendeteksi upaya akses yang tidak sah.	Hitungan
`KeyspaceHits`	Jumlah pencarian kunci hanya-baca yang berhasil di kamus utama. Metrik ini berasal dari statistik `keyspace_hits` di Redis INFO.	Hitungan
`KeyspaceMisses`	Jumlah pencarian kunci hanya-baca yang tidak berhasil di kamus utama. Metrik ini berasal dari statistik `keyspace_misses` di Redis INFO.	Hitungan

Metrik	Deskripsi	Unit
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.	Hitungan
MaxReplicationThroughput	Throughput replikasi maksimum yang diamati selama siklus pengukuran terakhir.	Byte per detik
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
NewConnections	Jumlah seluruh koneksi yang telah diterima oleh server selama periode ini. Ini berasal dari statistik total_connections_received di INFO Redis.	Hitungan
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
PrimaryLinkHealthStatus	Status ini memiliki dua nilai: 0 atau 1. Nilai 0 menunjukkan bahwa data dalam node utama MemoryDB tidak sinkron dengan Redis pada EC2. Nilai 1 menunjukkan bahwa data sudah sinkron.	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 cluster. Metrik ini berasal dari statistik `master_repl_offset` di Redis INFO.	Byte
`ReplicationDelayedWriteCommands`	Jumlah perintah tulis yang tertunda karena replikasi sinkron. Replikasi dapat tertunda karena berbagai faktor, misalnya kemacetan jaringan atau melebihi throughput replikasi maksimum.	Hitungan
`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.	Detik

Berikut ini adalah kumpulan jenis perintah tertentu, yang berasal dari info commandstats. Bagian commandstats menyediakan statistik berdasarkan jenis perintah, termasuk jumlah panggilan.

Untuk daftar lengkap perintah yang tersedia, lihat perintah redis di dokumentasi Redis.

Metrik	Deskripsi	Unit
`EvalBasedCmds`	Jumlah seluruh perintah untuk perintah berbasis eval. Ini berasal dari statistik `commandstats` Redis. Ini berasal dari statistik `commandstats` Redis dengan menjumlahkan eval, evalsha.	Hitungan

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.	Hitungan
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.)	Hitungan
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).	Hitungan
HyperLogLogBasedCmds	Jumlah seluruh perintah berbasis HyperLogLog . Ini berasal dari statistik commandstats Redis dengan menjumlahkan semua jenis perintah pf (pfadd, pfcount, pfmerge, dan sebagainya.).	Hitungan
JsonBasedCmds	Jumlah total perintah yang berbasis JSON. Ini berasal dari commandstats statistik Redis dengan menjumlahkan semua perintah yang bertindak atas satu atau lebih objek dokumen JSON.	Hitungan

Metrik	Deskripsi	Unit
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.).	Hitungan
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).	Hitungan
PubSubBasedCmds	Jumlah seluruh perintah untuk fungsionalitas pub/sub. Ini berasal dari commandstats statistik Redis dengan menjumlahkan semua perintah yang digunakan untuk fungsiona litas pub/sub:psubscribe,,,, publishpubsub, punsubscribe dan. subscribe unsubscribe	Hitungan
SearchBasedCmds	Jumlah total indeks sekunder dan perintah pencarian, termasuk perintah baca dan tulis. Ini berasal dari commandstats statistik Redis dengan menjumlahkan semua perintah pencarian yang bertindak atas indeks sekunder.	Hitungan
SearchBasedGetCmds	Jumlah total indeks sekunder dan perintah pencarian hanya-baca. Ini berasal dari commandstats statistik Redis dengan menjumlahkan semua perintah indeks sekunder dan pencarian get.	Hitungan

Metrik	Deskripsi	Unit
SearchBasedSetCmds	Jumlah total indeks sekunder dan perintah tulis pencarian. Ini berasal dari commandstats statistik Redis dengan menjumlahkan semua perintah indeks sekunder dan set pencarian.	Hitungan
SearchNumberOfIndexes	Jumlah total indeks.	Hitungan
SearchNumberOfIndexedKeys	Jumlah total kunci Redis yang diindeks	Hitungan
SearchTotalIndexSize	Memori (byte) yang digunakan oleh semua indeks.	Byte
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).	Hitungan
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.)	Hitungan
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).	Hitungan

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 beberapa string (strlen, setex, setrange, dan sebagainya).	Hitungan
`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).	Hitungan

Metrik Apa yang Harus Saya Pantau?

CloudWatch Metrik berikut menawarkan wawasan yang baik tentang kinerja MemoryDB. Dalam kebanyakan kasus, kami menyarankan Anda menyetel CloudWatch alarm untuk metrik ini sehingga Anda dapat mengambil tindakan korektif sebelum masalah kinerja terjadi.

Metrik yang Perlu Dipantau

- CPUUtilization
- EngineCPUUtilization
- SwapUsage
- Pengosongan
- CurrConnections
- Memori
- Jaringan
- Replikasi

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 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%. Untuk mengetahui jumlah core (vCPU) yang dimiliki tipe node Anda, lihat Harga MemoryDB.

Anda perlu menentukan ambang batas Anda sendiri, berdasarkan jumlah core di node yang Anda gunakan. Jika Anda melebihi ambang batas ini, dan beban kerja utama Anda berasal dari permintaan baca, skala klaster Anda dengan menambahkan replika baca. Jika beban kerja utama berasal dari permintaan tulis, kami sarankan Anda menambahkan lebih banyak pecahan untuk mendistribusikan beban kerja tulis di lebih banyak node utama.

> **Tip**
>
> Alih-alih menggunakan metrik Host-Level`CPUUtilization`, Anda mungkin dapat menggunakan metrik Redis`EngineCPUUtilization`, yang melaporkan persentase penggunaan pada inti mesin Redis. Untuk melihat apakah metrik ini tersedia di node Anda dan untuk informasi selengkapnya, lihat Metrik untuk MemoryDB.

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 node Anda dan untuk informasi selengkapnya, lihat Metrik untuk MemoryDB.

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 node Anda dan untuk informasi selengkapnya, lihat Metrik untuk MemoryDB.

SwapUsage

Ini adalah metrik tingkat host yang dilaporkan dalam byte. Untuk informasi selengkapnya, lihat Metrik Tingkat Host.

Metrik ini tidak boleh melebihi 50 MB.

Pengosongan

Ini adalah metrik mesin. Sebaiknya tentukan ambang batas alarm Anda sendiri untuk metrik ini berdasarkan kebutuhan aplikasi Anda.

CurrConnections

Ini adalah metrik mesin. Sebaiknya tentukan ambang batas alarm Anda sendiri untuk metrik ini berdasarkan kebutuhan aplikasi Anda.

Peningkatan jumlah CurrConnectionsmungkin menunjukkan masalah dengan aplikasi Anda; Anda perlu menyelidiki perilaku aplikasi untuk mengatasi masalah ini.

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.

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 node Anda, lihat harga Amazon MemoryDB.

Replikasi

Volume data yang direplikasi akan terlihat melalui metrik `ReplicationBytes`. Anda dapat memantau `MaxReplicationThroughput` terhadap throughput kapasitas replikasi. Disarankan untuk menambahkan lebih banyak pecahan saat mencapai throughput kapasitas replikasi maksimum.

`ReplicationDelayedWriteCommands`juga dapat menunjukkan apakah beban kerja melebihi throughput kapasitas replikasi maksimum. Untuk informasi lebih lanjut tentang replikasi di MemoryDB, lihat Memahami replikasi MemoryDB

Memilih Statistik dan Periode Metrik

Meskipun CloudWatch akan memungkinkan Anda untuk memilih statistik dan periode 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 MemoryDB diterbitkan untuk durasi 60 detik untuk setiap node individu. Untuk periode 60 detik, metrik simpul hanya akan berisi satu sampel.

CloudWatch Metrik pemantauan

MemoryDB dan CloudWatch terintegrasi sehingga Anda dapat mengumpulkan berbagai metrik. Anda dapat memantau metrik ini menggunakan CloudWatch.

> **Note**
>
> Contoh berikut memerlukan alat baris CloudWatch perintah. Untuk informasi selengkapnya tentang CloudWatch dan untuk mengunduh alat pengembang, lihat halaman CloudWatch produk.

Prosedur berikut menunjukkan kepada Anda cara menggunakan CloudWatch untuk mengumpulkan statistik ruang penyimpanan untuk sebuah cluster selama satu jam terakhir.

> **Note**
>
> Nilai `StartTime` dan `EndTime` nilai yang diberikan dalam contoh berikut adalah untuk tujuan ilustrasi. Pastikan untuk mengganti nilai waktu mulai dan akhir yang sesuai untuk node Anda.

Untuk informasi tentang batas MemoryDB, lihat batas AWS layanan untuk MemoryDB.

CloudWatch Metrik pemantauan (Konsol)

Untuk mengumpulkan statistik pemanfaatan CPU untuk sebuah cluster

1. Masuk ke AWS Management Console dan buka konsol MemoryDB for Redis di https://console.aws.amazon.com/memorydb/.
2. Pilih node yang ingin Anda lihat metrik.

> **Note**
>
> Memilih lebih dari 20 simpul akan menonaktifkan tampilan metrik pada konsol.

a. Pada halaman Clusters dari AWS Management Console, klik nama satu atau beberapa cluster.

   Halaman detail untuk cluster muncul.

b. Klik tab Simpul di bagian atas jendela.

c. Pada tab Nodes pada jendela detail, pilih node yang ingin Anda lihat metrik.

   Daftar CloudWatch Metrik yang tersedia muncul di bagian bawah jendela konsol.

d. Klik metrik Pemanfaatan CPU.

   CloudWatch Konsol akan terbuka, menampilkan metrik yang Anda pilih. Anda dapat menggunakan kotak daftar drop-down Statistik dan Periode, serta tab Rentang waktu untuk mengubah metrik yang ditampilkan.

Memantau CloudWatch metrik menggunakan CLI CloudWatch

Untuk mengumpulkan statistik pemanfaatan CPU untuk sebuah cluster

- Gunakan CloudWatch perintah aws cloudwatch get-metric-statistics dengan parameter berikut (perhatikan bahwa waktu mulai dan akhir ditampilkan sebagai contoh saja; Anda harus mengganti waktu mulai dan akhir yang sesuai):

  Untuk Linux, macOS, atau Unix:

```
aws cloudwatch get-metric-statistics CPUUtilization \
    --dimensions=ClusterName=mycluster,NodeId=0002" \
    --statistics=Average \
    --namespace="AWS/MemoryDB" \
    --start-time 2013-07-05T00:00:00 \
    --end-time 2013-07-06T00:00:00 \
    --period=60
```

  Untuk Windows:

```
mon-get-stats CPUUtilization ^
    --dimensions=ClusterName=mycluster,NodeId=0002" ^
    --statistics=Average ^
    --namespace="AWS/MemoryDB" ^
    --start-time 2013-07-05T00:00:00 ^
    --end-time 2013-07-06T00:00:00 ^
    --period=60
```

Memantau CloudWatch metrik menggunakan API CloudWatch

Untuk mengumpulkan statistik pemanfaatan CPU untuk sebuah cluster

- Panggil CloudWatch API `GetMetricStatistics` dengan parameter berikut (perhatikan bahwa waktu mulai dan akhir ditampilkan sebagai contoh saja; Anda perlu mengganti waktu mulai dan akhir yang sesuai):
  - `Statistics.member.1=Average`
  - `Namespace=AWS/MemoryDB`
  - `StartTime=2013-07-05T00:00:00`
  - `EndTime=2013-07-06T00:00:00`
  - `Period=60`
  - `MeasureName=CPUUtilization`
  - `Dimensions=ClusterName=mycluster,NodeId=0002`

  Example

```
http://monitoring.amazonaws.com/
    ?SignatureVersion=4
    &Action=GetMetricStatistics
    &Version=2014-12-01
    &StartTime=2013-07-16T00:00:00
    &EndTime=2013-07-16T00:02:00
    &Period=60
    &Statistics.member.1=Average
    &Dimensions.member.1="ClusterName=mycluster"
    &Dimensions.member.2="NodeId=0002"
```

```
&Namespace=Amazon/memorydb
&MeasureName=CPUUtilization
&Timestamp=2013-07-07T17%3A48%3A21.746Z
&AWS;AccessKeyId=<&AWS; Access Key ID>
&Signature=<Signature>
```

Memonitor MemoryDB untuk acara Redis

Ketika peristiwa penting terjadi untuk sebuah cluster, MemoryDB mengirimkan pemberitahuan 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 pemberitahuan MemoryDB Amazon SNS
- Melihat acara MemoryDB
- Pemberitahuan Acara dan Amazon SNS

Mengelola pemberitahuan MemoryDB Amazon SNS

Anda dapat mengonfigurasi MemoryDB 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 AWS Console, the AWS CLI, atau MemoryDB API.

Menambahkan topik Amazon SNS (Konsol)

Prosedur berikut menunjukkan cara menambahkan topik Amazon SNS untuk klaster.

> **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 MemoryDB for Redis di https://console.aws.amazon.com/memorydb/.
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 (CLI AWS)

Untuk menambah atau memodifikasi topik Amazon SNS untuk klaster, gunakan perintah. AWS CLI `update-cluster`

Contoh kode berikut menambahkan arn topik Amazon SNS ke my-cluster.

Untuk Linux, macOS, atau Unix:

```
aws memorydb update-cluster \
    --cluster-name my-cluster \
    --sns-topic-arn arn:aws:sns:us-east-1:565419523791:memorydbNotifications
```

Untuk Windows:

```
aws memorydb update-cluster ^
    --cluster-name my-cluster ^
    --sns-topic-arn arn:aws:sns:us-east-1:565419523791:memorydbNotifications
```

Untuk informasi lebih lanjut, lihat UpdateCluster.

Menambahkan topik Amazon SNS (MemoryDB API)

Untuk menambahkan atau memperbarui topik Amazon SNS untuk kluster, panggil `UpdateCluster` tindakan dengan parameter berikut:

- `ClusterName=my-cluster`
- `SnsTopicArn=arn%3Aaws%3Asns%3Aus-east-1%3A565419523791%3AmemorydbNotifications`

Untuk menambahkan atau memperbarui topik Amazon SNS untuk klaster, panggil tindakan tersebut. `UpdateCluster`

Untuk informasi lebih lanjut, lihat UpdateCluster.

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 MemoryDB for Redis di https://console.aws.amazon.com/memorydb/.
2. Pilih tombol radio di sebelah kiri cluster yang ingin Anda ubah notifikasi.
3. Pilih Ubah.
4. Di Ubah Klaster di bagian Topik untuk Notifikasi SNS, pilih Nonaktifkan Notifikasi.
5. Pilih Ubah.

Mengaktifkan dan menonaktifkan notifikasi Amazon SNS (CLI)AWS

Untuk nonaktifkan notifikasi Amazon SNS, gunakan perintah `update-cluster` dengan parameter berikut:

Untuk Linux, macOS, atau Unix:

```
aws memorydb update-cluster \
    --cluster-name my-cluster \
```

```
    --sns-topic-status inactive
```

Untuk Windows:

```
aws memorydb update-cluster ^
    --cluster-name my-cluster ^
    --sns-topic-status inactive
```

Mengaktifkan dan menonaktifkan notifikasi Amazon SNS (MemoryDB API)

Untuk nonaktifkan notifikasi Amazon SNS, panggil tindakan `UpdateCluster` dengan parameter berikut:

- `ClusterName=my-cluster`
- `SnsTopicStatus=inactive`

Panggilan ini menghasilkan output seperti yang berikut ini:

Example

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=UpdateCluster
    &ClusterName=my-cluster
    &SnsTopicStatus=inactive
    &Version=2021-01-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210801T220302Z
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Date=20210801T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

Melihat acara MemoryDB

MemoryDB mencatat peristiwa yang berhubungan dengan cluster, grup keamanan, dan grup parameter 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 MemoryDB, AWS CLI `describe-events` perintah, atau tindakan API MemoryDB. `DescribeEvents`

Prosedur berikut menunjukkan cara melihat semua acara MemoryDB selama 24 jam terakhir (1440 menit).

Melihat acara MemoryDB (Konsol)

Prosedur berikut menampilkan peristiwa menggunakan konsol MemoryDB.

Untuk melihat acara menggunakan konsol MemoryDB

1. Masuk ke AWS Management Console dan buka konsol MemoryDB for Redis di https://console.aws.amazon.com/memorydb/.
2. Di panel navigasi kiri, pilih Acara.

   Layar Acara muncul mencantumkan semua acara yang tersedia. Setiap baris daftar mewakili satu peristiwa dan menampilkan sumber peristiwa, jenis acara (seperti cluster, parameter-grup, acl, security-group atau subnet group), waktu GMT acara, dan deskripsi acara.

   Dengan menggunakan Filter Anda dapat menentukan apakah Anda ingin melihat semua peristiwa, atau hanya peristiwa dari jenis tertentu dalam daftar peristiwa.

Melihat acara MemoryDB (CLI AWS)

Untuk menghasilkan daftar peristiwa MemoryDB 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 cluster.

```
aws memorydb describe-events --source-type cluster --max-results 40
```

Kode berikut mencantumkan semua peristiwa selama 24 jam terakhir (1.440 menit).

```
aws memorydb describe-events --duration 1440
```

Output dari perintah `describe-events` terlihat seperti berikut ini.

```
{
    "Events": [
        {
            "Date": "2021-03-29T22:17:37.781Z",
            "Message": "Added node 0001 in Availability Zone us-east-1a",
            "SourceName": "memorydb01",
            "SourceType": "cluster"
        },
        {
            "Date": "2021-03-29T22:17:37.769Z",
            "Message": "cluster created",
            "SourceName": "memorydb01",
            "SourceType": "cluster"
        }
    ]
}
```

Untuk informasi selengkapnya, seperti parameter yang tersedia dan nilai parameter yang diizinkan, lihat `describe-events`.

Melihat acara MemoryDB (MemoryDB API)

Untuk menghasilkan daftar peristiwa MemoryDB menggunakan API MemoryDB, 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 -cluster terbaru.

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=DescribeEvents
   &MaxResults=40
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &SourceType=cluster
   &Timestamp=20210802T192317Z
   &Version=2021-01-01
```

```
   &X-Amz-Credential=<credential>
```

Kode berikut mencantumkan peristiwa cluster selama 24 jam terakhir (1440 menit).

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=DescribeEvents
   &Duration=1440
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &SourceType=cluster
   &Timestamp=20210802T192317Z
   &Version=2021-01-01
   &X-Amz-Credential=<credential>
```

Perintah di atas akan menghasilkan output yang serupa dengan berikut ini.

```
<DescribeEventsResponse xmlns="http://memory-db.us-east-1.amazonaws.com/
doc/2021-01-01/">
    <DescribeEventsResult>
        <Events>
            <Event>
                <Message>cluster created</Message>
                <SourceType>cluster</SourceType>
                <Date>2021-08-02T18:22:18.202Z</Date>
                <SourceName>my-memorydb-primary</SourceName>
            </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`.

Pemberitahuan Acara dan Amazon SNS

MemoryDB dapat mempublikasikan pesan menggunakan Amazon Simple Notification Service (SNS) ketika peristiwa penting terjadi pada klaster. Fitur ini dapat digunakan untuk me-refresh server-list pada mesin klien yang terhubung ke titik akhir node individual cluster.

> **Note**
>
> Untuk informasi selengkapnya tentang Amazon Simple Notification Service (SNS), termasuk informasi tentang harga dan tautan ke dokumentasi Amazon SNS, lihatHalaman produk Amazon SNS.

Notifikasi dipublikasikan ke Amazon SNS yang ditentukantema. Berikut ini adalah persyaratan untuk notifikasi:

- Hanya satu topik yang dapat dikonfigurasi untuk notifikasi MemoryDB.
- ParameterAWSakun yang memiliki topik Amazon SNS harus berupa akun yang sama yang memiliki klaster tempat notifikasi diaktifkan.

Peristiwa MemoryDB

Peristiwa MemoryDB berikut memicu pemberitahuan Amazon SNS:

Nama peristiwa	Message	Deskripsi
MemoryDB:addnoDecomplete	`"Modified number of nodes from %d to %d"`	Sebuah node telah ditambahkan ke cluster dan siap digunakan.
MemoryDB:addNoDefailed karena alamat IP gratis yang tidak mencukupi	`"Failed to modify number of nodes from %d to %d due to insufficient free IP addresses"`	Sebuah node tidak dapat ditambahkan karena tidak ada cukup alamat IP yang tersedia.
MemoryDB: ClusterParametersChanged	`"Updated parameter group for the cluster"`	Satu atau lebih parameter klaster telah diubah.

Nama peristiwa	Message	Deskripsi
	Dalam kasus membuat, juga mengirim`"Updated to use a ParameterGroup %s"`	
MemoryDB:ClusterProvisioningComplete	`"Cluster created."`	Penyediaan cluster selesai, dan node dalam cluster siap digunakan.
MemoryDB:ClusterProvisioningGagal karena keadaan jaringan yang tidak kompatibel	`"Failed to create cluster due to incompatible network state. %s"`	Upaya dilakukan untuk meluncurkan klaster baru ke cloud pribadi virtual (VPC) yang tidak ada.
MemoryDB:ClusterRestoregagal	`"Restore from %s failed for node %s. %s"`	MemoryDB tidak dapat mengisi cluster dengan data snapshot Redis. Hal ini dapat disebabkan oleh file snapshot yang tidak ada di Amazon S3, atau izin yang salah pada file tersebut. Jika Anda menggambarkan klaster, statusnya akan`restore-failed` . Anda harus menghapus cluster dan memulai dari awal. Untuk informasi selengkapnya, lihat Menyemai cluster baru dengan snapshot yang dibuat secara eksternal.
MemoryDB:ClusterscalingComplete	`"Succeeded applying modification to node type to %s."`	Skala untuk klaster berhasil diselesaikan.

Nama peristiwa	Message	Deskripsi
MemoryDB:ClusterScalingGagal	`"Failed applying modification to node type to %s."`	Operasi skala-up pada klaster gagal.
memorydb:ClusterSecurityGroupModified	`"Modified security group for cluster."`	Salah satu peristiwa berikut telah terjadi: • Daftar kelompok keamanan yang berwenang untuk klaster telah dimodifikasi. • Satu atau lebih grup keamanan EC2 baru telah diotorisasi pada salah satu kelompok keamanan yang terkait dengan klaster. • Satu atau lebih grup keamanan EC2 telah dicabut dari salah satu kelompok keamanan yang terkait dengan klaster.

Nama peristiwa	Message	Deskripsi
MemoryDB:noderePlaceTarted	`"Recovering node %s"`	MemoryDB telah mendeteksi bahwa host menjalankan node terdegradasi atau tidak terjangkau dan telah mulai mengganti node. **Note** Entri DNS untuk node diganti tidak berubah. Pada kebanyakan kasus, Anda tidak perlu menyegarkan daftar server untuk klien Anda ketika peristiwa ini terjadi. Namun, beberapa library klien mungkin berhenti menggunakan node bahkan setelah MemoryDB telah menggantikan node; dalam hal ini, aplikasi harus me-refresh server-list ketika peristiwa ini terjadi.

Nama peristiwa	Message	Deskripsi
MemoryDB:noderePlaceComplete	`"Finished recovery for node %s"`	MemoryDB telah mendeteksi bahwa host menjalankan node terdegradasi atau tidak terjangkau dan telah selesai menggantikan node. **Note** Entri DNS untuk node diganti tidak berubah. Pada kebanyakan kasus, Anda tidak perlu menyegarkan daftar server untuk klien Anda ketika peristiwa ini terjadi. Namun, beberapa library klien mungkin berhenti menggunakan node bahkan setelah MemoryDB telah menggantikan node; dalam hal ini, aplikasi harus me-refresh server-list ketika peristiwa ini terjadi.
MemoryDB:menciptakanLusterComplete	`"Cluster created"`	Cluster berhasil dibuat.
MemoryDB:CreateClusterGagal	`"Failed to create cluster due to unsuccessful creation of its node(s)."` dan `"Deleting all nodes belonging to this cluster."`	Cluster tidak diciptakan.

Nama peristiwa	Message	Deskripsi
MemoryDB:deleteclusterComplete	`"Cluster deleted."`	Penghapusan cluster dan semua node terkait telah selesai.
MemoryDB:FailOvercomplete	`"Failover to replica node %s completed"`	Failover ke simpul replika berhasil.
MemoryDB:NoderePlacementDibatalkan	`"The replacement of node %s which was scheduled during the maintenance window from start time: %s, end time: %s has been canceled"`	Sebuah simpul di klaster Anda yang dijadwalkan untuk penggantian tidak lagi dijadwalkan untuk pengganti an.
MemoryDB:NoderePlacementDirecheduled	`"The replacement in maintenance window for node %s has been re-scheduled from previous start time: %s, previous end time: %s to new start time: %s, new end time: %s"`	Sebuah simpul di klaster Anda yang sebelumnya dijadwalkan untuk pengganti an telah dijadwal ulang untuk penggantian selama jendela baru yang dijelaskan dalam notifikasi. Untuk informasi tentang tindakan yang dapat Anda lakukan, lihat Mengganti simpul.

Nama peristiwa	Message	Deskripsi
MemoryDB:noderePlacementDijadwalkan	`"The node %s is scheduled for replacement during the maintenance window from start time: %s to end time: %s"`	Sebuah simpul di klaster Anda dijadwalkan untuk pengganti an selama jendela yang dijelaskan dalam notifikasi. Untuk informasi tentang tindakan yang dapat Anda lakukan, lihat Mengganti simpul.
MemoryDB:removenoDecplete	`"Removed node %s"`	Sebuah node telah dihapus dari cluster.
MemoryDB:SnapshotComplete	`"Snapshot %s succeeded for node %s"`	Snapshot telah berhasil diselesaikan.
MemoryDB:SnapshotGagal	`"Snapshot %s failed for node %s"`	Snapshot telah gagal. Lihat peristiwa cluster untuk penyebab yang lebih rinci. Jika Anda menggambarkan snapshot, lihatDescribeSnapshots, statusnya akan`failed`.

Logging MemoryDB untuk Redis API panggilan denganAWS CloudTrail

MemoryDB untuk Redis terintegrasi denganAWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atauAWS layanan di MemoryDB untuk Redis. CloudTrail menangkap semua panggilan API untuk MemoryDB untuk Redis sebagai tindakan, termasuk panggilan dari MemoryDB untuk konsol Redis dan dari panggilan kode ke MemoryDB untuk operasi API Redis. Jika Anda membuat jejak, Anda dapat mengaktifkan pengiriman berkelanjutan dari CloudTrail bucket Amazon S3, termasuk peristiwa untuk MemoryDB untuk Redis. Jika Anda tidak mengonfigurasi jejak, Anda masih dapat melihat peristiwa terbaru dalam CloudTrail konsol di Riwayat peristiwa. Menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan

permintaan yang dibuat ke MemoryDB untuk Redis, alamat IP asal permintaan tersebut dibuat, siapa yang membuat permintaan, kapan permintaan dibuat, dan detail lainnya.

Untuk mempelajari lebih lanjut CloudTrail, lihat PanduanAWS CloudTrail Pengguna.

MemoryDB untuk informasi Redis di CloudTrail

CloudTrail diaktifkan diAWS akun Anda saat Anda membuat akun tersebut. Ketika aktivitas terjadi di MemoryDB untuk Redis, aktivitas tersebut dicatat dalam CloudTrail peristiwa bersama peristiwaAWS layanan lainnya dalam riwayat Peristiwa. Anda dapat melihat, mencari, dan mengunduh peristiwa terbaru di akun AWS Anda. Untuk informasi selengkapnya, lihat Melihat Peristiwa dengan Riwayat CloudTrail Peristiwa.

Untuk catatan berkelanjutan tentang peristiwa diAWS akun Anda, termasuk peristiwa untuk MemoryDB untuk Redis, buat jejak. Jejak CloudTrail memungkinkan pengiriman file log ke bucket Amazon S3. Secara default, saat Anda membuat lintasan di konsol, lintasan tersebut berlaku untuk semua wilayah . Jejak mencatat kejadian dari semua wilayah dalam partisi AWS dan mengirimkan file log ke bucket Amazon S3 yang Anda tentukan. Selain itu, Anda dapat mengonfigurasiAWS layanan lainnya untuk menganalisis lebih lanjut dan bertindak berdasarkan data peristiwa yang dikumpulkan di CloudTrail log. Untuk informasi selengkapnya, lihat yang berikut:

- Ikhtisar untuk Membuat Jejak
- CloudTrail Layanan dan Integrasi yang Didukung
- Mengonfigurasi Notifikasi Amazon SNS untuk CloudTrail
- Menerima Berkas CloudTrail Log dari Beberapa Wilayah dan Menerima Berkas CloudTrail Log dari Beberapa Akun

Semua MemoryDB untuk tindakan Redis dicatat oleh CloudTrail. Misalnya, panggilan ke`CreateCluster`,`DescribeClusters` dan`UpdateCluster` tindakan menghasilkan entri dalam file CloudTrail log.

Setiap entri peristiwa atau log berisi informasi tentang siapa yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan hal berikut:

- Jika permintaan tersebut dibuat dengan kredensial pengguna root atau IAM.
- Jika permintaan tersebut dibuat dengan kredensial keamanan sementara untuk peran atau pengguna federasi.
- Bahwa permintaan dibuat oleh layanan AWS lain.

Untuk informasi lain, lihat Elemen userIdentity CloudTrail .

Memahami MemoryDB untuk entri berkas log Redis

Jejak adalah konfigurasi yang memungkinkan pengiriman peristiwa sebagai berkas log ke bucket Amazon S3 yang Anda tentukan. CloudTrail berkas log berisi satu atau lebih entri log. Sebuah peristiwa mewakili permintaan tunggal dari sumber apa pun dan mencakup informasi tentang tindakan yang diminta, tanggal dan waktu tindakan, parameter permintaan, dan sebagainya. CloudTrail Berkas log bukan jejak tumpukan panggilan API publik yang berurutan, sehingga tidak muncul dalam urutan tertentu.

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan`CreateCluster` tindakan tersebut.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EKIAUAXQT3SWDEXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/john",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "john"
    },
    "eventTime": "2021-07-10T17:56:46Z",
    "eventSource": "memorydb.amazonaws.com",
    "eventName": "CreateCluster",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.01",
    "userAgent": "aws-cli/2.2.29 Python/3.9.6 Darwin/19.6.0 source/x86_64 prompt/off
 command/memorydb.create-cluster",
    "requestParameters": {
        "clusterName": "memorydb-cluster",
        "nodeType": "db.r6g.large",
        "subnetGroupName": "memorydb-subnet-group",
        "aCLName": "open-access"
    },
    "responseElements": {
        "cluster": {
            "name": "memorydb-cluster",
            "status": "creating",
            "numberOfShards": 1,
            "availabilityMode": "MultiAZ",
```

```
            "clusterEndpoint": {
                "port": 6379
            },
            "nodeType": "db.r6g.large",
            "engineVersion": "6.2",
            "enginePatchVersion": "6.2.6",
            "parameterGroupName": "default.memorydb-redis6",
            "parameterGroupStatus": "in-sync",
            "subnetGroupName": "memorydb-subnet-group",
            "tLSEnabled": true,
            "aRN": "arn:aws:memorydb:us-east-1:123456789012:cluster/memorydb-cluster",
            "snapshotRetentionLimit": 0,
            "maintenanceWindow": "tue:06:30-tue:07:30",
            "snapshotWindow": "09:00-10:00",
            "aCLName": "open-access",
            "dataTiering": "false",
            "autoMinorVersionUpgrade": true
        }
    },
    "requestID": "506fc951-9ae2-42bb-872c-98028dc8ed11",
    "eventID": "2ecf3dc3-c931-4df0-a2b3-be90b596697e",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management"
}
```

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan`DescribeClusters` tindakan tersebut. Perhatikan bahwa untuk semua MemoryDB untuk Redis Jelaskan dan Daftar panggilan (`Describe*`dan`List*`),`responseElements` bagian dihapus dan muncul sebagai`null`.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EKIAUAXQT3SWDEXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/john",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "john"
    },
    "eventTime": "2021-07-10T18:39:51Z",
```

```
    "eventSource": "memorydb.amazonaws.com",
    "eventName": "DescribeClusters",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.01",
    "userAgent": "aws-cli/2.2.29 Python/3.9.6 Darwin/19.6.0 source/x86_64 prompt/off
 command/memorydb.describe-clusters",
    "requestParameters": {
        "maxResults": 50,
        "showShardDetails": true
    },
    "responseElements": null,
    "requestID": "5e831993-52bb-494d-9bba-338a117c2389",
    "eventID": "32a3dc0a-31c8-4218-b889-1a6310b7dd50",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management"
}
```

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan`UpdateCluster` tindakan.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EKIAUAXQT3SWDEXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/john",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "john"
    },
    "eventTime": "2021-07-10T19:23:20Z",
    "eventSource": "memorydb.amazonaws.com",
    "eventName": "UpdateCluster",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.01",
    "userAgent": "aws-cli/2.2.29 Python/3.9.6 Darwin/19.6.0 source/x86_64 prompt/off
 command/memorydb.update-cluster",
    "requestParameters": {
        "clusterName": "memorydb-cluster",
        "snapshotWindow": "04:00-05:00",
        "shardConfiguration": {
```

```
            "shardCount": 2
        }
    },
    "responseElements": {
        "cluster": {
            "name": "memorydb-cluster",
            "status": "updating",
            "numberOfShards": 2,
            "availabilityMode": "MultiAZ",
            "clusterEndpoint": {
                "address": "clustercfg.memorydb-cluster.cde8da.memorydb.us-
east-1.amazonaws.com",
                "port": 6379
            },
            "nodeType": "db.r6g.large",
            "engineVersion": "6.2",
            "EnginePatchVersion": "6.2.6",
            "parameterGroupName": "default.memorydb-redis6",
            "parameterGroupStatus": "in-sync",
            "subnetGroupName": "memorydb-subnet-group",
            "tLSEnabled": true,
            "aRN": "arn:aws:memorydb:us-east-1:123456789012:cluster/memorydb-cluster",
            "snapshotRetentionLimit": 0,
            "maintenanceWindow": "tue:06:30-tue:07:30",
            "snapshotWindow": "04:00-05:00",
            "autoMinorVersionUpgrade": true,
            "DataTiering": "false"
        }
    },
    "requestID": "dad021ce-d161-4365-8085-574133afab54",
    "eventID": "e0120f85-ab7e-4ad4-ae78-43ba15dee3d8",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management"
}
```


Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan`CreateUser` tindakan tersebut. Perhatikan bahwa untuk MemoryDB untuk panggilan Redis yang berisi data sensitif, data tersebut akan disunting dalam CloudTrail acara yang sesuai seperti yang ditunjukkan pada`requestParameters` bagian di bawah ini.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EKIAUAXQT3SWDEXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/john",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "john"
    },
    "eventTime": "2021-07-10T19:56:13Z",
    "eventSource": "memorydb.amazonaws.com",
    "eventName": "CreateUser",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.01",
    "userAgent": "aws-cli/2.2.29 Python/3.9.6 Darwin/19.6.0 source/x86_64 prompt/off
 command/memorydb.create-user",
    "requestParameters": {
        "userName": "memorydb-user",
        "authenticationMode": {
            "type": "password",
            "passwords": [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ]
        },
        "accessString": "~* &* -@all +@read"
    },
    "responseElements": {
        "user": {
            "name": "memorydb-user",
            "status": "active",
            "accessString": "off ~* &* -@all +@read",
            "aCLNames": [],
            "minimumEngineVersion": "6.2",
            "authentication": {
                "type": "password",
                "passwordCount": 1
            },
            "aRN": "arn:aws:memorydb:us-east-1:123456789012:user/memorydb-user"
        }
    },
    "requestID": "ae288b5e-80ab-4ff8-989a-5ee5c67cd193",
    "eventID": "ed096e3e-16f1-4a23-866c-0baa6ec769f6",
```

```
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management"
}
```

Validasi kepatuhan untuk MemoryDB untuk Redis

Auditor pihak ketiga menilai keamanan dan kepatuhan MemoryDB untuk Redis sebagai bagian dari beberapa program kepatuhan. AWS Hal ini mencakup:

- Standar Keamanan Data Industri Kartu Pembayaran (PCI DSS). Untuk informasi lain, lihat PCI DSS.
- Undang-Undang Akuntabilitas dan Portabilitas Asuransi Kesehatan Perjanjian Rekan Bisnis (HIPAA BAA). Untuk informasi lebih lanjut, lihat Kepatuhan HIPAA.
- Kontrol Sistem dan Organisasi (SOC) 1, 2, dan 3. Untuk informasi selengkapnya, lihat SOC.
- Program Manajemen Risiko dan Otorisasi Federal (FedRAMP) Moderat. Untuk informasi lebih lanjut, lihat FedRAMP.
- ISO/IEC 27001:2013, 27017:2015, 27018:2019, dan ISO/IEC 9001:2015. Untuk informasi selengkapnya, lihat sertifikasi dan layanan AWS ISO dan CSA STAR.

Untuk daftar layanan AWS dalam cakupan program kepatuhan spesifik, lihat Layanan AWS dalam Cakupan berdasarkan Program Kepatuhan.

Anda bisa mengunduh laporan audit pihak ketiga menggunakan AWS Artifact. Untuk informasi selengkapnya, lihat Mengunduh Laporan di AWS Artifact.

Tanggung jawab kepatuhan Anda saat menggunakan MemoryDB ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, dan hukum dan peraturan yang berlaku. AWSmenyediakan sumber daya berikut untuk membantu kepatuhan:

- Panduan Quick Start Keamanan dan Kepatuhan – Panduan deployment ini membahas pertimbangan arsitektur dan menyediakan langkah untuk deployment lingkungan dasar yang fokus pada keamanan dan kepatuhan di AWS.
- Sumber Daya Kepatuhan AWS – Kumpulan buku kerja dan panduan ini mungkin berlaku untuk industri dan lokasi Anda.

- Mengevaluasi Sumber Daya dengan Aturan dalam Panduan Developer AWS Config – AWS Config menilai seberapa patuh konfigurasi sumber daya Anda terhadap praktik internal, panduan industri, dan peraturan.
- AWS Security Hub – Layanan AWS ini memberikan pandangan yang komprehensif tentang status keamanan Anda di dalam AWS yang membantu Anda memeriksa kepatuhan terhadap standar industri dan praktik terbaik yang terkait dengan keamanan.
- AWSAudit Manager — AWS Layanan ini membantu Anda terus mengaudit AWS penggunaan Anda untuk menyederhanakan cara Anda mengelola risiko dan kepatuhan terhadap peraturan dan standar industri.

Keamanan infrastruktur di Amazon MemoryDB untuk Redis

Sebagai suatu layanan terkelola, MemoryDB dilindungi oleh prosedur keamanan jaringan AWS global yang dijelaskan dalam laporan resmi Amazon Web Services: Gambaran Umum dari Proses Keamanan.

Anda menggunakan panggilan API yang AWS dipublikasikan untuk mengakses MemoryDB 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 suite cipher 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 utama utama IAM. Atau Anda dapat menggunakan AWS Security Token Service (AWS STS) untuk membuat kredensial keamanan sementara untuk menandatangani permintaan.

Privasi lalu lintas jaringan Internet

MemoryDB untuk Redis menggunakan teknik berikut untuk mengamankan data Anda dan melindunginya dari akses yang tidak diotorisasi:

- MemoryDB dan Amazon VPC menjelaskan jenis grup keamanan yang Anda butuhkan untuk instalasi Anda.
- MemoryDB untuk API Redis dan VPC endpoint antarmuka (AWS PrivateLink)memungkinkan Anda untuk membangun koneksi privat antara VPC Anda dan MemoryDB untuk Redis API endpoint.

- Manajemen identitas dan akses di MemoryDB untuk Redis untuk memberikan dan membatasi tindakan pengguna, grup, dan peran.

MemoryDB dan Amazon VPC

Layanan Amazon Virtual Private Cloud (Amazon VPC) mendefinisikan jaringan virtual yang mirip dengan pusat data tradisional. Anda dapat membuat konfigurasi virtual private cloud (VPC) dengan Amazon VPC, Anda dapat memilih baris alamat IP, membuat subnet, dan mengonfigurasi tabel rute, gateway jaringan, dan pengaturan keamanan. Anda juga dapat menambahkan sebuah klaster ke jaringan virtual, dan mengontrol akses ke klaster dengan menggunakan kelompok-kelompok keamanan Amazon VPC.

Bagian ini menjelaskan cara mengonfigurasi klaster MemoryDB di VPC secara manual. Informasi ini ditujukan untuk pengguna yang menginginkan pemahaman yang lebih dalam tentang cara MemoryDB dan Amazon VPC bekerja bersama.

Topik

- Memahami MemoryDB dan VPC
- Pola Akses untuk Mengakses Cluster MemoryDB di Amazon VPC
- Membuat Virtual Private Cloud (VPC)

Memahami MemoryDB dan VPC

MemoryDB sepenuhnya terintegrasi dengan Amazon VPC. Untuk pengguna MemoryDB, hal ini berarti sebagai berikut:

- MemoryDB selalu meluncurkan klaster Anda di VPC.
- Jika Anda baruAWS, VPC default akan dibuat untuk Anda secara otomatis.
- Jika Anda memiliki VPC default dan tidak menentukan subnet saat meluncurkan sebuah klaster, maka klaster itu akan diluncurkan ke Amazon VPC default Anda.

Untuk informasi lain, lihat Mendeteksi Platform Anda yang Didukung dan Apakah Anda Memiliki VPC Default.

Dengan Amazon VPC, Anda dapat membuat jaringan virtual diAWSCloud yang sangat menyerupai pusat data tradisional. Anda dapat mengonfigurasi VPC Anda, termasuk memilih baris alamat IP, membuat subnet, dan mengonfigurasi tabel rute, gateway jaringan, dan pengaturan keamanan.

MemoryDB mengelola peningkatan perangkat lunak, patching, deteksi kegagalan, dan pemulihan perangkat lunak.

Sekilas MemoryDB di VPC

1	VPC adalah bagian terisolasi dariAWSCloud yang diberikan blok alamat IP sendiri.
2	Gateway internet menghubungkan VPC Anda langsung ke internet dan menyediakan akses ke lainnyaAWSsumber daya seperti Amazon Simple Storage Service (Amazon S3) yang berjalan di luar VPC Anda.
3	Subnet Amazon VPC adalah segmen rentang alamat IP VPC tempat Anda dapat mengisolasiAWSsumber daya sesuai dengan keamanan dan kebutuhan operasional Anda.
4	Tabel perutean di VPC Anda mengarahkan lalu lintas jaringan antara subnet dan internet. Amazon VPC memiliki router tersirat.
5	

	Grup keamanan Amazon VPC mengontrol lalu lintas masuk dan keluar untuk klaster MemoryDB dan instans Amazon EC2.
6	Anda dapat meluncurkan klaster MemoryDB di subnet. Simpul memiliki alamat IP privat dari rentang alamat subnet.
7	Anda juga dapat meluncurkan instans Amazon EC2 di subnet. Setiap instans Amazon EC2 memiliki alamat IP privat dari rentang alamat subnet. Instans Amazon EC2 dapat menyambung ke simpul apa pun di subnet yang sama.
8	Agar instans Amazon EC2 di VPC Anda dapat dicapai dari internet, Anda perlu menetapkan alamat IP publik dan statis yang disebut alamat IP Elastis untuk instans tersebut.

Prasyarat

Untuk membuat klaster MemoryDB dalam VPC, VPC Anda harus memenuhi persyaratan berikut:

- VPC Anda harus mengizinkan instans Amazon EC2 nondedicated. Anda tidak dapat menggunakan MemoryDB di VPC yang mempunyai konfigurasi untuk penghuni instans khusus.
- Grup subnet harus ditentukan untuk VPC Anda. MemoryDB menggunakan grup subnet untuk memilih subnet dan alamat IP dalam subnet tersebut untuk mengasosiasikan dengan simpul Anda.
- Grup keamanan harus didefinisikan untuk VPC Anda, atau Anda dapat menggunakan grup default yang disediakan.
- Blok CIDR untuk setiap subnet harus cukup besar untuk menyediakan alamat IP cadangan untuk MemoryDB untuk digunakan selama kegiatan pemeliharaan.

Perutean dan keamanan

Anda dapat membuat konfigurasi perutean di VPC Anda untuk mengontrol aliran lalu lintas (misalnya, ke gateway internet atau gateway privat virtual). Dengan gateway internet, VPC Anda memiliki akses langsung ke yang lainAWSsumber daya yang tidak berjalan di 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 dan menggunakan kebijakan keamanan lokal dan firewall untuk mengontrol egress. Dalam kasus ini, Anda dikenakan biaya bandwidth tambahan saat Anda mengaksesAWSsumber daya melalui internet.

Anda dapat menggunakan kelompok-kelompok keamanan Amazon VPC untuk membantu mengamankan klaster MemoryDB 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 Anda, karena alamat IP yang mendasari dapat berubah seiring waktu.

Dokumentasi Amazon VPC

Amazon VPC memiliki kumpulan dokumentasinya sendiri untuk menjelaskan cara membuat dan menggunakan Amazon VPC Anda. Tabel berikut menunjukkan tempat menemukan informasi dalam panduan Amazon VPC.

Deskripsi	Dokumentasi
Cara memulai menggunakan Amazon VPC	Memulai dengan Amazon VPC
Cara menggunakan Amazon VPC melaluiAWS Management Console	Panduan Pengguna Amazon VPC
Deskripsi lengkap dari semua perintah Amazon VPC	Referensi Baris Perintah Amazon EC2(perintah Amazon VPC ditemukan di referensi Amazon EC2)
Deskripsi lengkap dari operasi API Amazon VPC, tipe data, dan kesalahan	Referensi API Amazon EC2(operasi Amazon VPC API ditemukan di referensi Amazon EC2)
Informasi untuk administrator jaringan yang perlu membuat konfigurasi gateway di ujung koneksi VPN IPsec opsional Anda	ApaAWSSite-to-Site VPN?

Untuk informasi selengkapnya tentang Amazon Virtual Private Cloud, lihatAmazon Virtual Private Cloud.

Pola Akses untuk Mengakses Cluster MemoryDB di Amazon VPC

MemoryDB untuk Redis mendukung skenario berikut untuk mengakses klaster di Amazon VPC:

Daftar Isi

Mengakses Klaster MemoryDB jika Klaster MemoryDB jika Klaster dan Instans Amazon EC2 ada di Amazon VPC

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 dan klaster EC2 di dalam VPC yang sama adalah dengan melakukan hal berikut:

1. Buat grup keamanan VPC untuk klaster Anda. Grup keamanan ini dapat digunakan untuk membatasi akses ke 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 MemoryDB adalah`6379`.

2. Buat grup keamanan VPC untuk instans EC2 Anda (server web dan aplikasi). Grup keamanan ini dapat, jika diperlukan, mengizinkan akses ke instans EC2 dari Internet melalui 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 khusus dalam grup keamanan untuk klaster Anda yang memungkinkan koneksi dari grup keamanan yang Anda buat untuk instans EC2. Hal ini akan mengizinkan semua anggota grup keamanan untuk mengakses klaster.

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

1. Login ke Konsol Manajemen AWS dan buka konsol Amazon VPC di https://console.aws.amazon.com/vpc.
2. Di panel navigasi sebelah kiri, pilih Grup Keamanan.
3. Pilih atau buat grup keamanan yang akan Anda gunakan untuk klaster Anda. Di bawah Aturan Masuk, pilih Edit Aturan Masuk dan kemudian pilih Tambahkan Aturan. Grup keamanan ini akan mengizinkan akses untuk anggota dari grup keamanan lain.
4. Dari Jenis, pilih Aturan TCP Khusus.
   a. Untuk Rentang Port, tentukan port yang digunakan saat Anda membuat klaster Anda.

      Port default untuk klaster MemoryDB adalah`6379`.
   b. Pada kotak Sumber, mulai ketikkan ID dari grup keamanan. Dari daftar, pilih grup keamanan yang akan Anda gunakan untuk instans Amazon EC2.
5. Pilih Simpan saat Anda selesai.

Mengakses Cluster MemoryDB saat dan Instans Amazon EC2 berada di VPC Amazon yang Berbeda

Jika klaster Anda ada dalam VPC yang berbeda dengan instans EC2 yang Anda gunakan untuk mengaksesnya, ada beberapa cara untuk mengakses klaster. Jika klaster dan instans EC2 berada di VPC yang berbeda tetapi di wilayah yang sama, Anda dapat menggunakan peering VPC. Jika klaster dan instans EC2 berada di wilayah yang berbeda, Anda dapat menciptakan konektivitas VPN di antara wilayah.

Topik

- Mengakses Klaster MemoryDB jika Klaster MemoryDB jika Klaster dan Instans Amazon EC2 ada di Wilayah yang Sama
- Mengakses Klaster MemoryDB jika Klaster MemoryDB jika Klaster dan Instans Amazon EC2 Berada di Wilayah yang Berbeda

Mengakses Klaster MemoryDB jika Klaster MemoryDB jika Klaster dan Instans Amazon EC2 ada di Wilayah yang Sama

Klaster yang diakses oleh instans Amazon EC2 dalam 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 pada kedua VPC dapat berkomunikasi satu sama lain seolah-olah mereka ada di jaringan yang sama. Anda dapat membuat koneksi peering VPC di antara VPC milik Anda sendiri, atau dengan Amazon VPC di VPC lainAWSakun dalam satu wilayah. Untuk mempelajari selengkapnya tentang peering Amazon VPC, lihatDokumentasi VPC.

Mengakses klaster di Amazon VPC yang berbeda melalui peering

1. Pastikan bahwa kedua VPC tidak memiliki rentang IP yang tumpang tindih atau Anda tidak akan dapat membuat koneksi peering.
2. Buat koneksi peering di antara kedua VPC. Untuk informasi selengkapnya, lihatMembuat dan Menerima Koneksi Peering VPC Amazon.
3. Perbarui tabel perutean Anda. Untuk informasi lain, lihat.Memperbarui Tabel Rute Anda untuk Koneksi Peering VPC
4. Ubah Grup Keamanan dari klaster MemoryDB Anda agar mengizinkan koneksi masuk dari grup keamanan Aplikasi di dalam VPC yang tersambung peering. Untuk informasi lain, lihat Grup Keamanan VPC Peer Referensi.

Mengakses klaster melalui koneksi peering akan dikenakan biaya transfer data tambahan.

Menggunakan Transit Gateway

Transit gateway memungkinkan Anda untuk menyematkan attachment koneksi VPC dan VPN di Wilayah AWS yang sama dan mengarahkan rute lalu lintas di antara keduanya. Transit gateway bekerja di seluruh akun AWS, dan Anda dapat menggunakan AWSResource Access Manager untuk berbagi transit gateway Anda dengan akun lain. Setelah Anda berbagi transit gateway dengan akun AWS lain, pemilik akun dapat menyematkan attachment VPC mereka ke transit gateway Anda. Pengguna dari kedua akun dapat menghapus attachment ini kapan saja.

Anda dapat mengaktifkan multicast pada transit gateway, dan kemudian membuat domain multicast transit gateway yang mengizinkan lalu lintas multicast dikirim dari sumber multicast Anda untuk anggota grup multicast melalui attachment VPC yang Anda kaitkan dengan domain.

Anda juga dapat membuat attachment koneksi peering antara transit gateway di berbagai Wilayah AWS yang berbeda. Hal ini memungkinkan Anda untuk mengatur rute lalu lintas antara beberapa transit gateway attachment di Wilayah yang berbeda.

Untuk informasi lain, lihat Transit gateway.

Mengakses Klaster MemoryDB jika Klaster MemoryDB jika Klaster dan Instans Amazon EC2 Berada di Wilayah yang Berbeda

Menggunakan Transit VPC

Alternatif dari menggunakan peering VPC, strategi umum lain untuk menghubungkan beberapa VPC dan jaringan jarak jauh yang terpisah 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. Desain ini dapat menghemat waktu dan tenaga dan juga mengurangi biaya, karena diimplementasikan secara virtual tanpa biaya tradisional untuk membangun kehadiran fisik di hub transit kolokasi atau men-deploy peralatan jaringan fisik.

Menghubungkan seluruh VPC yang berbeda di berbagai wilayah

Setelah Transit Amazon VPC dibuat, aplikasi yang di-deploy di sebuah “kisi” VPC di satu wilayah dapat terhubung ke klaster MemoryDB di “kisi” VPC di wilayah lain.

Untuk mengakses klaster di VPC yang berbeda dalam Wilayah AWS yang berbeda

1. Deploy Solusi Transit VPC. Untuk informasi lain, lihat Transit Gateway AWS.

2. Perbarui tabel perutean VPC di dalam Aplikasi dan VPC untuk mengarahkan lalu lintas melalui VGW (Gateway Privat Virtual) dan Perangkat VPN. Dalam kasus Perutean Dinamis dengan Border Gateway Protocol (BGP), rute Anda dapat secara otomatis disebarkan.
3. Ubah Grup Keamanan dari klaster MemoryDB 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 klaster di seluruh wilayah akan memperkenalkan latensi jaringan dan biaya transfer data lintas wilayah tambahan.

Mengakses Klaster MemoryDB dari Aplikasi yang Berjalan di Pusat Data Pelanggan

Skenario lain yang mungkin adalah arsitektur Hybrid di mana klien atau aplikasi di pusat data pelanggan mungkin perlu mengakses Klaster MemoryDB di VPC. Skenario ini juga didukung asalkan ada konektivitas antara VPC pelanggan dan pusat data baik melalui VPN atau Direct Connect.

Topik

- Mengakses Klaster MemoryDB dari Aplikasi yang Berjalan di Pusat Data Pelanggan Menggunakan Konektivitas VPN
- Mengakses Klaster MemoryDB dari Aplikasi yang Berjalan di Pusat Data Pelanggan Menggunakan Direct Connect Direct Connect

Mengakses Klaster MemoryDB dari Aplikasi yang Berjalan di Pusat Data Pelanggan Menggunakan Konektivitas VPN

Menyambung ke MemoryDB dari pusat data Anda melalui VPN

Untuk mengakses klaster di VPC dari aplikasi on-prem melalui koneksi VPN

1. Membuat Konektivitas VPN dengan menambahkan perangkat keras Gateway Privat Virtual ke VPC Anda. Untuk informasi lain, lihat Menambahkan Perangkat Keras Gateway Privat Virtual ke VPC Anda.
2. Perbarui tabel perutean VPC untuk subnet di mana klaster MemoryDB Anda di-deploy untuk memungkinkan lalu lintas dari server aplikasi on-premise Anda. Dalam kasus Perutean Dinamis dengan BGP, rute Anda dapat secara otomatis disebarkan.

3. Ubah Grup Keamanan dari klaster MemoryDB Anda agar mengizinkan koneksi masuk dari server aplikasi on-premise.

Mengakses klaster melalui koneksi VPN akan menimbulkan latensi jaringan dan biaya transfer data tambahan.

Mengakses Klaster MemoryDB dari Aplikasi yang Berjalan di Pusat Data Pelanggan Menggunakan Direct Connect Direct Connect

Menyambung ke MemoryDB dari pusat data Anda melalui Direct

Untuk mengakses klaster MemoryDB dari aplikasi yang berjalan di jaringan Anda menggunakan Direct Connect

1. Buat konektivitas Direct Connect. Untuk informasi lain, lihat Memulai dengan AWS Direct Connect.
2. Ubah Grup Keamanan dari klaster MemoryDB Anda agar mengizinkan koneksi masuk dari server aplikasi on-premise.

Mengakses klaster melalui koneksi DX dapat menimbulkan latensi jaringan dan biaya transfer data tambahan.

Membuat Virtual Private Cloud (VPC)

Dalam contoh ini, Anda membuat virtual private cloud (VPC) berdasarkan layanan Amazon VPC dengan subnet privat untuk setiap Availability Zone.

Membuat VPC (Console)

Untuk membuat klaster MemoryDB di dalam Amazon Virtual Private Cloud

1. Masuk keAWSManagement Console, dan buka konsol Amazon VPC dihttps://console.aws.amazon.com/vpc/.
2. Di dasbor VPC, pilihBuat VPC.
3. Di bawahSumber dayauntuk membuat, memilihVPC dan banyak lagi.
4. Di bawahJumlah Availability Zone (AZ), pilih jumlah Availability Zone yang ingin Anda luncurkan subnet Anda.
5. Di bawahJumlah subnet, pilih jumlah subnet publik yang ingin Anda tambahkan ke VPC Anda.
6. Di bawahJumlah subnet, pilih jumlah subnet pribadi yang ingin Anda tambahkan ke VPC Anda.

   > **Tip**
   >
   > Perhatikan pengidentifikasi subnet Anda, dan 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 dan instans Amazon EC2 Anda.

   a. Di panel navigasi kiriAWS Management Console, pilihKelompok Keamanan.
   b. Pilih Create Security Group (Buat Grup Keamanan).
   c. Masukkan nama dan deskripsi untuk grup keamanan Anda di kotak yang sesuai. UntukVPC, pilih pengenal untuk VPC Anda.
   d. Bila pengaturan sudah sesuai keinginan Anda, pilih Ya, Buat.

8. Tentukan aturan ingress jaringan untuk grup keamanan Anda. Aturan ini akan memungkinkan Anda terhubung ke instans Amazon EC2 Anda menggunakan Secure Shell (SSH).

   a. Di panel navigasi sebelah kiri, pilih Grup Keamanan.
   b. Temukan grup keamanan Anda di dalam daftar, dan kemudian pilih grup itu.

c. Di bawah Grup Keamanan, pilih tab Masuk. Pada kotak Buat aturan baru, pilih SSH, lalu pilih Tambahkan Aturan.

   Tetapkan nilai berikut untuk aturan masuk baru Anda untuk memungkinkan akses HTTP:

   - Jenis: HTTP
   - Sumber: 0.0.0.0/0

d. Tetapkan nilai berikut untuk aturan masuk baru Anda untuk memungkinkan akses HTTP:

   - Jenis: HTTP
   - Sumber: 0.0.0.0/0

   Pilih Terapkan Perubahan Aturan.

Sekarang Anda siap untuk membuatgrupdanmembuat sebuah klasterdi VPC Anda.

Subnet dan grup subnet

Grup subnet adalah sekumpulan subnet (biasanya private) yang dapat Anda tetapkan untuk klaster Anda yang berjalan di lingkungan Amazon Virtual Private Cloud (VPC).

Saat Anda membuat klaster di Amazon VPC, Anda dapat menentukan grup subnet atau menggunakan grup subnet atau menggunakan grup subnet yang disediakan. MemoryDB menggunakan grup subnet tersebut untuk memilih subnet dan alamat IP dalam subnet tersebut untuk mengasosiasikan dengan simpul Anda.

Bagian ini mencakup cara membuat dan memanfaatkan subnet dan grup subnet untuk mengelola akses ke sumber daya MemoryDB Anda.

Untuk informasi lain tentang penggunaan grup subnet di lingkungan Amazon VPC, lihatLangkah 2: Otorisasi akses ke cluster.

ID MemoryDB AZ yang didukung

Nama Wilayah/Wilayah	ID AZ yang didukung		
Wilayah US East (Ohio) `us-east-2`	`use2-az1, use2-az2, use2-az3`		
Wilayah US East (N. Virginia) `us-east-1`	`use1-az2, use1-az4, use1-az6`		
Wilayah US West (N. California) `us-west-1`	`usw1-az1, usw1-az2, usw1-az3`		
Wilayah US West (Oregon) `us-west-2`	`usw2-az1, usw2-az2, usw2-az3`		

Nama Wilayah/Wilayah	ID AZ yang didukung		
Wilayah Canada (Central) `ca-central-1`	`cac1-az1, cac1-az2, cac1-az4`		
Wilayah Asia Pacific (Hong Kong) `ap-east-1`	`ape1-az1, ape1-az2, ape1-az3`		
Wilayah Asia Pacific (Mumbai) `ap-south-1`	`aps1-az1, aps1-az2, aps1-az3`		
Wilayah Asia Pacific (Tokyo) `ap-northeast-1`	`apne1-az1, apne1-az2, apne1-az4`		
Wilayah Asia Pasifik (Seoul) `ap-northeast-2`	`apne2-az1, apne2-az2, apne2-az3`		
Wilayah Asia Pacific (Singapore) `ap-southeast-1`	`apse1-az1, apse1-az2, apse1-az3`		
Wilayah Asia Pasifik (Sydney) `ap-southeast-2`	`apse2-az1, apse2-az2, apse2-az3`		

Nama Wilayah/Wilayah	ID AZ yang didukung		
Wilayah Europe (Frankfurt) `eu-central-1`	`euc1-az1, euc1-az2, euc1-az3`		
Wilayah Europe (Ireland) `eu-west-1`	`euw1-az1, euw1-az2, euw1-az3`		
Wilayah Europe (London) `eu-west-2`	`euw2-az1, euw2-az2, euw2-az3`		
Wilayah Europe (Stockholm) `eu-north-1`	`eun1-az1, eun1-az2, eun1-az3`		
Wilayah South America (Sao Paulo) `sa-east-1`	sae1-az2, sae1-az2, sae1-az3		
Wilayah China (Beijing) `cn-north-1`	cnn1-az1, cnn1-az2		
Wilayah China (Ningxia) `cn-northwest-1`	CNW1-az1, CNW1-az2, CNW1-az3		

Topik

- Membuat Grup Subnet

- Memperbarui grup subnet
- Melihat detail grup subnet
- Menghapus Grup Subnet

Membuat Grup Subnet

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 lebih subnet ke grup subnet sehingga Anda memiliki jumlah alamat IP yang cukup di dalam Availability Zone dari klaster Anda. Setelah itu, Anda dapat menambahkan lebih banyak simpul ke klaster Anda.

Prosedur berikut menunjukkan cara membuat grup subnet bernama`mysubnetgroup` (konsol)AWS CLI, dan API MemoryDB.

Membuat grup subnet (konsol)

Prosedur berikut menunjukkan cara membuat grup subnet (konsol).

Untuk membuat grup subnet (konsol)

1. Masuk ke konsolAWS manajemen dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/.
2. Di panel navigasi bagian kiri, pilih Grup Subnet.
3. Pilih Buat Grup Subnet.
4. Di Buat grup subnet, lakukan langkah berikut:
   a. Pada kotak Nama, ketik nama grup subnet Anda.

      Kendala penamaan klaster adalah sebagai berikut:

      - Harus mengandung dari 1—40 karakter alfanumerik atau tanda hubung.
      - Harus dimulai dengan huruf.
      - Tidak dapat mengandung dua tanda hubung berturut-turut.
      - Tidak dapat diakhiri dengan tanda hubung.

   b. Pada kotak Deskripsi, ketik deskripsi untuk grup subnet Anda.
   c. Pada kotak ID VPC, pilih Amazon VPC yang sudah dibuat. Jika Anda belum membuatnya, pilih tombol Buat VPC dan ikuti langkah-langkah untuk membuatnya.
   d. Di subnet yang dipilih, pilih Availability Zone dan ID subnet privat Anda, lalu pilih Pilih.
5. Untuk Tag, Anda dapat menerapkan tag secara opsional untuk mencari dan memfilter subnet atau melacakAWS biaya Anda.

6. Saat semua pengaturan sesuai keinginan Anda, pilih Buat.
7. Pada pesan konfirmasi yang muncul, pilih Tutup.

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

Membuat grup subnet (AWSCLI)

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

Untuk Linux, macOS, atau Unix:

```
aws memorydb create-subnet-group \
    --subnet-group-name mysubnetgroup \
    --description "Testing" \
    --subnet-ids subnet-53df9c3a
```

Untuk Windows:

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

Perintah ini seharusnya menghasilkan keluaran yang serupa dengan berikut ini:

```
    {
        "SubnetGroup": {
            "Subnets": [
                {
                    "Identifier": "subnet-53df9c3a",
                    "AvailabilityZone": {
                    "Name": "us-east-1a"
                    }
                }
            ],
            "VpcId": "vpc-3cfaef47",
            "Name": "mysubnetgroup",
            "ARN": "arn:aws:memorydb:us-east-1:012345678912:subnetgroup/
mysubnetgroup",
```

```
            "Description": "Testing"
        }
    }
```

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

Membuat grup subnet (API MemoryDB)

Menggunakan API MemoryDB, panggil`CreateSubnetGroup` dengan parameter berikut ini:

- `SubnetGroupName=`*`mysubnetgroup`*
- `Description=`*`Testing`*
- `SubnetIds.member.1=`*`subnet-53df9c3a`*

Memperbarui grup subnet

Anda dapat memperbarui deskripsi grup subnet, atau mengubah daftar ID subnet yang terhubung dengan grup subnet. Anda tidak dapat menghapus ID subnet dari grup subnet jika klaster saat ini menggunakan subnet tersebut.

Prosedur berikut menunjukkan cara memperbarui grup subnet.

Memperbarui grup subnet (Konsol)

Untuk memperbarui grup subnet

1. Masuk keAWS Management Console dan buka konsol MemoryDB untuk Redis di https://console.aws.amazon.com/memorydb/.
2. Di panel navigasi bagian kiri, pilih Grup Subnet.
3. Pada daftar grup subnet, pilih grup subnet yang ingin Anda ubah.
4. Bidang Nama, VPCid, dan Deskripsi tidak dapat dimodifikasi.
5. Di bagian subnet yang dipilih, klik Kelola untuk membuat perubahan pada Availability Zone yang Anda butuhkan untuk subnet. Untuk menyimpan perubahan Anda, pilih Simpan.

Memperbarui grup subnet (AWSCLI)

Pada baris perintah, gunakan perintah`update-subnet-group` untuk memperbarui grup subnet.

Untuk Linux, macOS, atau Unix:

```
aws memorydb update-subnet-group \
    --subnet-group-name mysubnetgroup \
    --description "New description" \
    --subnet-ids "subnet-42df9c3a" "subnet-48fc21a9"
```

Untuk Windows:

```
aws memorydb update-subnet-group ^
    --subnet-group-name mysubnetgroup ^
    --description "New description" ^
    --subnet-ids "subnet-42df9c3a" "subnet-48fc21a9"
```

Perintah ini seharusnya menghasilkan keluaran yang serupa dengan berikut ini:

```
{
    "SubnetGroup": {
        "VpcId": "vpc-73cd3c17",
        "Description": "New description",
        "Subnets": [
            {
                "Identifier": "subnet-42dcf93a",
                "AvailabilityZone": {
                    "Name": "us-east-1a"
                }
            },
            {
                "Identifier": "subnet-48fc12a9",
                "AvailabilityZone": {
                    "Name": "us-east-1a"
                }
            }
        ],
        "Name": "mysubnetgroup",
        "ARN": "arn:aws:memorydb:us-east-1:012345678912:subnetgroup/mysubnetgroup",
    }
}
```

Untuk informasi lain, lihat topik AWS CLI update-subnet-group.

Memperbarui kelompok subnet (MemoryDB API)

Menggunakan API MemoryDB, panggil`UpdateSubnetGroup` dengan parameter berikut ini:

- `SubnetGroupName=`*`mysubnetgroup`*
- Parameter lain apa pun yang nilainya ingin Anda ubah. Contoh ini menggunakan `Description=`*`New%20description`* untuk mengubah deskripsi grup subnet.

Example

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=UpdateSubnetGroup
    &Description=New%20description
    &SubnetGroupName=mysubnetgroup
    &SubnetIds.member.1=subnet-42df9c3a
    &SubnetIds.member.2=subnet-48fc21a9
```

```
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Timestamp=20141201T220302Z
    &Version=2014-12-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Credential=<credential>
    &X-Amz-Date=20141201T220302Z
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Signature=<signature>
    &X-Amz-SignedHeaders=Host
```

> **Note**
>
> 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 lebih subnet ke grup subnet sehingga Anda memiliki jumlah alamat IP yang cukup di dalam Availability Zone dari klaster Anda. Setelah itu, Anda dapat menambahkan lebih banyak simpul ke klaster Anda.

Melihat detail grup subnet

Prosedur berikut menunjukkan cara melihat detail grup subnet.

Melihat rincian kelompok subnet (konsol)

Untuk melihat detail grup subnet (konsol)

1. Masuk keAWS Management Console dan buka konsol MemoryDB untuk Redis di https://console.aws.amazon.com/memorydb/.
2. Di panel navigasi bagian kiri, pilih Grup Subnet.
3. Pada halaman Grup subnet, pilih grup subnet di bawah Nama atau masukkan nama grup subnet di bilah pencarian.
4. Pada halaman Grup subnet, pilih grup subnet di bawah Nama atau masukkan nama grup subnet di bilah pencarian.
5. Di bawah pengaturan grup Subnet, Anda dapat melihat nama, deskripsi, ID VPC, dan Amazon Resource Name (ARN) grup subnet.

6. Di bawah Subnet, Anda dapat melihat Availability Zones, Subnet ID, dan blok CIDR dari grup subnet
7. Di bawah Tag, Anda dapat melihat tag apa pun yang terkait dengan grup subnet.

Melihat rincian kelompok subnet (AWSCLI)

Pada baris perintah, gunakan perintah`describe-subnet-groups` untuk melihat detail grup subnet yang ditentukan.

Untuk Linux, macOS, atau Unix:

```
aws memorydb describe-subnet-groups \
    --subnet-group-name mysubnetgroup
```

Untuk Windows:

```
aws memorydb describe-subnet-groups ^
    --subnet-group-name mysubnetgroup
```

Perintah ini seharusnya menghasilkan keluaran yang serupa dengan berikut ini:

```
{
  "subnetgroups": [
    {
      "Subnets": [
        {
          "Identifier": "subnet-060cae3464095de6e",
          "AvailabilityZone": {
            "Name": "us-east-1a"
          }
        },
        {
          "Identifier": "subnet-049d11d4aa78700c3",
          "AvailabilityZone": {
            "Name": "us-east-1c"
          }
        },
        {
          "Identifier": "subnet-0389d4c4157c1edb4",
          "AvailabilityZone": {
            "Name": "us-east-1d"
```

```
          }
        }
      ],
      "VpcId": "vpc-036a8150d4300bcf2",
      "Name": "mysubnetgroup",
      "ARN": "arn:aws:memorydb:us-east-1:53791xzzz7620:subnetgroup/mysubnetgroup",
      "Description": "test"
    }
  ]
}
```

Untuk melihat detail pada semua grup subnet, gunakan perintah yang sama tetapi tanpa menentukan nama grup subnet.

```
aws memorydb describe-subnet-groups
```

Untuk informasi lain, lihat topik AWS CLI describe-subnet-groups.

Melihat kelompok subnet (MemoryDB API)

Menggunakan API MemoryDB, panggil`DescribeSubnetGroups` dengan parameter berikut ini:

`SubnetGroupName=`*`mysubnetgroup`*

Example

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=UpdateSubnetGroup
    &Description=New%20description
    &SubnetGroupName=mysubnetgroup
    &SubnetIds.member.1=subnet-42df9c3a
    &SubnetIds.member.2=subnet-48fc21a9
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Timestamp=20211801T220302Z
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Credential=<credential>
    &X-Amz-Date=20210801T220302Z
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Signature=<signature>
```

```
    &X-Amz-SignedHeaders=Host
```

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 sebuah klaster. Anda juga tidak dapat menghapus grup subnet pada klaster dengan pengaktifan Multi-AZ jika melakukannya meninggalkan klaster dengan kurang dari dua subnet. Anda harus menghapus centang Multi-AZ terlebih dahulu dan kemudian menghapus subnet.

Prosedur berikut menunjukkan cara menghapus grup subnet.

Menghapus grup subnet (konsol)

Untuk menghapus grup subnet

1. Masuk keAWS Management Console dan buka konsol MemoryDB untuk Redis di https://console.aws.amazon.com/memorydb/.
2. Di panel navigasi bagian kiri, pilih Grup Subnet.
3. Pada daftar grup subnet, pilih grup subnet yang ingin dihapus, pilih Tindakan dan kemudian pilih Hapus.

   > **Note**
   >
   > Anda tidak dapat menghapus grup subnet default atau grup subnet yang terhubung dengan klaster apa pun.

4. Layar konfirmasi Hapus Grup Subnet akan muncul.
5. Untuk menghapus grup subnet, masukkan`delete` di kotak teks konfirmasi. Untuk menyinmpan grup subnet, pilih Batal.

Menghapus grup subnet (AWSCLI)

Menggunakan AWS CLI, panggil perintah delete-subnet-group dengan parameter berikut:

- `--subnet-group-name`*`mysubnetgroup`*

Untuk Linux, macOS, atau Unix:

```
aws memorydb delete-subnet-group \
```

```
    --subnet-group-name mysubnetgroup
```

Untuk Windows:

```
aws memorydb delete-subnet-group ^
    --subnet-group-name mysubnetgroup
```

Untuk informasi lain, lihat topik AWS CLI delete-subnet-group.

Menghapus grup subnet (API MemoryDB)

Menggunakan API MemoryDB, panggil`DeleteSubnetGroup` dengan parameter berikut ini:

- `SubnetGroupName=mysubnetgroup`

Example

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=DeleteSubnetGroup
    &SubnetGroupName=mysubnetgroup
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Timestamp=20210801T220302Z
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Credential=<credential>
    &X-Amz-Date=20210801T220302Z
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Signature=<signature>
    &X-Amz-SignedHeaders=Host
```

Perintah ini tidak menghasilkan output.

Untuk informasi lebih lanjut, lihat topik API MemoryDB DeleteSubnetGroup.

MemoryDB untuk API Redis dan VPC endpoint antarmuka (AWS PrivateLink)

Anda dapat membuat koneksi privat antara VPC Anda danAmazon MemoryDB untuk titik akhir Redis API dengan membuatVPC endpoint antarmuka. Endpoint Antarmukadidukung oleh AWS

PrivateLink.AWS PrivateLink memungkinkanAnda untuk mengakses MemoryDB untuk operasi API Redis tanpa Gateway internet, perangkat NAT, koneksi VPN, atauAWSKoneksi Direct Connect.

Instans di VPC Anda tidak memerlukan alamat IP publik untuk berkomunikasi dengan MemoryDB untuk titik akhir API Redis. Instans Anda juga tidak memerlukan alamat IP publik untuk menggunakan salah satu operasi API MemoryDB yang tersedia. Lalu lintas antara VPC Anda dan MemoryDB untuk Redis tidak meninggalkan jaringan Amazon. Setiap titik akhir antarmuka diwakili oleh satu atau lebih antarmuka jaringan elastis di subjaringan Anda. Untuk informasi lebih lanjut tentang antarmuka jaringan elastis, lihat Antarmuka jaringan elastis dalam Panduan Pengguna Amazon EC2.

- Untuk informasi lebih lanjut tentang VPC endpoint, lihatTitik Akhir VPC antarmuka (AWS PrivateLink)diPanduan Pengguna Amazon VPC.
- Untuk informasi lebih lanjut tentang operasi API MemoryDB, lihatOperasi API MemoryDB.

Setelah Anda membuat antarmuka VPC endpoint, jika Anda mengaktifkanDNS pribadihostnames untuk endpoint, endpoint memoryDB default (https://memorydb.*Wilayah*.amazonaws.com) menyelesaikan VPC endpoint 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.memorydb.Region.vpce.amazonaws.com
```

Untuk informasi selengkapnya, lihatTitik Akhir VPC Antarmuka (AWS PrivateLink)diPanduan Pengguna Amazon VPC. MemoryDB mendukung panggilan ke semua nyaTindakan APIdi dalam VPC Anda.

> **Note**
>
> Nama host DNS pribadi dapat diaktifkan hanya untuk satu titik akhir VPC di VPC. Jika Anda ingin membuat endpoint VPC tambahan maka hostname DNS pribadi harus dinonaktifkan untuk itu.

Pertimbangan untuk VPC endpoint

Sebelum Anda menyiapkan VPC endpoint antarmuka untuk MemoryDB untuk Redis API endpoint, pastikan bahwa Anda meninjauProperti endpoint antarmuka dan keterbatasandiPanduan Pengguna Amazon VPC. Semua operasi API MemoryDByangrelevan untuk mengelola MemoryDB untuk sumber daya Redis tersedia dari VPC Anda menggunakanAWS PrivateLink. Kebijakan VPC endpoint

didukung untuk titik akhir API MemoryDB. Secara default, akses penuh ke operasi API MemoryDB diizinkan melalui endpoint. Untuk informasi lebih lanjut, lihat Mengendalikan akses ke layanan dengan VPC endpoint di Panduan Pengguna Amazon VPC.

Membuat VPC endpoint antarmuka untuksangAPI MemoryDB

Anda dapat membuat VPC endpoint untuk API MemoryDB untuk Redis menggunakan konsol Amazon VPC atauAWS CLI. Untuk informasi lebih lanjut, lihat Membuat titik akhir antarmuka di Panduan Pengguna Amazon VPC.

Setelah Anda membuat VPC endpoint antarmuka, Anda dapat mengaktifkan nama host DNS privat untuk endpoint. Ketika Anda melakukannya, default MemoryDB untuk Redis endpoint (https://memorydb.*Wilayah*.amazonaws.com) menyelesaikan VPC endpoint Anda. Untuk informasi lebih lanjut, lihat Mengakses layanan melalui titik akhir antarmuka di Panduan Pengguna Amazon VPC.

Membuat kebijakan VPC endpoint untuksangAPI Amazon MemoryDB

Anda dapat melampirkan kebijakan titik akhir ke VPC endpoint Anda yang mengontrol akses ke API MemoryDB. Kebijakan menentukan hal-hal berikut:

- Principal yang dapat melakukan tindakan.
- Tindakan yang dapat dilakukan.
- Sumber daya yang dapat digunakan untuk mengambil tindakan.

Untuk informasi lebih lanjut, lihat Mengendalikan akses ke layanan dengan VPC endpoint di Panduan Pengguna Amazon VPC.

Example Kebijakan VPC endpoint untuk tindakan API MemoryDB

Berikut adalah contoh kebijakan titik akhir untuk API MemoryDB. Jika dilampirkan ke sebuah endpoint, kebijakan ini memberikan akses ke tindakan API MemoryDB terdaftar untuk semua prinsipal di semua sumber daya.

```
{
 "Statement": [{
  "Principal": "*",
  "Effect": "Allow",
  "Action": [
   "memorydb:CreateCluster",
   "memorydb:UpdateCluster",
```

```
    "memorydb:CreateSnapshot"
   ],
   "Resource": "*"
  }]
}
```

Example Kebijakan VPC endpoint yang menolak semua akses dari yang ditentukanAWSakun

Kebijakan VPC endpoint berikut mneolak akun AWS *123456789012* semua akses ke sumber daya menggunakan titik akhir. Kebijakan ini memungkinkan semua tindakan dari akun lain.

```
{
 "Statement": [{
   "Action": "*",
   "Effect": "Allow",
   "Resource": "*",
   "Principal": "*"
  },
  {
   "Action": "*",
   "Effect": "Deny",
   "Resource": "*",
   "Principal": {
    "AWS": [
     "123456789012"
    ]
   }
  }
 ]
}
```

Layanan update di MemoryDB untuk Redis

MemoryDB for Redis secara otomatis memonitor armada kluster dan node Anda untuk menerapkan pembaruan layanan saat tersedia. Biasanya, Anda mengatur jendela pemeliharaan yang telah ditentukan sehingga MemoryDB 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 kapan dan pembaruan mana yang diterapkan. Anda juga dapat memantau kemajuan pembaruan ini ke klaster MemoryDB yang Anda pilih secara real time.

Mengelola pembaruan layanan

Pembaruan layanan MemoryDB dirilis secara teratur. Jika Anda memiliki satu atau beberapa klaster yang memenuhi syarat untuk pembaruan layanan tersebut, Anda menerima pemberitahuan melalui email, SNS, Personal Health Dashboard (PHD), dan peristiwa Amazon CloudWatch saat pembaruan dirilis. Pembaruan juga ditampilkan padaPembaruan layananhalaman pada konsol MemoryDB. Dengan menggunakan dasbor ini, Anda dapat melihat semua pembaruan layanan dan status mereka untuk armada MemoryDB Anda.

Anda mengontrol kapan harus menerapkan pembaruan sebelum pemutakhiran otomatis dimulai. Kami sangat menyarankan Anda menerapkan pembaruan jenis apa punPembaruan keamanansesegera mungkin untuk memastikan bahwa MemoryDB Anda selalu up-to-date dengan patch keamanan saat ini.

Bagian berikut membahas opsi ini secara terperinci.

Topik

Menerapkan pembaruan layanan

Anda dapat mulai menerapkan pembaruan layanan ke armada Anda sejak pembaruan memilikitersediastatus. Pembaruan layanan bersifat kumulatif. Dengan kata lain, pembaruan apa pun yang belum Anda terapkan disertakan dengan pembaruan terbaru Anda.

Jika pembaruan layanan telah mengaktifkan pembaruan otomatis, Anda dapat memilih untuk tidak mengambil tindakan apa pun saat tersedia. MemoryDB akan menjadwalkan untuk menerapkan pembaruan selama jendela pemeliharaan cluster Anda setelahTanggal mulai diperbarui. Anda akan menerima pemberitahuan terkait untuk setiap tahap pembaruan.

> **Note**
>
> Anda hanya dapat menerapkan pembaruan layanan yang memilikitersediaataudijadwalkanstatus.

Untuk informasi selengkapnya tentang meninjau dan menerapkan pembaruan khusus layanan apa pun ke klaster MemoryDB yang berlaku, lihatMenerapkan pembaruan layanan menggunakan konsol.

Ketika pembaruan layanan baru tersedia untuk satu atau lebih cluster MemoryDB Anda, Anda dapat menggunakan konsol MemoryDB, API, atauAWS CLIuntuk menerapkan pembaruan. Bagian berikut menjelaskan opsi yang dapat Anda gunakan untuk menerapkan pembaruan.

Menerapkan pembaruan layanan menggunakan konsol

Untuk menampilkan daftar pembaruan layanan yang tersedia, beserta informasi lainnya, bukaPembaruan layananhalaman di konsol.

1. Masuk keAWS Management Consoledan buka memoryDB untuk konsol Redis dihttps://console.aws.amazon.com/memorydb/.
2. Di panel navigasi, pilihPembaruan layanan.

Di bawahDetail pembaruan layananAnda dapat melihat yang berikut ini:

- Nama pembaruan layanan: Nama unik dari pembaruan layanan
- Memperbarui deskripsi: Informasi lengkap tentang pembaruan layanan
- Tanggal mulai diperbarui: Jika atribut ini diatur, MemoryDB akan mulai menjadwalkan cluster Anda untuk diperbarui secara otomatis di jendela pemeliharaan yang sesuai setelah tanggal ini. Anda akan menerima pemberitahuan terlebih dahulu pada jendela pemeliharaan terjadwal yang tepat, yang mungkin bukan yang langsung setelahTanggal mulai diperbarui. Anda masih dapat menerapkan pembaruan ke kluster kapan pun Anda pilih. Jika atribut tidak diatur, pembaruan layanan tidak otomatis update diaktifkan dan MemoryDB tidak akan memperbarui klaster Anda secara otomatis.

DiStatus pembaruan klasterbagian, Anda dapat melihat daftar cluster di mana pembaruan layanan belum diterapkan atau baru saja diterapkan baru-baru ini. Untuk setiap klaster, Anda dapat melihat yang berikut ini:

- Nama klaster: Nama kluster
- Node diperbarui: Rasio node individual dalam klaster tertentu yang diperbarui atau tetap tersedia untuk pembaruan layanan tertentu.
- Jenis Perbarui: Jenis pembaruan layanan, yang merupakan salah satuPembaruan keamananatauPembaruan mesin
- Status: Status pembaruan layanan di kluster, yang merupakan salah satu dari berikut ini:
  - tersedia: Pembaruan tersedia untuk cluster yang diperlukan.

- dalam progres: Pembaruan sedang diterapkan ke klaster ini.
- dijadwalkan: Tanggal pembaruan telah dijadwalkan.
- lengkap: Pembaruan telah berhasil diterapkan. Cluster dengan status lengkap akan ditampilkan selama 7 hari setelah selesai.

Jika Anda memilih salah satu atau semua kluster dengantersediaataudijadwalkanstatus, dan kemudian memilihTerapkan sekarang, pembaruan akan mulai diterapkan pada cluster tersebut.

Menerapkan pembaruan layanan menggunakan AWS CLI

Setelah Anda menerima notifikasi bahwa pembaruan layanan tersedia, Anda dapat memeriksa dan menerapkannya menggunakan AWS CLI:

- Untuk mendapatkan deskripsi pembaruan layanan yang tersedia, jalankan perintah berikut:

  ```
  aws memorydb describe-service-updates --status available
  ```

  Untuk informasi selengkapnya, lihatmenjelaskan-layanan-update.
- Untuk menerapkan pembaruan layanan pada daftar kluster, jalankan perintah berikut:

  ```
  aws memorydb batch-update-cluster --service-update
  ServiceUpdateNameToApply=sample-service-update --cluster-names cluster-1
  cluster2
  ```

  Untuk informasi selengkapnya, lihatbatch update-cluster.

Referensi

Topik di bagian ini mencakup bekerja dengan MemoryDB API dan bagian MemoryDB dariAWS CLI. Juga disertakan pada bagian ini adalah pesan kesalahan dan notifikasi layanan yang umum.

- Menggunakan API MemoryDB
- Referensi API MemoryDB
- MemoryDB bagianAWS CLIReferensi

Menggunakan API MemoryDB

Bagian ini memberikan deskripsi berorientasi tugas tentang cara menggunakan dan menerapkan operasi MemoryDB. Untuk penjelasan lengkap tentang operasi ini, lihatReferensi API MemoryDB.

Topik

- Menggunakan API kueri
- Pustaka yang tersedia
- Memecahkan masalah aplikasi

Menggunakan API kueri

Parameter kueri

Permintaan berbasis Kueri HTTP adalah permintaan HTTP yang menggunakan HTTP verb GET atau POST dan parameter Query yang 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 bilangan bulat mulai dari 1.

Autentikasi permintaan Query

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 terdaftar dengan AWS dan memiliki Access Key ID dan Secret Access Key.

Proses Autentikasi Query

1. Pengirim membangun permintaan ke AWS.
2. Pengirim menghitung tanda tangan permintaan, Keyed-Hashing 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 Access Key ID (kunci-identifier dari Secret Access Key yang digunakan) untukAWS.
4. AWS menggunakan Access Key ID untuk mencari Secret Access Key.
5. AWS menghasilkan tanda tangan dari data permintaan dan Secret Access Key menggunakan algoritma yang sama dengan yang digunakan untuk menghitung tanda tangan pada permintaan.
6. Jika tanda tangan cocok, maka permintaan tersebut dianggap otentik. Jika perbandingan gagal, maka permintaan dibuang, dan AWS menghasilkan respons kesalahan.

> **Note**
>
> Jika permintaan berisi parameter `Timestamp`, tanda tangan yang dihitung untuk permintaan akan kedaluawrsa dalam 15 menit setelah nilainya.
> Jika permintaan berisi parameter `Expires`, tanda tangan berakhir pada waktu yang ditentukan oleh parameter `Expires`.

Untuk menghitung tanda tangan permintaan

1. Buat string kueri canonicalized yang 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 tubuh POST (jika Content-Type adalah aplikasi/x-www-form-urlencoded).
   b. URL mengkodekan nama parameter dan nilainya sesuai dengan aturan berikut:
      i. Jangan melakukan enkode URL karakter unreserved apa pun yang didefiniskan RFC 3986. Karakter yang tidak dicadangkan adalah A-Z, a-z, 0-9, tanda hubung (-), garis bawah (_), titik (.), dan tilde (~).
      ii. Gunakan enkode persen pada semua karakter lain dengan%XY, di mana X dan Y adalah karakter hex 0-9 dan huruf besar A-F.
      iii. Gunakan enkode persen pada karakter UTF-8 yang diperluas dalam bentuk%XY%ZA...
      iv. Gunakan enkode persen pada karakter spasi sebagai%20 (dan bukan +, seperti skema pengkodean yang umum dilakukan).
   c. Pisahkan nama parameter yang dienkodekan dari nilai yang dienkodekan dengan tanda sama dengan (=) (karakter ASCII 61), meskipun jika nilai parameter itu kosong.

   d. Pisahkan pasangan nama-nilai dengan tanda ampersand (&) (kode ASCII 38).

2. Buat string untuk menandatangani sesuai dengan pseudo-tata bahasa berikut (tanda “\ n” merupakan 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 query. Jika HTTPRequestURI kosong, gunakan garis miring ke depan (/).

3. Hitung HMAC sesuai RFC 2104 dengan string yang baru saja Anda buat, Secret Access Key Anda sebagai kunci, dan SHA256 atau SHA1 sebagai algoritma hash.

   Untuk informasi lain, lihat https://www.ietf.org/rfc/rfc2104.txt.

4. Mengonversi nilai yang dihasilkan ke base64.

5. Sertakan nilai sebagai nilai dari parameter `Signature` dalam permintaan.

Misalnya, berikut ini adalah permintaan sampel (baris baru ditambahkan untuk kejelasan).

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=DescribeClusters
    &ClusterName=myCluster
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2021-01-01
```

Untuk string query sebelumnya, Anda akan menghitung tanda tangan HMAC atas string berikut.

```
GET\n
    memory-db.amazonaws.com\n
    Action=DescribeClusters
    &ClusterName=myCluster
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
```

```
    &X-Amz-Credential=AKIADQKE4SARGYLE%2F20140523%2Fus-east-1%2Fmemorydb%2Faws4_request
    &X-Amz-Date=20210801T223649Z
    &X-Amz-SignedHeaders=content-type%3Bhost%3Buser-agent%3Bx-amz-content-sha256%3Bx-
amz-date
        content-type:
        host:memory-db.us-east-1.amazonaws.com
        user-agent:ServicesAPICommand_Client
    x-amz-content-sha256:
    x-amz-date:
```

Hasilnya adalah permintaan yang ditandatangani berikut.

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=DescribeClusters
    &ClusterName=myCluster
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20141201/us-east-1/memorydb/aws4_request
    &X-Amz-Date=20210801T223649Z
    &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 menghitung tanda tangan permintaan, lihat topikProses tanda tangan Signature Version 4dan subtopiknya.

Pustaka yang tersedia

AWS menyediakan kit pengembangan perangkat lunak (SDK) untuk developer perangkat lunak yang lebih suka membangun aplikasi menggunakan API bahasa tertentu alih-alih API Kueri. SDK ini menyediakan fungsi dasar (tidak termasuk 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 lainnya, lihatContoh kode & perpustakaan.

Memecahkan masalah aplikasi

MemoryDB memberikan kesalahan deskriptif dan spesifik untuk membantu Anda memecahkan masalah saat berinteraksi dengan API MemoryDB.

Menarik 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`Error`node dalam respon dari MemoryDB API.

Sintaks XPath menyediakan cara sederhana untuk mencari keberadaan simpul `Error`, serta cara mudah untuk menarik kode kesalahan dan pesan. Snippet kode berikut menggunakan Perl dan modul XML::XPath untuk menentukan jika kesalahan terjadi selama permintaan. Jika terjadi kesalahan, kode akan mencetak kode kesalahan pertama dan pesan 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"; }
```

Tips penyelesaian masalah

Kami merekomendasikan proses berikut untuk mendiagnosis dan menyelesaikan masalah dengan API MemoryDB.

- Verifikasi bahwa MemoryDB berjalan dengan benar.

  Untuk melakukan ini, cukup buka jendela browser dan kirimkan permintaan permintaan ke layanan MemoryDB (seperti https://memory-db.us-east-1.amazonaws.com). Sebuah MissingAuthenticationTokenException atau UnknowNOperationException mengkonfirmasi bahwa layanan tersedia dan menanggapi permintaan.
- Periksa struktur permintaan Anda.

Setiap operasi MemoryDB memiliki halaman referensi diReferensi API MemoryDB. Periksa ulang bahwa Anda menggunakan parameter dengan benar. Untuk memberikan Anda gambaran apa yang mungkin salah, lihat contoh permintaan atau skenario pengguna untuk melihat apakah contoh itu melakukan operasi yang serupa.

- Periksa ke forum.

  MemoryDB 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/.

Kuota untuk Amazon MemoryDB untuk Redis

AWS Akun Anda memiliki kuota default, 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 meminta penambahan kuota, lihat Meminta Penambahan Kuota dalam Panduan Pengguna Kuota Layanan. Jika kuota belum tersedia dalam Kuota Layanan, gunakan formulir penambahan batas.

AWS Akun Anda memiliki kuota berikut yang terkait dengan MemoryDB.

Sumber daya	Default
Simpul per Wilayah	300
Node per cluster per jenis instance	90
Node per pecahan	6
Grup parameter per Wilayah	150
Grup subnet per Wilayah	150
Subnet per grup subnet	20
Pengguna per grup pengguna	100
Jumlah total pengguna	1000
Jumlah grup pengguna	100

Riwayat dokumen untuk Panduan Pengguna MemoryDB

Tabel berikut menguraikan dokumentasi rilis untuk MemoryDB.

Perubahan	Deskripsi	Tanggal
MemoryDB sekarang mendukung otentikasi pengguna menggunakan IAM	IAM Authentication memungkinkan Anda untuk mengotentikasi koneksi ke MemoryDB untuk Redis menggunakan identitas. AWS Identity and Access Management Ini memungkin kan Anda untuk memperkua t model keamanan Anda dan menyederhanakan banyak tugas keamanan administratif. Untuk informasi selengkapnya, lihat Mengautentikasi dengan IAM.	10 Mei 2023
MemoryDB sekarang mendukung Redis 7	Rilis ini membawa beberapa fitur baru untuk MemoryDB untuk Redis: fungsi Redis, perbaikan ACL, Sharded Pub/Sub dan ditingkatkan I/O multiplexing. Untuk informasi selengkapnya, lihat Redis versi mesin.	9 Mei 2023
MemoryDB sekarang menawarkan node reserved	Node pesanan memberikan discount signifikan dibanding kan harga node sesuai permintaan. Node pesanan bukanlah node fisik, melainkan discount penagihan yang diterapkan untuk penggunaa	27 Desember 2022

	n node sesuai permintaa n di akun Anda. Untuk informasi selengkapnya, lihat MemoryDB node.	
MemoryDB sekarang mendukung Data Tiering	MemoryDB untuk Redis data tiering. Anda dapat menggunakan tiering data sebagai cara berbiaya lebih rendah untuk menskalakan klaster Anda hingga ratusan terabyte kapasitas. Untuk informasi selengkapnya, lihat Tingkat data.	Selasa, 3 November 2022
MemoryDB sekarang mendukung format JavaScript Object Notation (JSON)	Format JavaScript Object Notation (JSON) asli adalah cara sederhana, schemaless untuk menyandikan kumpulan data kompleks di dalam cluster Redis. Anda dapat menyimpan dan mengakses data secara native menggunak an format JavaScript Object Notation (JSON) di dalam klaster Redis dan memperbar ui data JSON yang tersimpan dalam klaster tersebut, tanpa perlu mengelola kode kustom untuk membuat serial dan deserialisasi. Untuk informasi selengkapnya, lihat Memulai dengan JSON.	25 Mei 2022

MemoryDB sekarang mendukung AWS PrivateLink	AWSPrivateLinkmemungkinkan Anda mengakses operasi API, atau AWS Direct Connect Connect. Untuk informasi selengkapnya, lihat API MemoryDB dan VPC endpoint antarmuka (). AWS PrivateLink	24 Januari 2022
Rilis awal	Rilis awal Panduan Pengguna MemoryDB. Untuk informasi selengkapnya, lihat Apa yang dimaksud dengan MemoryDB?	19 Agustus 2021

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.