Panduan Pengguna

AWS Modernisasi Mainframe

AWS Modernisasi Mainframe: Panduan Pengguna

Table of Contents

Apa itu Modernisasi AWS Mainframe?

AWS Modernisasi Mainframe membantu Anda memodernisasi aplikasi mainframe Anda ke lingkungan runtime yang dikelola. AWS Ini menyediakan alat dan sumber daya untuk membantu Anda merencanakan dan menerapkan migrasi dan modernisasi. Anda dapat menganalisis aplikasi mainframe yang ada, mengembangkan atau memperbaruinya menggunakan COBOL atau PL/I, dan menerapkan pipa otomatis untuk integrasi berkelanjutan dan pengiriman berkelanjutan (CI/CD) aplikasi. Anda dapat memilih antara pola refactoring dan replatforming otomatis, tergantung pada kebutuhan klien Anda. Jika Anda seorang konsultan yang membantu klien memigrasikan beban kerja mainframe mereka, Anda dapat menggunakan alat Modernisasi AWS Mainframe untuk semua fase perjalanan migrasi dan modernisasi, mulai dari perencanaan awal hingga operasi cloud pasca-migrasi.

Anda dapat menggunakan Modernisasi AWS Mainframe untuk membantu Anda membuat dan mengelola lingkungan runtime secara efisien AWS untuk aplikasi mainframe Anda, serta untuk mengelola dan memantau aplikasi modern Anda.

Topik

> **Note**
>
> Sudahkah Anda terlibat dengan Mitra Kompetensi Migrasi AWS Mainframe atau Layanan AWS Profesional untuk proyek modernisasi mainframe Anda? Jika tidak, kami sangat menyarankan Anda melibatkan ahli untuk proyek Anda.
>
> - AWS Mitra Kompetensi Modernisasi Mainframe
> - AWS Layanan Profesional

Fitur dan kasus penggunaan Modernisasi AWS Mainframe mendukung pendekatan modernisasi evolusioner, yang memberikan kemenangan jangka pendek dengan meningkatkan kelincahan dan banyak peluang untuk mengoptimalkan dan berinovasi di kemudian hari. Untuk informasi selengkapnya, lihat Pendekatan modernisasi.

Fitur Modernisasi AWS Mainframe

AWS Fitur Modernisasi Mainframe mendukung kasus penggunaan berikut:

- Menilai: Kemampuan penilaian Modernisasi AWS Mainframe dapat membantu Anda menilai, cakupan, dan merencanakan proyek migrasi dan modernisasi.
- Refactor: didukung oleh AWS Blu Age, Anda dapat menggunakan refactoring untuk mengonversi bahasa pemrograman aplikasi lama, untuk membuat layanan makro atau layanan mikro, dan untuk memodernisasi antarmuka pengguna (UI) dan tumpukan perangkat lunak aplikasi.

  AWS Blu Insights sekarang tersedia dari AWS Management Console melalui single sign-on. Anda tidak perlu lagi mengelola kredensyal AWS Blu Insights yang terpisah. Anda dapat mengakses fitur AWS AWS Blu Age Codebase dan Transformation Center langsung dari file. AWS Management Console
- Replatform: didukung oleh solusi Micro Focus Enterprise, Anda dapat mem-port aplikasi di mana banyak kode sumber aplikasi dikompilasi ulang tanpa perubahan.
- IDE Pengembang: Modernisasi AWS Mainframe menawarkan lingkungan pengembangan terintegrasi (IDE) sesuai permintaan sehingga pengembang dapat menulis kode lebih cepat dengan pengeditan dan debugging cerdas, kompilasi kode instan, dan pengujian unit.
- Managed runtime: Lingkungan eksekusi terkelola Modernisasi AWS Mainframe terus memantau klaster Anda untuk menjaga beban kerja perusahaan tetap berjalan dengan komputasi penyembuhan mandiri dan penskalaan otomatis.
- Integrasi dan pengiriman berkelanjutan (CI/CD): Fitur CI/CD Modernisasi AWS Mainframe membantu tim pengembangan aplikasi memberikan perubahan kode lebih sering dan andal, yang mempercepat kecepatan migrasi, meningkatkan kualitas, dan membantu mengurangi untuk merilis fungsi bisnis baru. time-to-market
- Integrasi dengan AWS layanan lain: Modernisasi AWS Mainframe mendukung AWS CloudFormation, AWS PrivateLink, dan AWS Key Management Service untuk penyebaran berulang dan keamanan dan kepatuhan yang lebih besar.

- Ketersediaan yang diperluas: Modernisasi AWS Mainframe sekarang tersedia di AS Timur (Ohio), AS Barat (California N.), Asia Pasifik (Mumbai), Asia Pasifik (Seoul), Asia Pasifik (Singapura), Asia Pasifik (Tokyo), Eropa (London), dan Eropa (Paris).

Untuk informasi selengkapnya tentang fitur Modernisasi AWS Mainframe, lihat. https://aws.amazon.com/mainframe-modernization/features/

Pola

Pola Refactoring Otomatis, didukung oleh AWS Blu Age, difokuskan pada percepatan modernisasi dengan mengubah tumpukan aplikasi warisan lengkap dan lapisan datanya menjadi aplikasi berbasis Java modern sambil mempertahankan kesetaraan fungsional. Selama transformasi otomatis ini, ia menciptakan aplikasi multi-tier dengan front-end berbasis Angular, backend Java berkemampuan API dan lapisan data yang mengakses penyimpanan data modern. Proses refactoring menyediakan fungsionalitas yang setara dengan tumpukan lama untuk meningkatkan otomatisasi proyek yang menghasilkan kecepatan, kualitas, dan biaya yang lebih rendah untuk mencapai manfaat bisnis lebih cepat. Untuk informasi selengkapnya, lihat Modernisasi AWS Mainframe Automated Refactor.

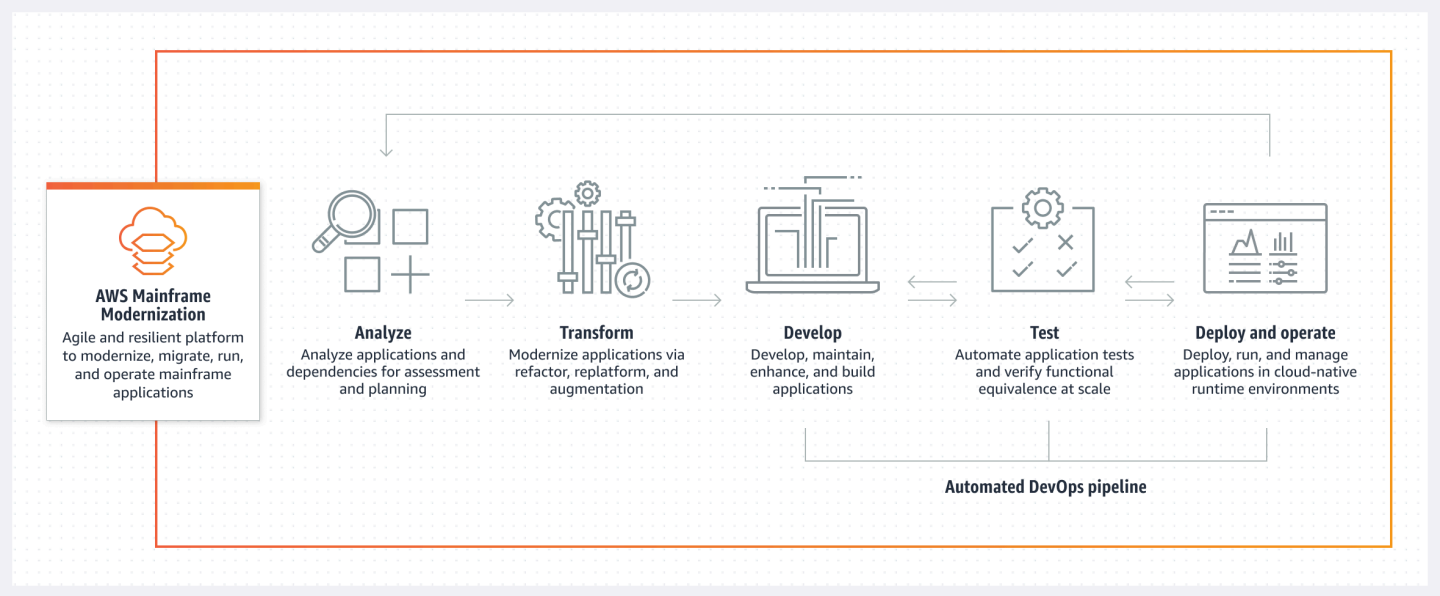
Pola Replatforming, didukung oleh rangkaian Micro Focus Enterprise, difokuskan pada pelestarian bahasa aplikasi, kode, dan artefak untuk meminimalkan dampak terhadap aset dan tim aplikasi. Ini membantu pelanggan mempertahankan pengetahuan dan keterampilan aplikasi. Sementara perubahan aplikasi terbatas, pola ini juga memfasilitasi modernisasi infrastruktur dan proses. Infrastruktur diubah menjadi layanan terkelola berbasis cloud modern sementara prosesnya diubah untuk mengikuti praktik terbaik untuk pengembangan aplikasi dan operasi TI. Untuk informasi selengkapnya, lihat AWS Mainframe Modernisasi Replatform.

Cara Memulai Modernisasi AWS Mainframe

Cobalah! Kami menawarkan tutorial dan contoh aplikasi untuk membantu Anda memahami apa yang ditawarkan Modernisasi AWS Mainframe. Pilih salah satu Tutorial: Runtime Terkelola untuk AWS Blu Age atau Tutorial: Runtime terkelola untuk Micro Focus untuk step-by-step tutorial lengkap.

Jika Anda tertarik dengan refactoring otomatis, lihat alat AWS Blu Age di. BluInsights Anda juga dapat mengatur AppStream 2.0 untuk mengakses IDE Pengembang AWS Blu Age, atau alat Micro Focus Enterprise Analyzer dan Micro Focus Enterprise Developer.

Tutorial dan contoh aplikasi hanya memberi Anda gambaran tentang apa yang disediakan oleh Modernisasi AWS Mainframe. Saat Anda siap untuk memulai proyek modernisasi, lihat Pendekatan modernisasi detail tentang tahapan dan tugas proyek modernisasi.



Layanan terkait

Selain Blu Insights untuk refactoring otomatis, Anda dapat menggunakan layanan berikut AWS dengan Modernisasi Mainframe. AWS

- Amazon RDS untuk hosting database yang dimigrasikan.
- Amazon S3 untuk menyimpan binari aplikasi dan file definisi.
- Amazon FSx atau Amazon EFS untuk menyimpan data aplikasi.
- Amazon AppStream untuk akses ke alat Micro Focus Enterprise Analyzer dan Micro Focus Enterprise Developer.
- AWS CloudFormation untuk DevOps pipeline otomatis yang dapat Anda gunakan untuk menyiapkan CI/CD untuk aplikasi yang dimigrasi.
- AWS Migration Hub
- AWS DMS untuk memigrasikan database Anda.

Mengakses Modernisasi AWS Mainframe

Saat ini, Anda dapat mengakses Modernisasi AWS Mainframe melalui konsol di https://console.aws.amazon.com/m2/. Untuk daftar wilayah di mana Modernisasi AWS Mainframe tersedia, lihat Titik akhir Modernisasi AWS Mainframe dan kuota di. Referensi Umum Amazon Web Services

Apakah Anda pengguna Modernisasi AWS Mainframe pertama kali?

Jika Anda adalah pengguna pertama kali Modernisasi AWS Mainframe, kami sarankan Anda mulai dengan membaca bagian berikut:

- Memulai
- Menyiapkan

Harga

AWS Biaya Modernisasi Mainframe untuk penggunaan instance yang mendukung lingkungan runtime terkelola. Selain itu, Modernisasi AWS Mainframe menawarkan beberapa alat tanpa biaya tambahan. Anda bertanggung jawab atas biaya yang dikeluarkan untuk AWS layanan lain yang Anda gunakan sehubungan dengan Modernisasi AWS Mainframe. AWS akan memberikan pemberitahuan 30 hari sebelum perubahan harga berlaku untuk penggunaan Modernisasi AWS Mainframe. Untuk informasi lebih lanjut, lihat Modernisasi Mainframe dengan. AWS

Dengan AWS Blu Insights, Anda membayar untuk penggunaan Pusat Transformasi. Untuk informasi selengkapnya, lihat Harga Modernisasi AWS Mainframe.

Menyiapkan AWS Modernisasi Mainframe

Sebelum Anda dapat mulai menggunakan Modernisasi AWS Mainframe, Anda atau administrator Anda perlu menyelesaikan beberapa langkah.

Topik

Mendaftar untuk Akun AWS

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

Untuk mendaftar untuk Akun AWS

1. Buka https://portal.aws.amazon.com/billing/signup.
2. Ikuti petunjuk secara online.

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

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

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

Membuat pengguna administratif

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

Amankan Pengguna root akun AWS

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

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

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

Membuat pengguna administratif

1. Aktifkan Pusat Identitas IAM.

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

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

Masuk sebagai pengguna administratif

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

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

Memulai Modernisasi AWS Mainframe

Untuk memulai Modernisasi AWS Mainframe, Anda dapat mengikuti tutorial yang memperkenalkan Anda pada layanan dan setiap mesin runtime.

Topik

- Tutorial: Runtime Terkelola untuk AWS Blu Age
- Tutorial: Runtime terkelola untuk Micro Focus

Untuk melanjutkan pembelajaran, lihat tutorial berikut.

- Tutorial: Menyiapkan build Micro Focus untuk aplikasi BankDemo sampel
- Tutorial: Menyiapkan pipeline CI/CD untuk digunakan dengan Pengembang Micro Focus Enterprise

Tutorial: Runtime Terkelola untuk AWS Blu Age

Tutorial ini menunjukkan cara menerapkan aplikasi modern AWS Blu Age ke dalam lingkungan runtime Modernisasi AWS Mainframe.

Topik

- Prasyarat
- Langkah 1: Unggah aplikasi demo
- Langkah 2: Buat definisi aplikasi
- Langkah 3: Buat lingkungan runtime
- Langkah 4: Buat aplikasi
- Langkah 5: Menyebarkan aplikasi
- Langkah 6: Mulai aplikasi
- Langkah 7: Akses aplikasi
- Langkah 8: Uji aplikasi
- Pembersihan sumber daya

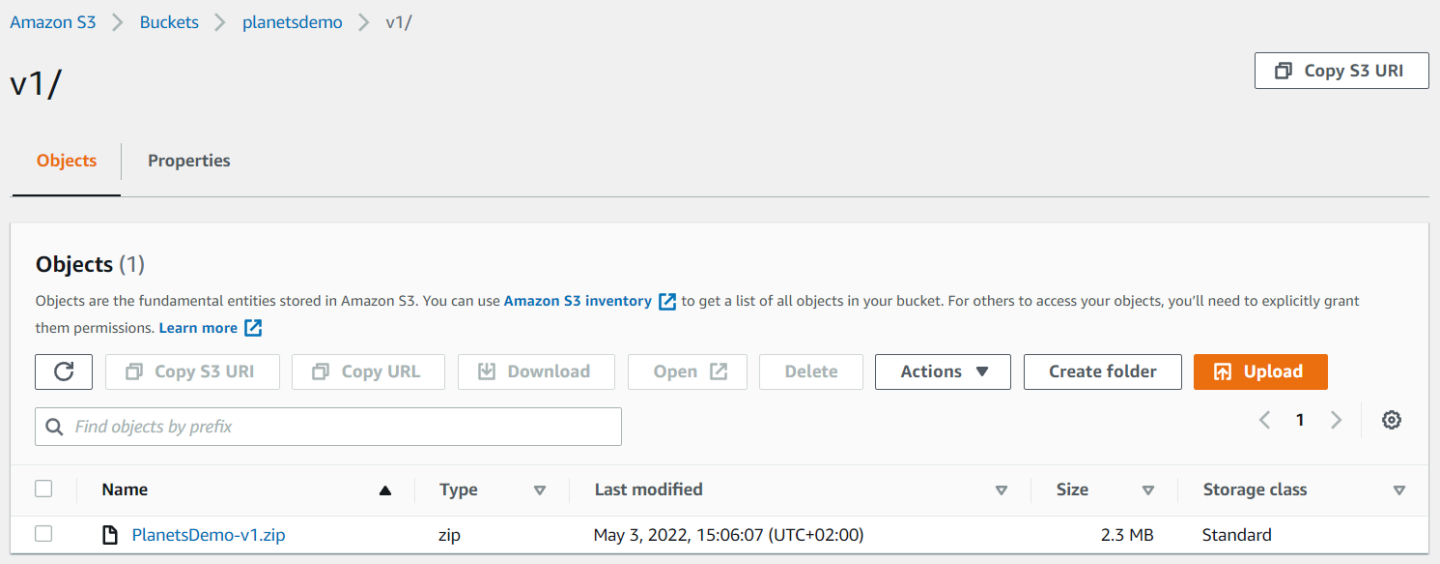
Prasyarat

Untuk menyelesaikan tutorial ini, unduh arsip aplikasi demo PlanetsDemo-v1.zip.

Aplikasi demo yang berjalan membutuhkan browser modern untuk akses. Apakah Anda menjalankan browser ini dari desktop atau dari instans Amazon Elastic Compute Cloud, misalnya, dalam VPC, menentukan pengaturan keamanan Anda.

Langkah 1: Unggah aplikasi demo

Unggah aplikasi demo ke bucket Amazon S3. Pastikan bucket ini berada di tempat yang sama Wilayah AWS di mana Anda akan menyebarkan aplikasi. Contoh berikut menunjukkan bucket bernama planetsdemo, dengan key prefix, atau folder, bernama v1 dan arsip bernama. `planetsdemo-v1.zip`



> **Note**
>
> Folder dalam ember diperlukan.

Langkah 2: Buat definisi aplikasi

Untuk menerapkan aplikasi ke runtime terkelola, Anda memerlukan definisi aplikasi Modernisasi AWS Mainframe. Definisi ini adalah file JSON yang menjelaskan lokasi dan pengaturan aplikasi. Contoh berikut adalah definisi aplikasi untuk aplikasi demo:

```
{
    "template-version": "2.0",
    "source-locations": [{
        "source-id": "s3-source",
        "source-type": "s3",
        "properties": {
            "s3-bucket": "planetsdemo",
            "s3-key-prefix": "v1"
        }
    }],
    "definition": {
        "listeners": [{
            "port": 8196,
            "type": "http"
        }],
        "ba-application": {
            "app-location": "${s3-source}/PlanetsDemo-v1.zip"
        }
    }
}
```

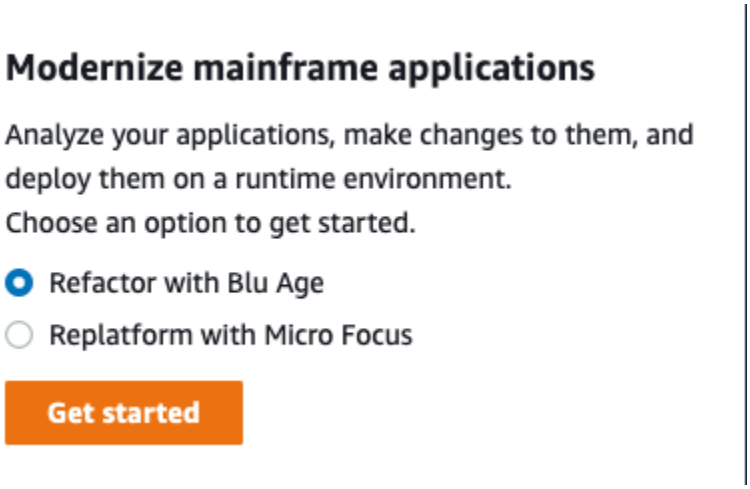
Ubah `s3-bucket` entri ke nama bucket tempat Anda menyimpan file zip aplikasi sampel.

Untuk informasi selengkapnya tentang definisi aplikasi, lihatAWS Sampel definisi aplikasi Blu Age.

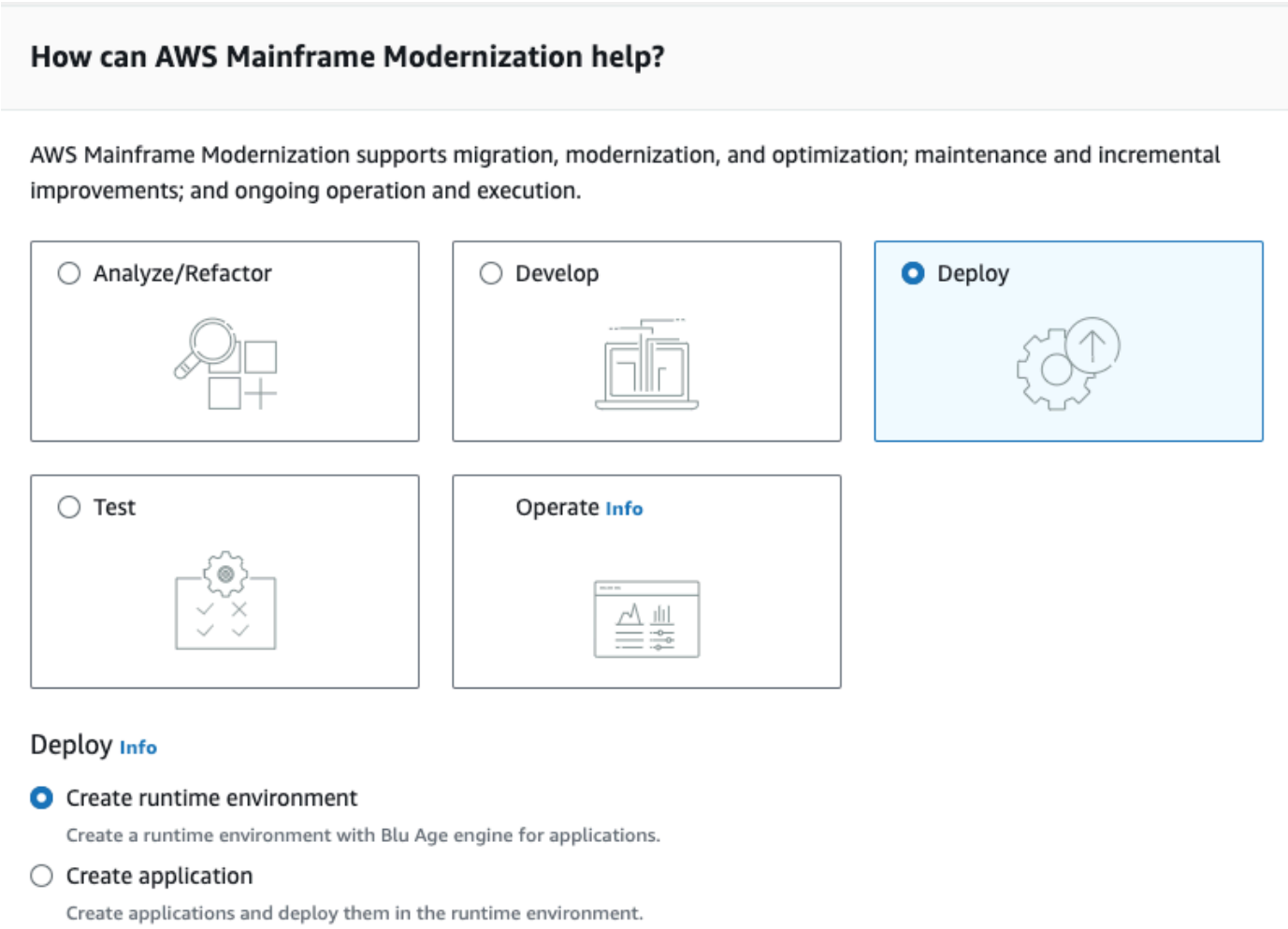
Langkah 3: Buat lingkungan runtime

Untuk membuat lingkungan runtime Modernisasi AWS Mainframe, lakukan langkah-langkah berikut:

1. Buka konsol Modernisasi AWS Mainframe.
2. Di Wilayah AWS pemilih, pilih Wilayah tempat Anda ingin membuat lingkungan. Ini Wilayah AWS harus cocok dengan Wilayah tempat Anda membuat bucket S3. Langkah 1: Unggah aplikasi demo
3. Di bawah Modernisasi aplikasi mainframe, pilih Refactor with Blu Age, lalu pilih Mulai.

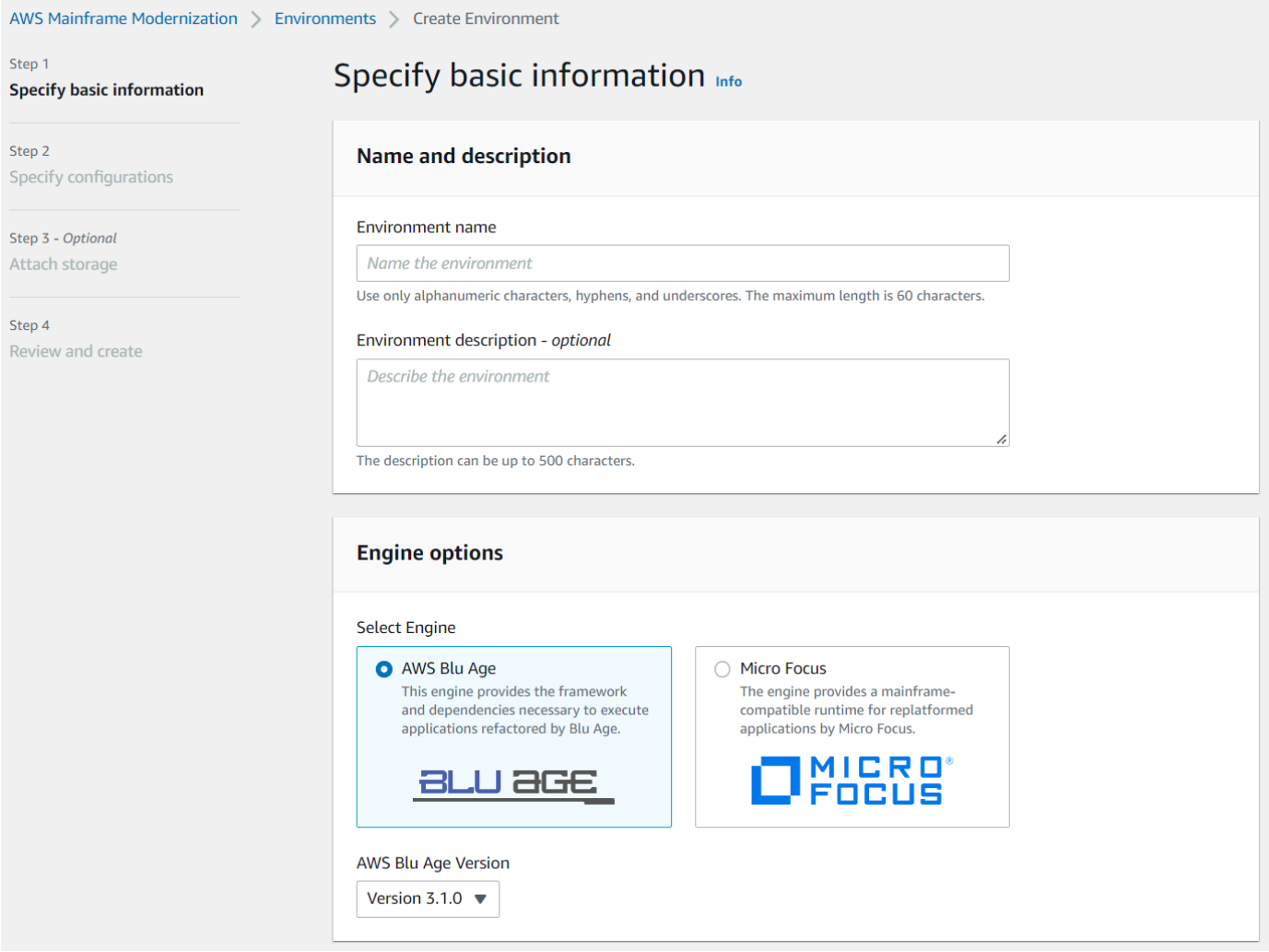


4. Di bawah Bagaimana Modernisasi AWS Mainframe dapat membantu, pilih Menerapkan dan Membuat lingkungan runtime.

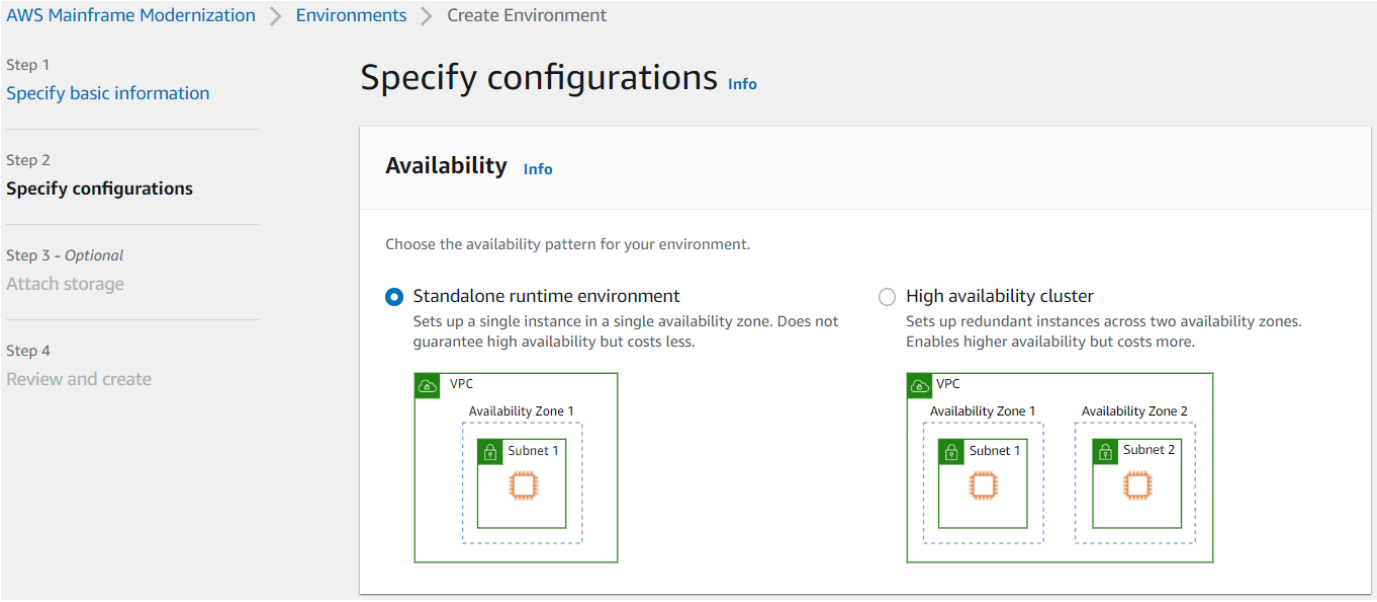


5. Di navigasi kiri, pilih Lingkungan, lalu pilih Buat lingkungan. Pada halaman Tentukan informasi dasar, masukkan nama dan deskripsi untuk lingkungan Anda, lalu pastikan mesin AWS Blu Age

dipilih. Secara opsional, Anda dapat menambahkan tag ke sumber daya yang dibuat. Lalu pilih Selanjutnya.

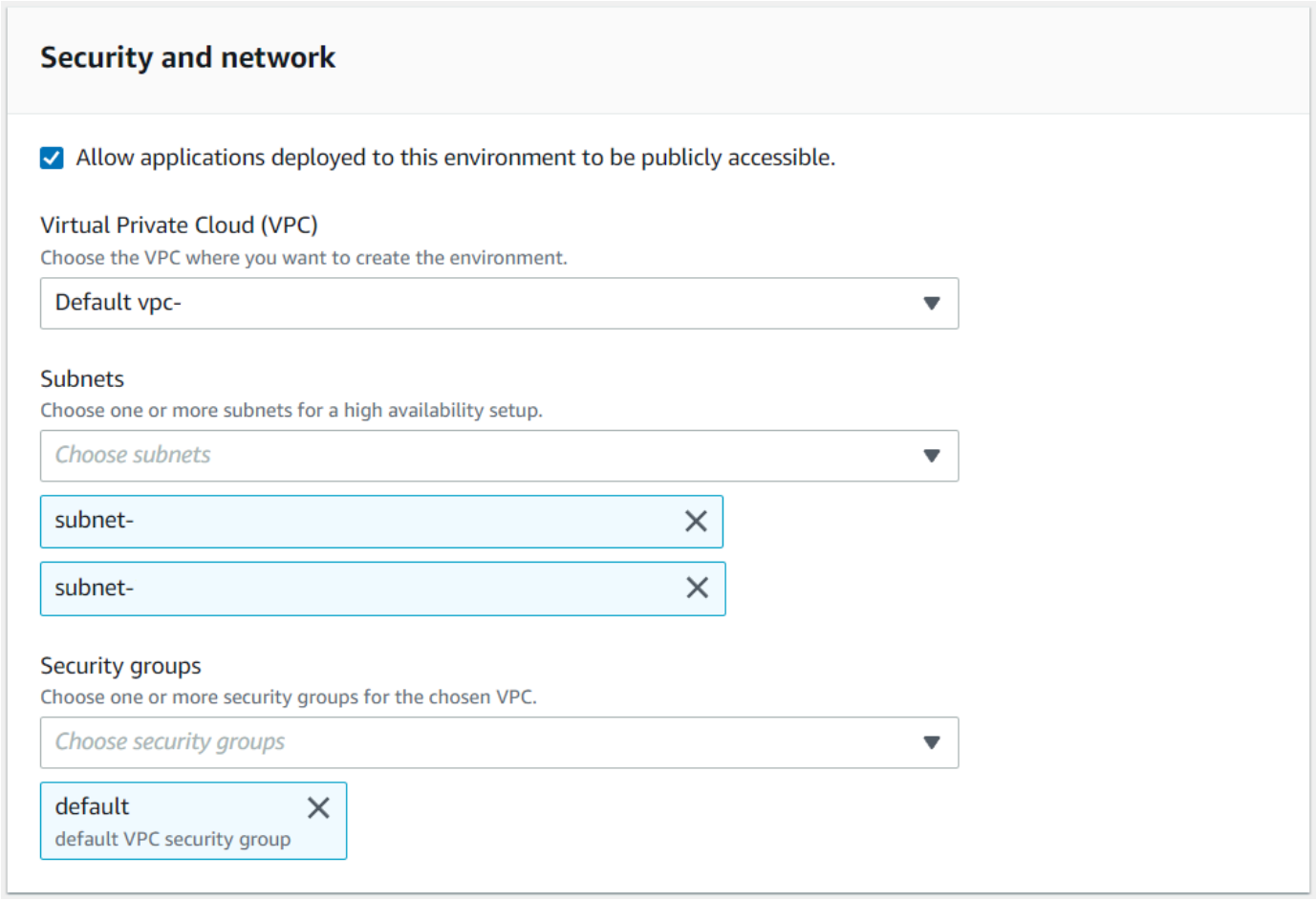


6. Pada halaman Tentukan konfigurasi, pilih Lingkungan runtime mandiri.



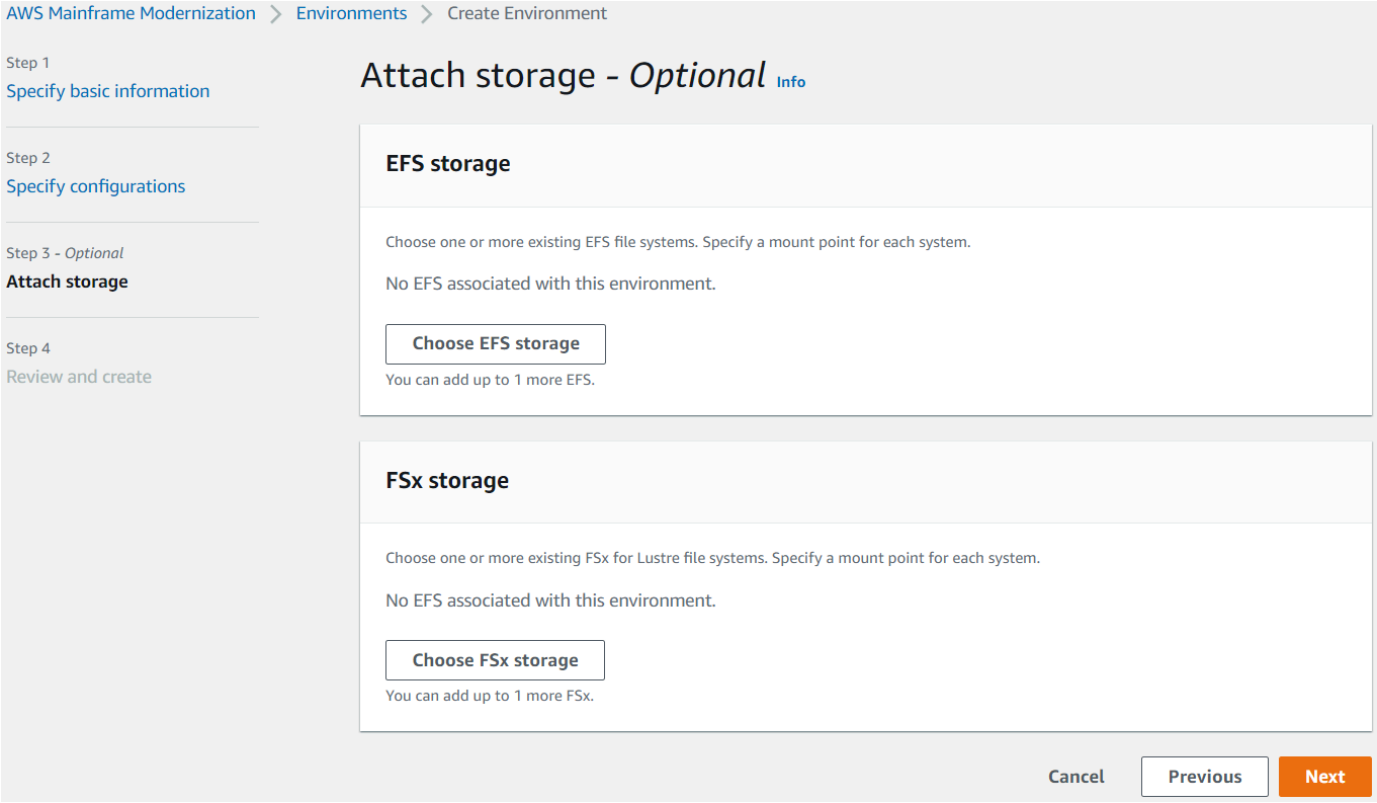
7. Di bawah Keamanan dan jaringan, lakukan perubahan berikut:

   - Pilih Izinkan aplikasi yang disebarkan ke lingkungan ini agar dapat diakses publik. Opsi ini memberikan alamat IP publik ke aplikasi sehingga Anda dapat mengaksesnya dari desktop Anda.
   - Pilih VPC. Anda dapat menggunakan Default.
   - Pilih dua subnet. Pastikan bahwa subnet memungkinkan penetapan alamat IP publik.
   - Pilih grup keamanan. Anda dapat menggunakan Default. Pastikan bahwa grup keamanan yang Anda pilih mengizinkan akses dari alamat IP browser ke port yang Anda tentukan di `listener` properti definisi aplikasi. Untuk informasi selengkapnya, lihat Langkah 2: Buat definisi aplikasi.



Jika Anda ingin mengakses aplikasi dari luar VPC yang Anda pilih, pastikan aturan masuk untuk VPC tersebut dikonfigurasi dengan benar. Untuk informasi selengkapnya, lihat Tidak dapat mengakses URL aplikasi.

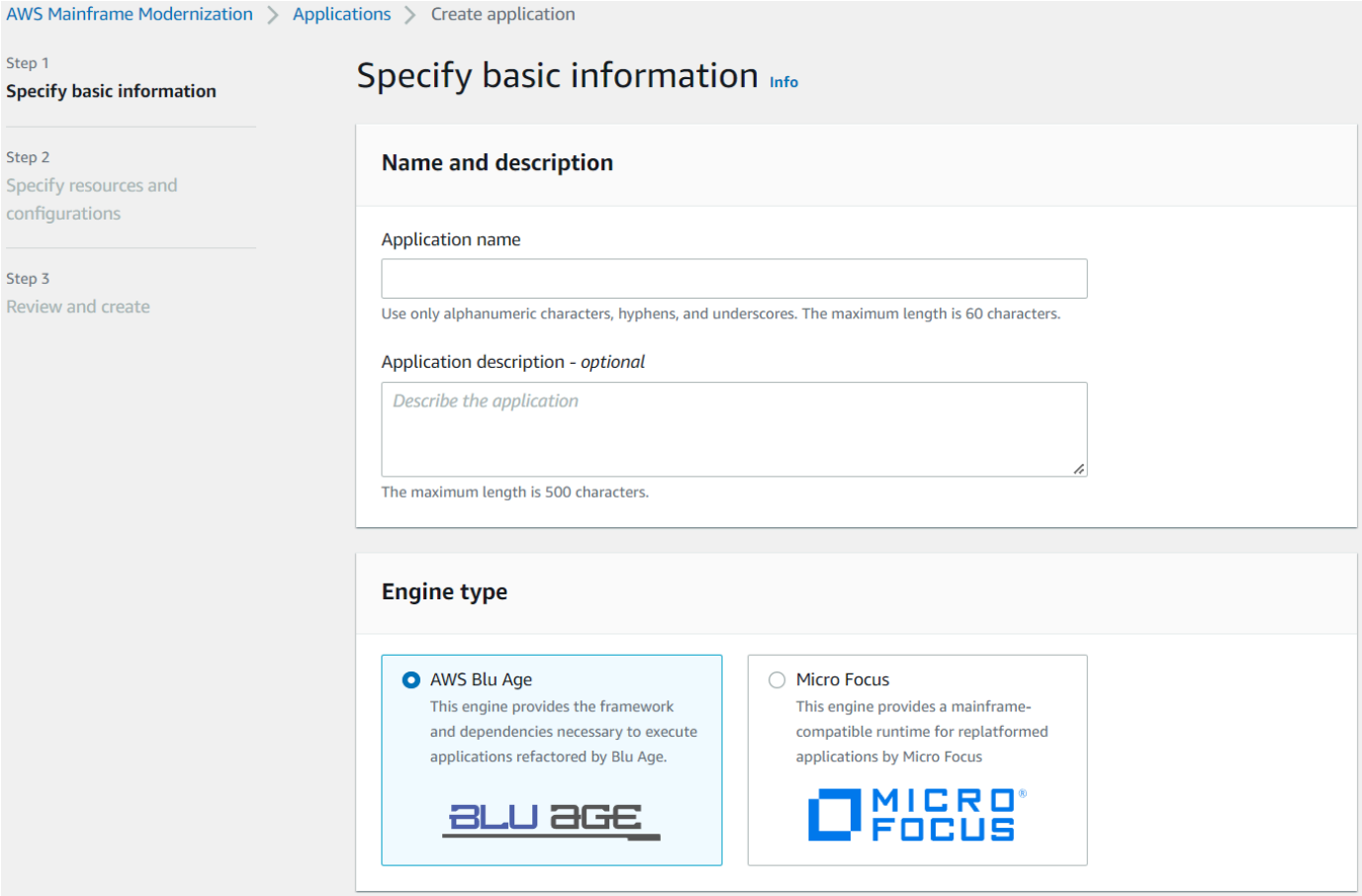
8. Pilih Berikutnya.
9. Di Lampirkan penyimpanan - Opsional, tinggalkan pilihan default dan pilih Berikutnya.



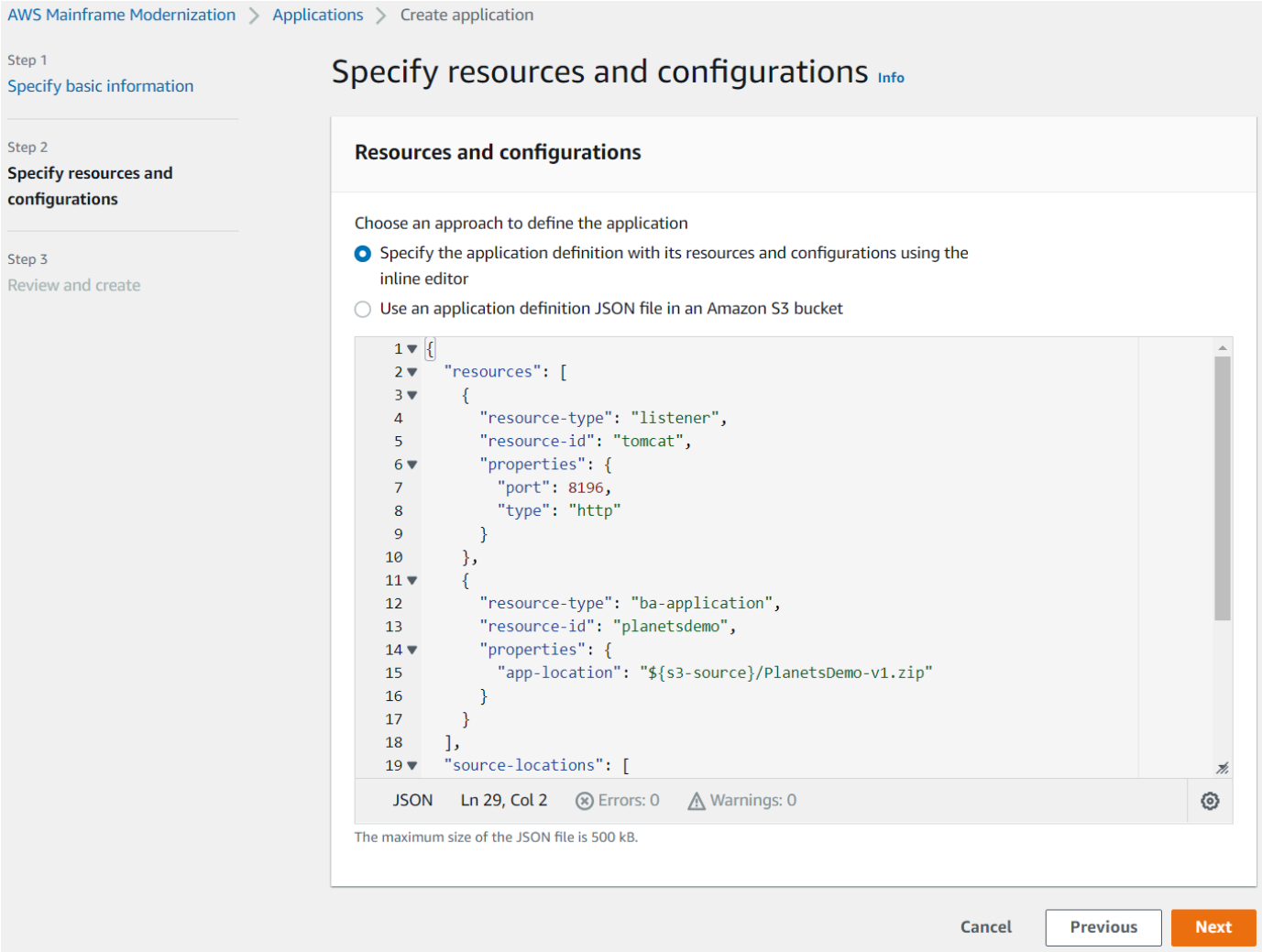
10. Dalam Jadwal pemeliharaan, pilih Tidak ada preferensi, lalu pilih Berikutnya.
11. Di Tinjau dan buat, tinjau informasi, lalu pilih Buat lingkungan.

Langkah 4: Buat aplikasi

1. Arahkan ke AWS Mainframe Modernisasi di. AWS Management Console
2. Di panel navigasi, pilih Aplikasi, lalu pilih Buat aplikasi. Pada halaman Tentukan informasi dasar, masukkan nama dan deskripsi untuk aplikasi, dan pastikan bahwa mesin AWS Blu Age dipilih. Lalu pilih Selanjutnya.



3. Pada halaman Tentukan sumber daya dan konfigurasi, salin dan tempel JSON definisi aplikasi yang diperbarui yang Anda buat. the section called “Langkah 2: Buat definisi aplikasi”

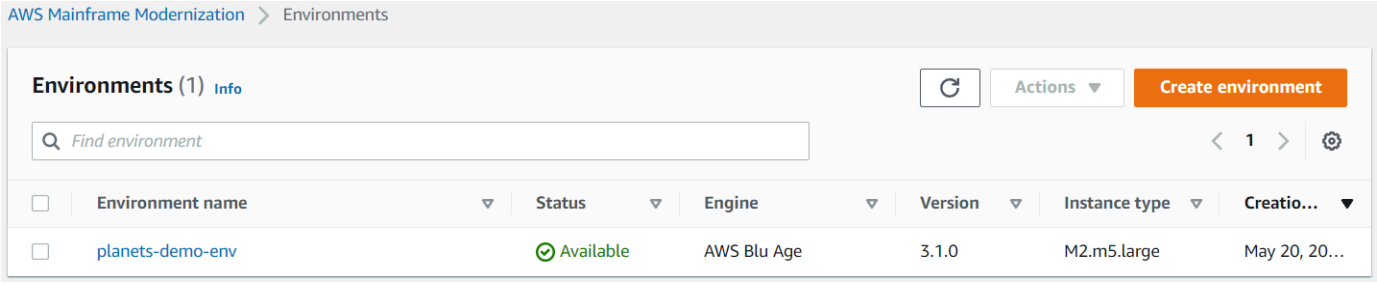


4. Di Tinjau dan buat, tinjau pilihan Anda, lalu pilih Buat aplikasi.

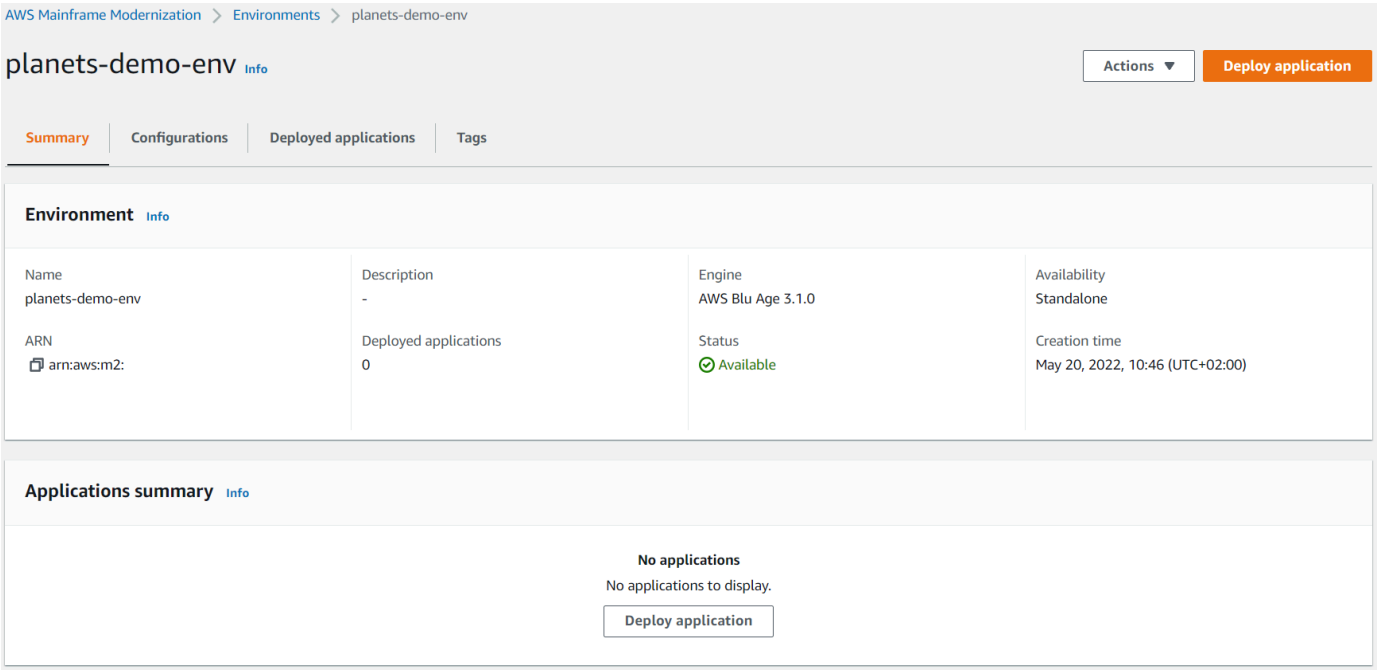
Langkah 5: Menyebarkan aplikasi

Setelah Anda berhasil membuat lingkungan runtime Modernisasi AWS Mainframe dan aplikasi, dan keduanya berada dalam status Tersedia, Anda dapat menyebarkan aplikasi ke lingkungan runtime. Caranya, lakukan langkah-langkah berikut:

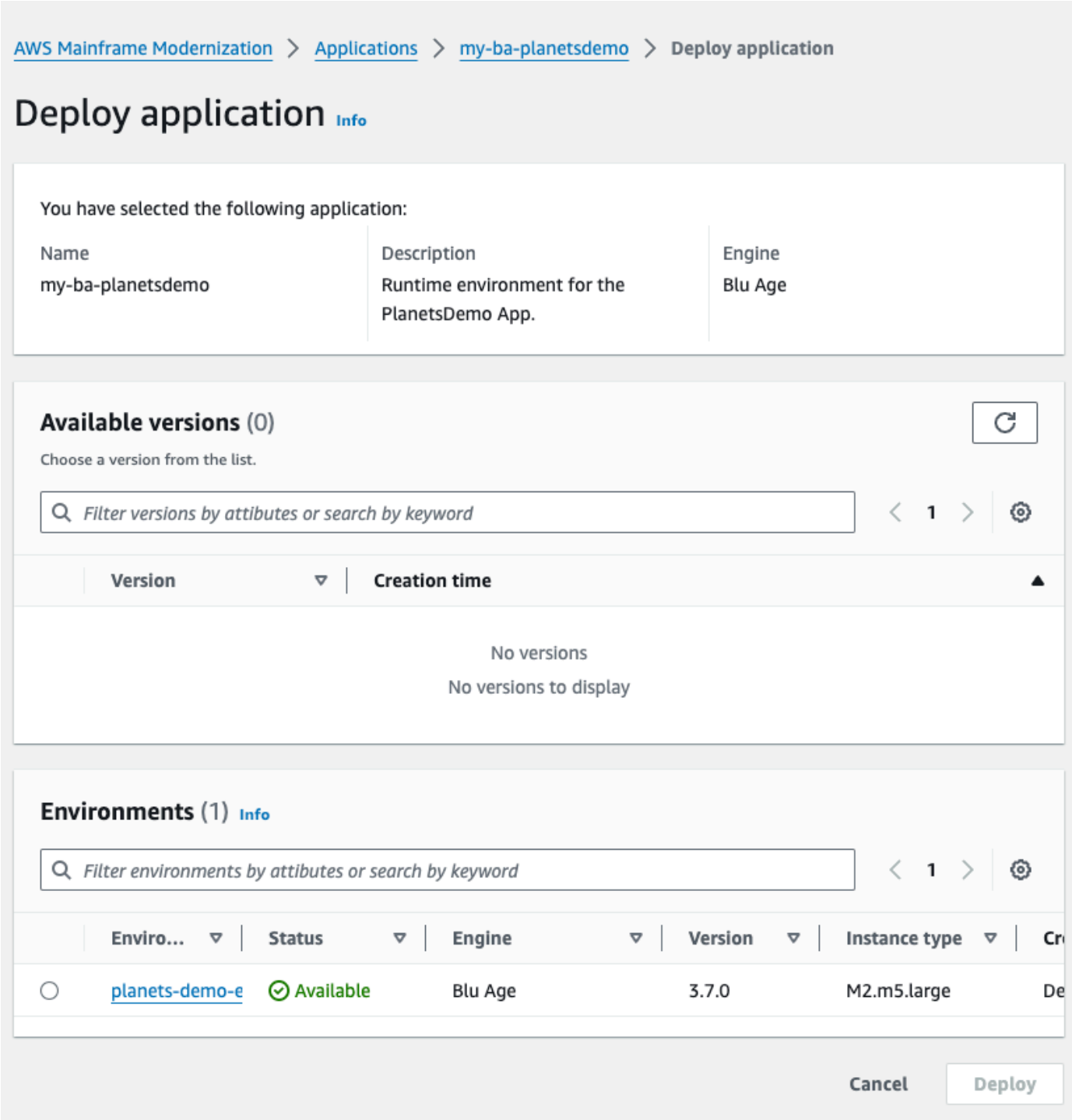
1. Arahkan ke Modernisasi Mainframe AWS di Konsol Manajemen. AWS Pada panel navigasi, pilih Lingkungan. Halaman daftar Lingkungan ditampilkan.



2. Pilih lingkungan runtime yang dibuat sebelumnya. Halaman detail lingkungan ditampilkan.
3. Pilih Menyebarkan aplikasi.



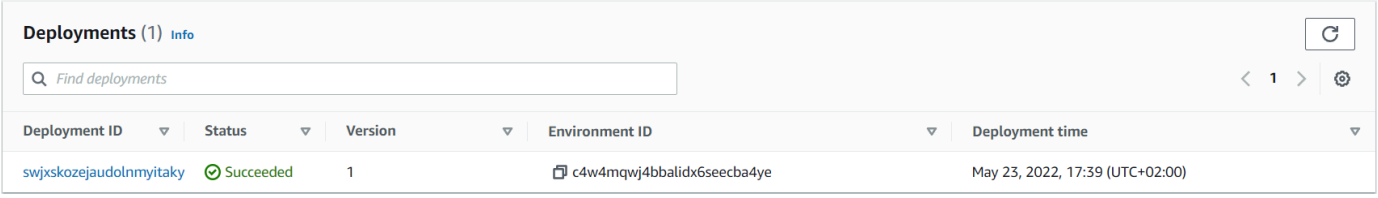
4. Pilih aplikasi yang dibuat sebelumnya, lalu pilih versi yang ingin Anda gunakan untuk aplikasi Anda. Kemudian pilih Deploy.



5. Tunggu hingga aplikasi menyelesaikan penerapannya. Anda akan melihat spanduk dengan pesan Aplikasi telah berhasil digunakan.

Langkah 6: Mulai aplikasi

1. Arahkan ke AWS Mainframe Modernisasi di AWS Management Console dan pilih Aplikasi.
2. Pilih aplikasi Anda, dan kemudian pergi ke Deployments. Status aplikasi harus Berhasil.



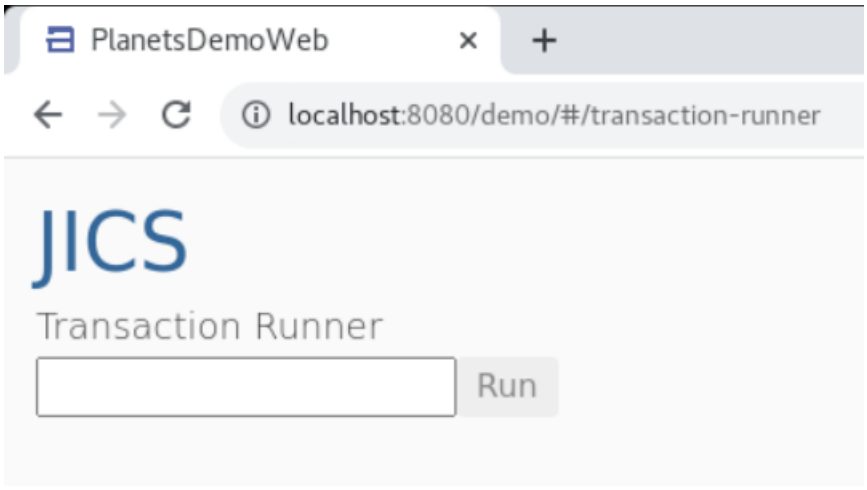
3. Pilih Tindakan, lalu pilih Mulai aplikasi.

Langkah 7: Akses aplikasi

1. Tunggu hingga aplikasi dalam status Running. Anda akan melihat spanduk dengan pesan Aplikasi telah dimulai dengan sukses.
2. Salin nama host DNS aplikasi. Anda dapat menemukan nama host ini di bagian Informasi aplikasi aplikasi.
3. Di browser, navigasikan kehttp://{hostname}:{portname}/PlanetsDemo-web-1.0.0/, di mana:

   - `hostname`adalah nama host DNS yang disalin sebelumnya.
   - `portname`adalah port Tomcat yang didefinisikan dalam definisi aplikasi yang Anda buat. Langkah 2: Buat definisi aplikasi

   Layar JICS muncul.



Jika Anda tidak dapat mengakses aplikasi, lihatTidak dapat mengakses URL aplikasi.

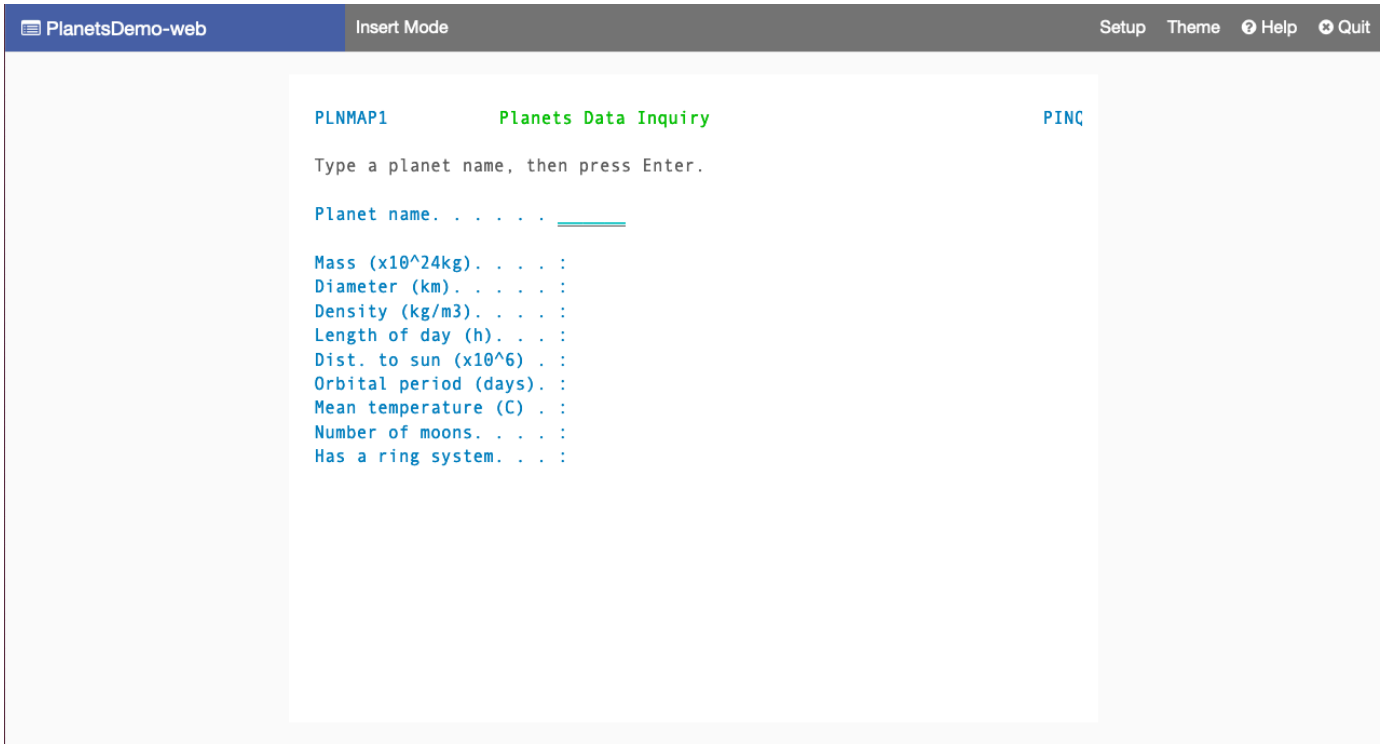
> **Note**
>
> Jika aplikasi tidak dapat diakses, dan aturan masuk pada grup keamanan memiliki 'IP Saya' yang dipilih pada port 8196, tentukan aturan untuk mengizinkan lalu lintas dari LB i/p pada port 8196.

Langkah 8: Uji aplikasi

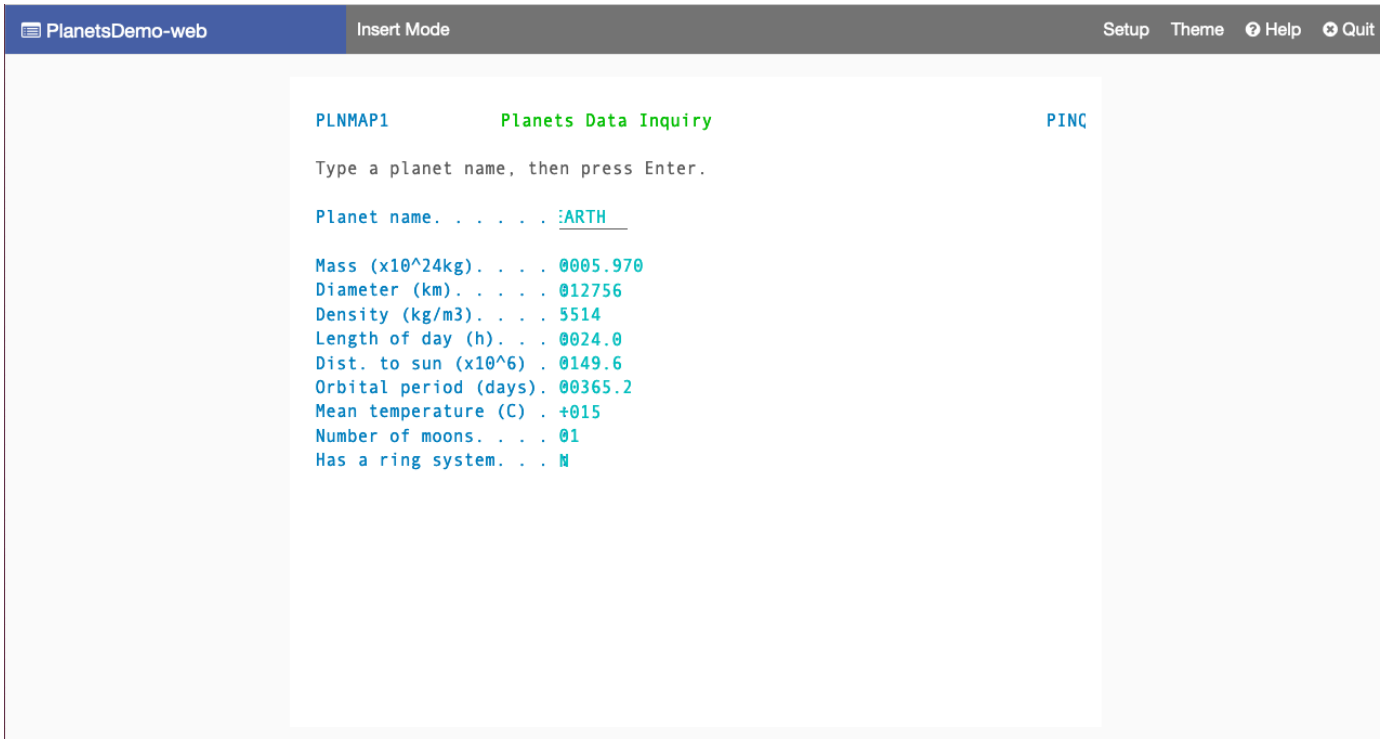
Pada langkah ini, Anda menjalankan transaksi di aplikasi yang dimigrasi.

1. Pada layar JICS, masukkan `PINQ` di kolom input, dan pilih Jalankan (atau tekan Enter) untuk memulai transaksi aplikasi.

   Layar aplikasi demo akan muncul.



2. Ketik nama planet di bidang yang sesuai dan tekan Enter.



Anda harus melihat detail tentang planet ini.

Pembersihan sumber daya

Jika Anda tidak lagi membutuhkan sumber daya yang Anda buat untuk tutorial ini, hapus untuk menghindari biaya tambahan. Untuk melakukannya, selesaikan langkah-langkah berikut:

- Jika aplikasi Modernisasi AWS Mainframe masih berjalan, hentikan.
- Hapus aplikasi. Untuk informasi selengkapnya, lihat Hapus aplikasi Modernisasi AWS Mainframe.
- Hapus lingkungan runtime. Lihat informasi yang lebih lengkap di Menghapus lingkungan AWS runtime Modernisasi Mainframe.

Tutorial: Runtime terkelola untuk Micro Focus

Tutorial ini menunjukkan cara menerapkan dan menjalankan aplikasi CardDemo sampel dalam lingkungan runtime terkelola Modernisasi AWS Mainframe dengan mesin runtime Micro Focus. Aplikasi CardDemo sampel adalah aplikasi kartu kredit yang disederhanakan yang dikembangkan untuk menguji dan memamerkan dan teknologi mitra untuk kasus AWS penggunaan modernisasi mainframe.

Dalam tutorial, Anda membuat sumber daya di tempat lainLayanan AWS. Ini termasuk Amazon Simple Storage Service, Amazon Relational Database ServiceAWS Key Management Service, AWS Secrets Manager dan.

Topik

Prasyarat

- Pastikan Anda memiliki akses ke emulator 3270 untuk menggunakan koneksi CICS. Emulator 3270 gratis dan uji coba tersedia dari situs web pihak ketiga. Atau, Anda dapat memulai instance AWS Mainframe Modernization AppStream 2.0 Micro Focus dan menggunakan emulator Rumba 3270 (tidak tersedia secara gratis).

  Untuk informasi tentang AppStream 2.0, lihatthe section called “Tutorial: Mengatur AppStream 2.0 untuk Enterprise Analyzer dan Enterprise Developer”.

  > **Note**
  >
  > Saat membuat tumpukan, pilih opsi Enterprise Developer (ED) dan bukan Enterprise Analyzer (EA).

- Unduh aplikasi CardDemo sampel dan unzip file yang diunduh ke direktori lokal mana pun. Direktori ini akan berisi subdirektori berjudul`CardDemo`.
- Identifikasi VPC di akun Anda di mana Anda dapat menentukan sumber daya yang dibuat dalam tutorial ini. VPC akan membutuhkan subnet di setidaknya dua Availability Zone. Untuk informasi selengkapnya tentang Amazon VPC, lihat Cara kerja Amazon VPC.

Langkah 1: Buat dan muat bucket Amazon S3

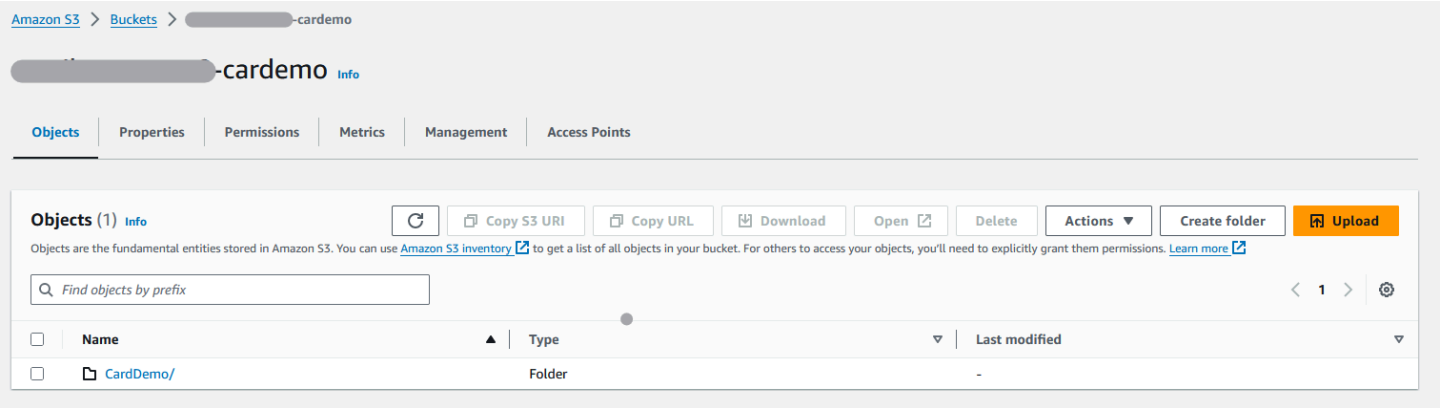
Pada langkah ini, Anda membuat bucket Amazon S3 dan mengunggah CardDemo file ke bucket ini. Kemudian dalam tutorial ini, Anda menggunakan file-file ini untuk menyebarkan dan menjalankan aplikasi CardDemo sampel dalam lingkungan Runtime Terkelola Modernisasi Mikro Fokus AWS Mainframe.

> **Note**
>
> Anda tidak perlu membuat bucket S3 baru tetapi bucket yang Anda pilih harus berada di Region yang sama dengan sumber daya lain yang digunakan dalam tutorial ini.

Untuk membuat bucket Amazon S3

1. Buka konsol Amazon S3, dan pilih Buat bucket.
2. Dalam konfigurasi Umum, pilih Wilayah AWS tempat Anda ingin membangun Modernisasi AWS Mainframe Micro Focus Managed Runtime.
3. Masukkan nama Bucket, misalnya,`yourname-aws-region-carddemo`. Pertahankan pengaturan default, dan pilih Buat ember. Atau, Anda juga dapat menyalin setelan dari bucket Amazon S3 yang sudah ada, lalu pilih Buat bucket.
4. Pilih bucket yang baru saja Anda buat, lalu pilih Unggah.
5. Di bagian Unggah, pilih Tambahkan Folder, lalu telusuri `CardDemo` direktori dari komputer lokal Anda.
6. Pilih Upload untuk memulai proses upload. Waktu upload bervariasi berdasarkan kecepatan koneksi Anda.
7. Ketika unggahan selesai, konfirmasikan bahwa semua file telah berhasil diunggah, lalu pilih Tutup.

Bucket Amazon S3 Anda sekarang berisi folder. `CardDemo`



Untuk informasi tentang bucket S3, lihat Membuat, mengonfigurasi, dan bekerja dengan bucket Amazon S3.

Langkah 2: Buat dan konfigurasikan database

Pada langkah ini, Anda membuat database PostgreSQL di Amazon Relational Database Service (Amazon RDS). Untuk tutorial, database ini berisi kumpulan data yang digunakan aplikasi CardDemo sampel untuk tugas-tugas pelanggan mengenai transaksi kartu kredit.

Untuk membuat database di Amazon RDS

1. Buka konsol Amazon RDS.
2. Pilih Wilayah AWS tempat Anda ingin membuat instance database.
3. Dari panel navigasi, pilih Databases.
4. Pilih Buat database, lalu pilih Standard create.
5. Untuk tipe Engine, pilih PostgreSQL.
6. Pilih versi Engine 15 atau lebih tinggi.

> **Note**
>
> Simpan versi mesin karena Anda membutuhkannya nanti dalam tutorial ini.

7. Di Templat, pilih Tingkat gratis.
8. Ubah pengidentifikasi instans DB menjadi sesuatu yang bermakna, misalnya,`MicroFocus-Tutorial`.

9. Menahan diri dari mengelola kredensi master di. AWS Secrets Manager Sebagai gantinya, masukkan kata sandi utama dan konfirmasikan.

> **Note**
>
> Simpan nama pengguna dan kata sandi yang Anda gunakan untuk database. Anda akan menyimpannya dengan aman di langkah selanjutnya dari tutorial ini.

10. Di bawah Konektivitas, pilih VPC tempat Anda ingin membuat lingkungan runtime terkelola Modernisasi AWS Mainframe.
11. Pilih Buat basis data.

Untuk membuat grup parameter kustom di Amazon RDS

1. Di panel navigasi konsol Amazon RDS, pilih Grup parameter, lalu pilih Buat grup parameter.
2. Di jendela Buat grup parameter, untuk keluarga grup Parameter, pilih opsi Postgres yang cocok dengan versi database Anda.

> **Note**
>
> Beberapa versi Postgres memerlukan Type. Pilih Grup Parameter DB jika diperlukan. Masukkan nama Grup dan Deskripsi untuk grup parameter.

3. Pilih Buat.

Untuk mengkonfigurasi grup parameter kustom

1. Pilih grup parameter yang baru dibuat.
2. Pilih Tindakan, dan kemudian pilih Edit.
3. Filter `max_prepared_transactions` dan ubah nilai parameter menjadi 100.
4. Pilih Simpan Perubahan.

Untuk mengaitkan grup parameter kustom dengan database

1. Di panel navigasi konsol Amazon RDS, pilih Database, lalu pilih instance database yang ingin Anda ubah.

2. Pilih Modifikasi. Halaman Modifikasi instans DB akan muncul.

> **Note**
>
> Opsi Modify tidak tersedia sampai database selesai membuat dan mencadangkan, yang mungkin memakan waktu beberapa menit.

3. Pada halaman Modify DB instans, navigasikan ke Konfigurasi tambahan, dan ubah grup parameter DB ke grup parameter Anda. Jika grup parameter Anda tidak tersedia dalam daftar, periksa apakah itu dibuat dengan versi database yang benar.
4. Pilih Lanjutkan, dan periksa ringkasan modifikasi.
5. Pilih Terapkan segera untuk menerapkan perubahan secara instan.
6. Pilih Modifikasi instans DB untuk menyimpan perubahan Anda.

Untuk informasi selengkapnya tentang grup parameter, lihat Bekerja dengan grup parameter.

> **Note**
>
> Anda juga dapat menggunakan database Amazon Aurora PostgreSQL dengan Modernisasi AWS Mainframe tetapi tidak ada opsi tingkat gratis. Untuk informasi selengkapnya, lihat Bekerja dengan Amazon Aurora PostgreSQL.

Langkah 3: Buat dan konfigurasikan AWS KMS key

Untuk menyimpan kredensil dengan aman untuk instans Amazon RDS, buat dulu file. AWS KMS key

Untuk membuat AWS KMS key

1. Buka konsol Layanan Manajemen Kunci.
2. Pilih Buat Kunci.
3. Biarkan default Symmetric untuk tipe kunci dan Enkripsi dan dekripsi untuk penggunaan kunci.
4. Pilih Berikutnya.
5. Berikan kunci Alias seperti `MicroFocus-Tutorial-RDS-Key` dan deskripsi opsional.
6. Pilih Berikutnya.

7. Tetapkan administrator kunci dengan mencentang kotak di samping pengguna atau peran Anda.
8. Pilih Berikutnya, lalu pilih Berikutnya lagi.
9. Pada layar peninjauan, edit kebijakan Kunci, lalu masukkan yang berikut ini:

```
{
     "Sid" : "Allow access for Mainframe Modernization Service",
     "Effect" : "Allow",
        "Principal" : {
           "Service" : "m2.amazonaws.com"
                 },
      "Action" : "kms:Decrypt",
      "Resource" : "*"
},
```

   Kebijakan ini memberikan izin dekripsi Modernisasi AWS Mainframe menggunakan kebijakan kunci khusus ini.
10. Pilih Selesai untuk membuat kunci.

Untuk informasi selengkapnya, lihat Membuat kunci di Panduan AWS Key Management Service Pengembang.

Langkah 4: Buat dan konfigurasikan rahasia AWS Secrets Manager database

Sekarang simpan kredensi database dengan aman menggunakan dan. AWS Secrets Manager AWS KMS key

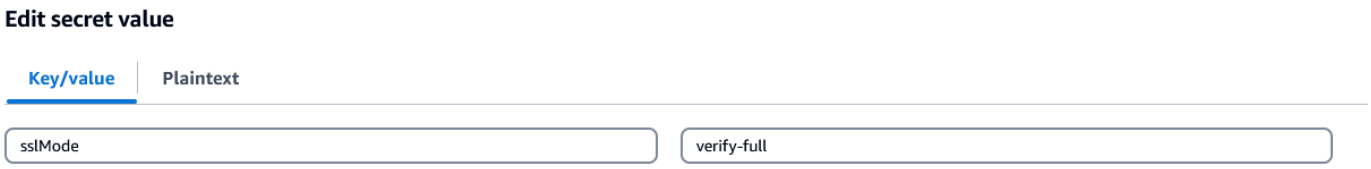
Untuk membuat dan mengkonfigurasi rahasia AWS Secrets Manager database

1. Buka konsol Secrets Manager.
2. Di panel navigasi, pilih Rahasia.
3. Dalam Rahasia, pilih Simpan rahasia baru.
4. Setel jenis Rahasia ke Kredensial untuk database Amazon RDS.
5. Masukkan Credentials yang Anda tentukan saat Anda membuat database.
6. Di bawah kunci Enkripsi, pilih kunci yang Anda buat di langkah 3.
7. Di bagian Database, pilih database yang Anda buat untuk tutorial ini, lalu pilih Berikutnya.

8. Di bawah nama Rahasia, masukkan nama seperti `MicroFocus-Tutorial-RDS-Secret` dan deskripsi opsional.
9. Di bagian Izin sumber daya, pilih Edit izin, dan ganti konten dengan kebijakan berikut:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect" : "Allow",
            "Principal" : {
                "Service" : "m2.amazonaws.com"
            },
            "Action" : "secretsmanager:GetSecretValue",
            "Resource" : "*"
        }
    ]
}
```

10. Pilih Simpan.
11. Pilih Berikutnya untuk layar berikutnya, lalu pilih Store. Segarkan daftar rahasia untuk melihat rahasia baru.
12. Pilih rahasia yang baru dibuat dan perhatikan `Secret ARN` karena Anda membutuhkannya nanti di tutorial.
13. Di tab Ikhtisar rahasia, pilih Ambil nilai rahasia.
14. Pilih Edit, lalu pilih Tambah baris.
15. Tambahkan Kunci untuk `sslMode` dengan Nilai`verify-full`:



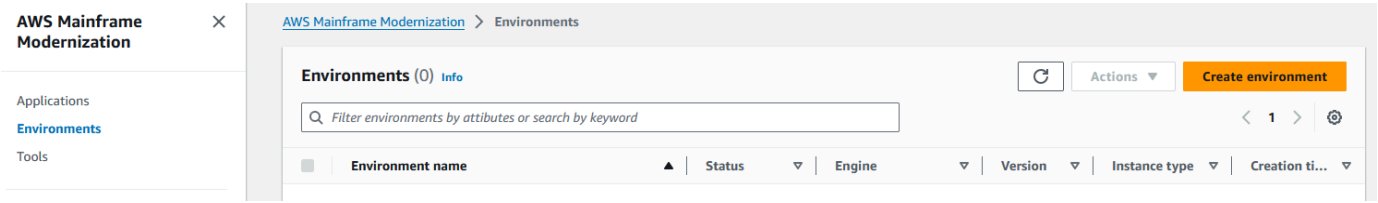
16. Pilih Simpan.

Langkah 5: Buat lingkungan runtime

Untuk membuat lingkungan runtime

1. Buka konsol Modernisasi AWS Mainframe.

2. Pada panel navigasi, pilih Lingkungan. Kemudian pilih Buat lingkungan.



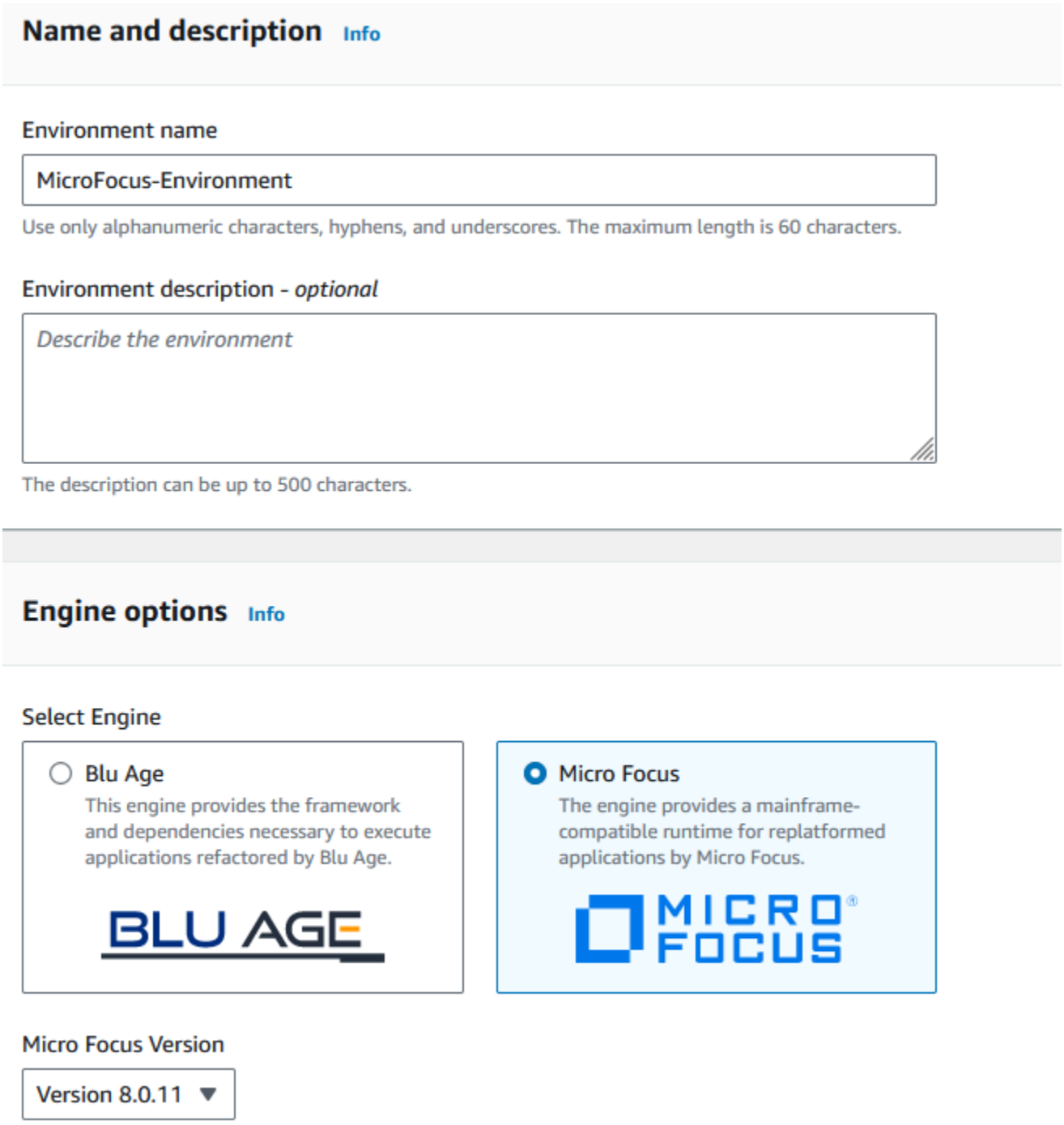
3. Di bawah Tentukan informasi dasar,

   a. Masukkan `MicroFocus-Environment` untuk nama lingkungan.

   b. Di bawah opsi engine, pastikan Micro Focus dipilih.

   c. Pilih Versi Fokus Mikro terbaru.

   d. Pilih Berikutnya.



4. Konfigurasikan lingkungan

   a. Di bawah Ketersediaan, pilih Cluster ketersediaan tinggi.

   b. Di bawah Sumber Daya, pilih salah satu `M2.c5.large` atau `M2.m5.large` untuk jenis instans, dan jumlah instance yang Anda inginkan. Tentukan hingga dua contoh.

c. Di bawah Keamanan dan jaringan, pilih Izinkan aplikasi yang disebarkan ke lingkungan ini agar dapat diakses publik dan pilih setidaknya dua subnet publik.

d. Pilih Berikutnya.

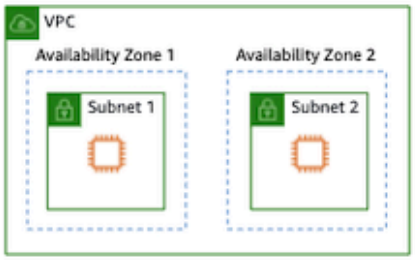
Specify configurations Info

Availability Info

Choose the availability pattern for your environment.

- Standalone runtime environment
  Sets up a single instance in a single availability zone. Does not guarantee high availability but costs less.

- High availability cluster
  Sets up redundant instances across two availability zones. Enables higher availability but costs more.



Resources

Instance type
Choose the instance type for your high availability cluster.

M2.m5.large

Desired capacity
Specify the desired number of instances.

2

Security and network

Allow applications deployed to this environment to be publicly accessible.

Virtual Private Cloud (VPC)
Choose the VPC where you want to create the environment.

Default vpc-15

Subnets
Choose one or more subnets for a high availability setup.

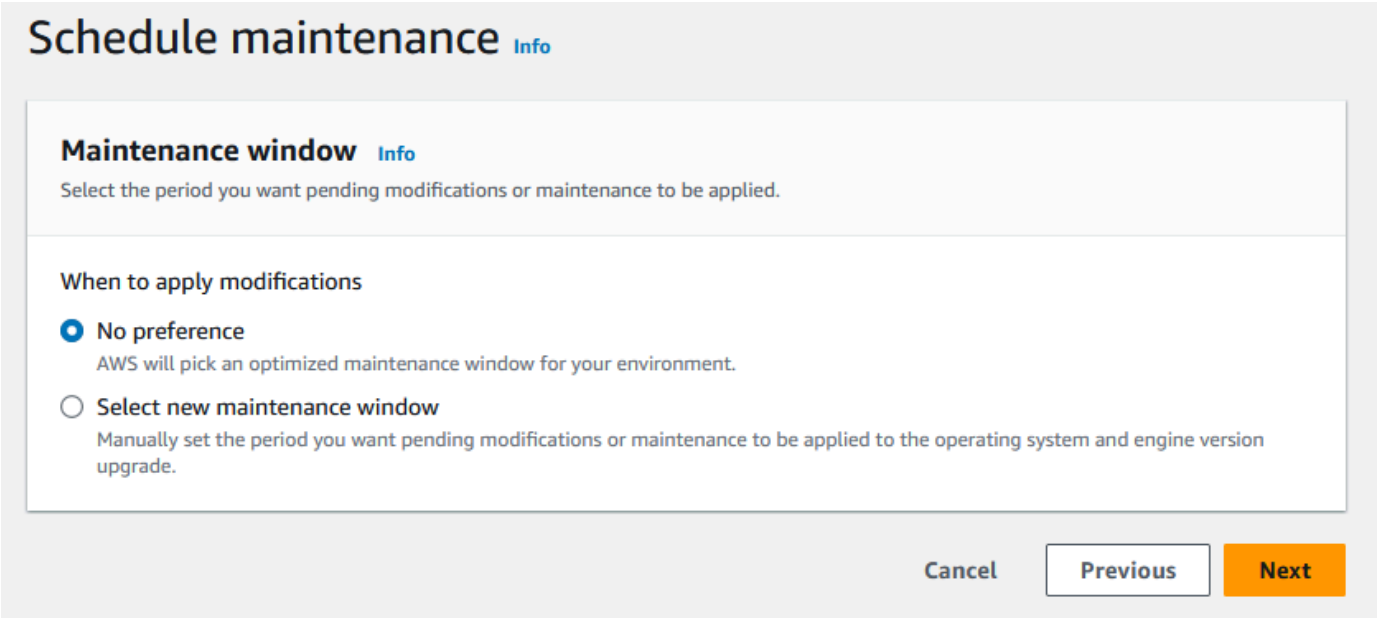
Choose subnets

subnet-56f1a | us-west-2a

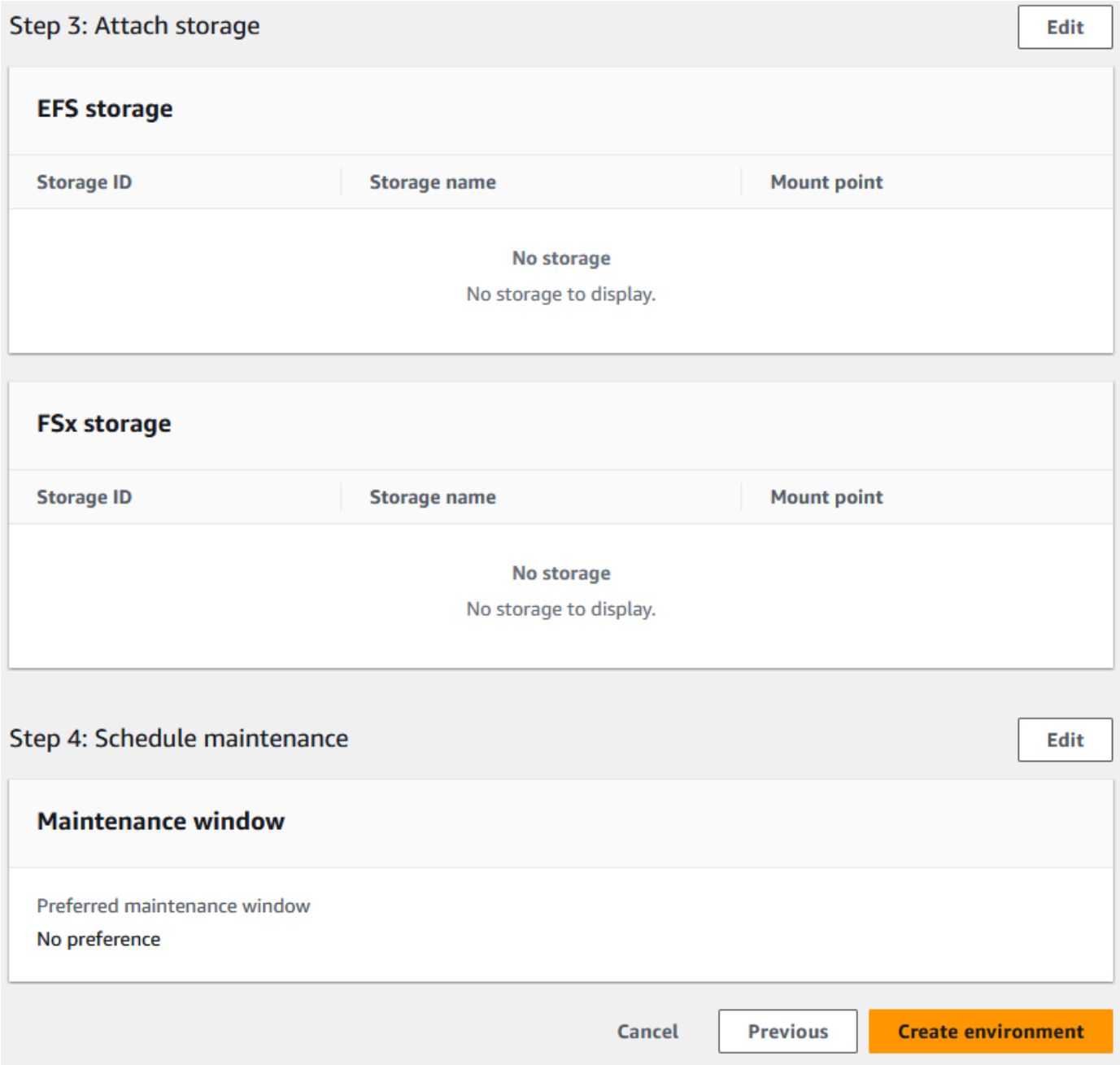
subnet-6685 | us-west-2b

Security groups
Choose one or more security groups for the chosen VPC.

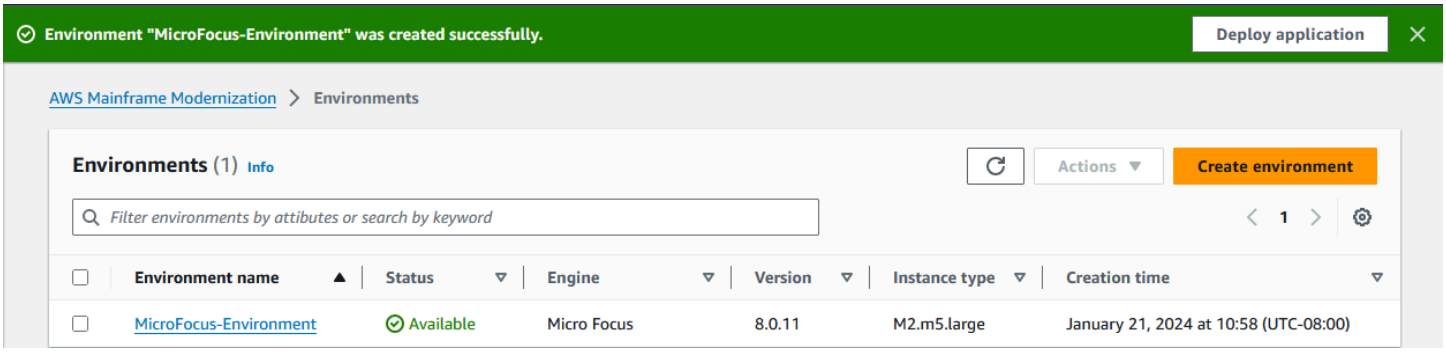
5. Pada halaman Lampirkan penyimpanan, pilih Berikutnya.
6. Pada halaman Jadwal pemeliharaan, pilih Tidak ada preferensi dan kemudian pilih Berikutnya.



7. Pada halaman Tinjau dan buat, tinjau semua konfigurasi yang Anda berikan untuk lingkungan runtime, lalu pilih Buat lingkungan.



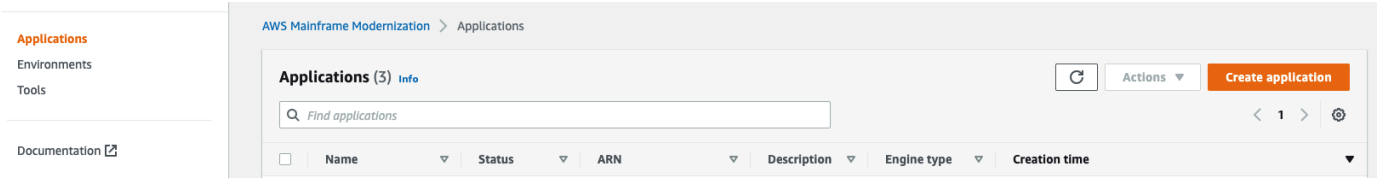
Saat Anda membuat lingkungan, spanduk muncul yang bertuliskan`Environment` *`name`* `was created successfully`, dan bidang Status berubah menjadi Tersedia. Proses pembuatan lingkungan memakan waktu beberapa menit tetapi Anda dapat melanjutkan langkah selanjutnya saat berjalan.



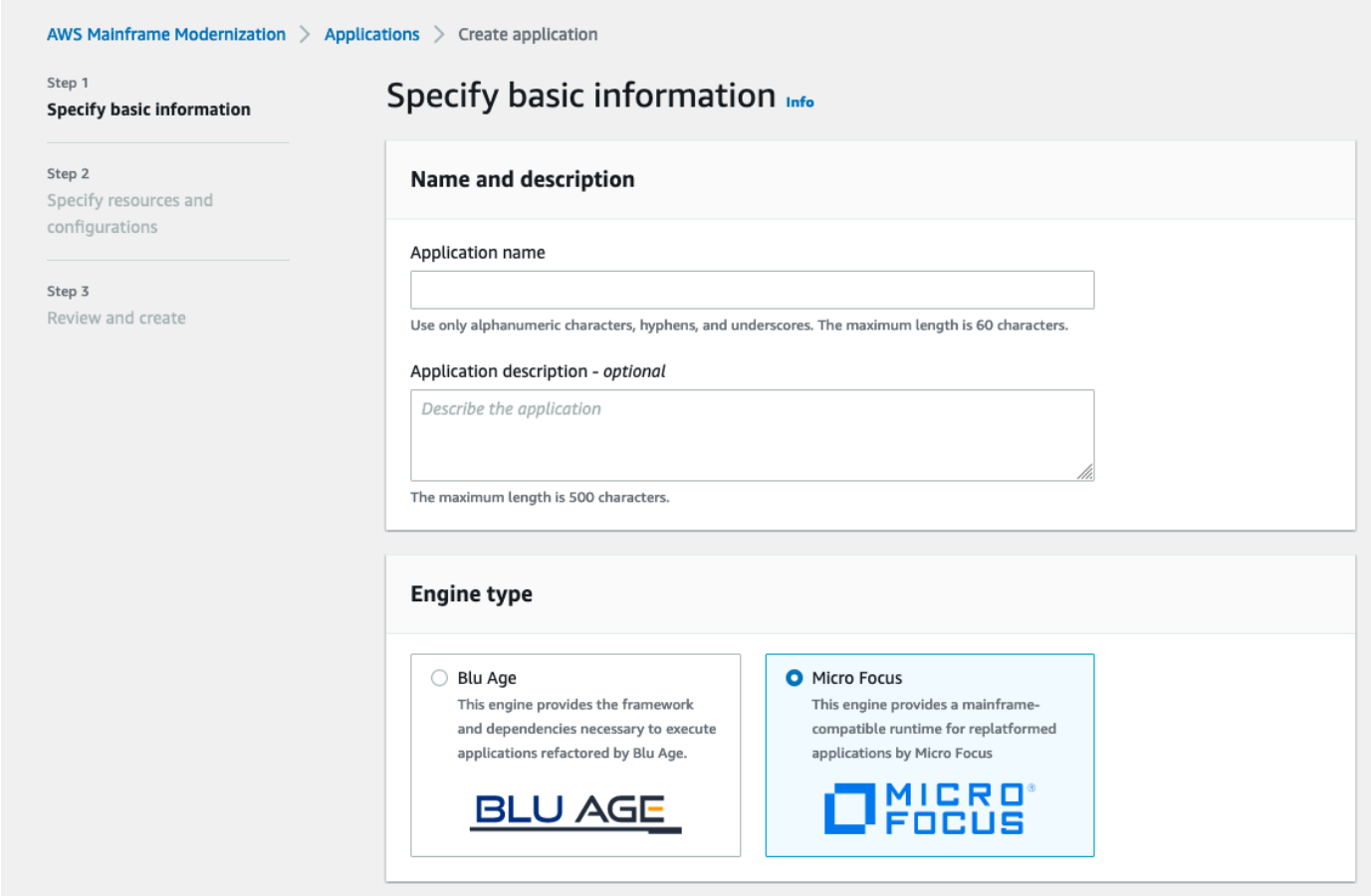
Langkah 6: Buat aplikasi

Untuk membuat aplikasi

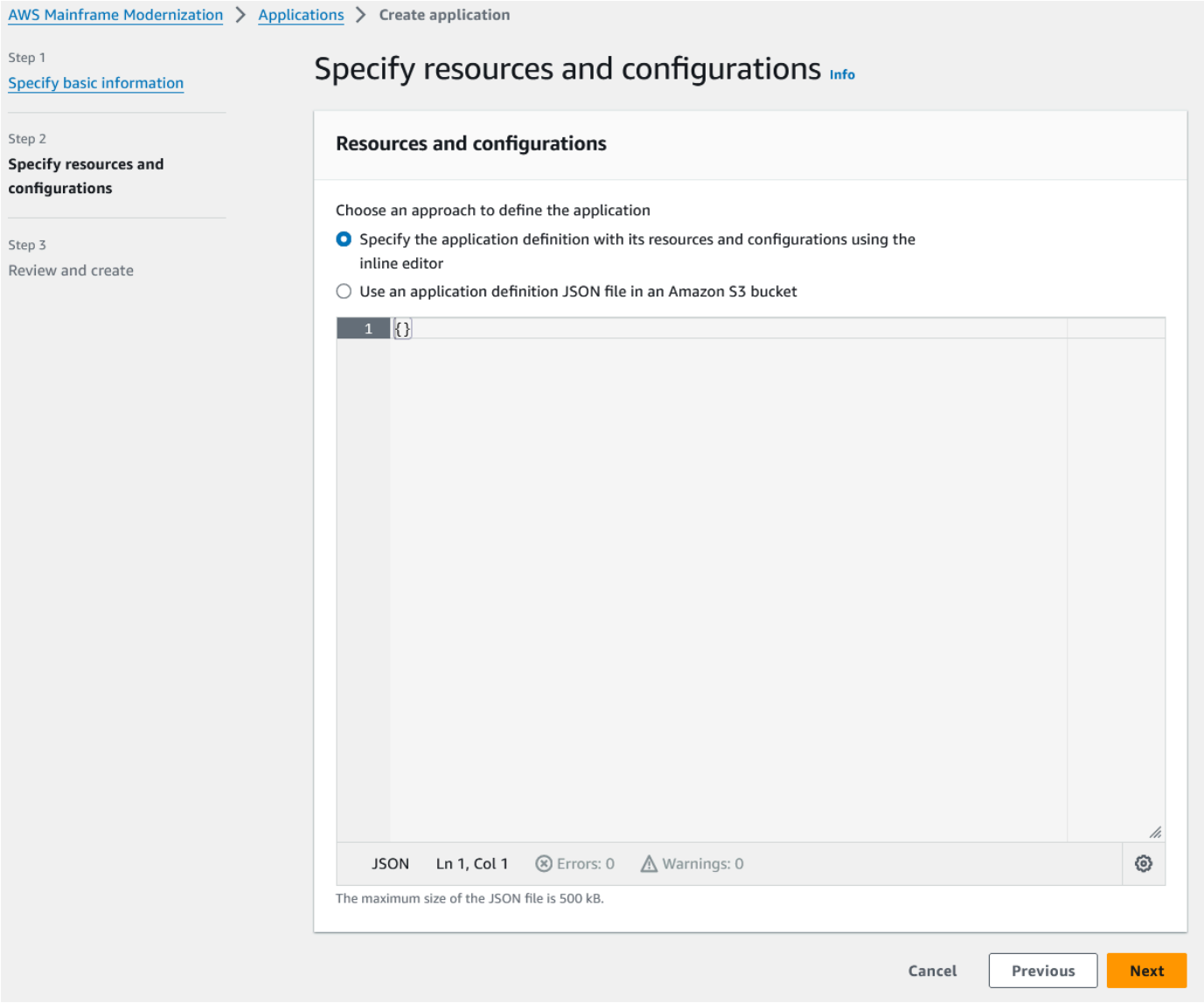
1. Pada panel navigasi, pilih Aplikasi. Kemudian pilih Buat aplikasi.



2. Pada halaman Buat aplikasi, di bawah Tentukan informasi dasar, masukkan `MicroFocus-CardDemo` untuk nama aplikasi dan di bawah Jenis mesin pastikan Micro Focus dipilih. Lalu pilih Selanjutnya.



3. Di bawah Tentukan sumber daya dan konfigurasi, pilih opsi untuk menentukan definisi aplikasi dengan sumber daya dan konfigurasinya menggunakan editor sebaris.



Masukkan definisi aplikasi berikut di editor:

```
{
  "template-version": "2.0",
  "source-locations": [
    {
      "source-id": "s3-source",
      "source-type": "s3",
      "properties": {
        "s3-bucket": "yourname-aws-region-carddemo",
        "s3-key-prefix": "CardDemo"
      }
    }
```

```
  ],
  "definition": {
    "listeners": [
      {
        "port": 6000,
        "type": "tn3270"
      }
    ],
    "dataset-location": {
      "db-locations": [
        {
          "name": "Database1",
          "secret-manager-arn":
 "arn:aws:secretsmanager:Region:123456789012:secret:MicroFocus-Tutorial-RDS-Secret-
xxxxxx"
        }
      ]
    },
    "batch-settings": {
      "initiators": [
        {
          "classes": [
               "A",
               "B"
            ],
          "description": "initiator_AB...."
        },
        {
          "classes": [
                "C",
                "D"
             ],
          "description": "initiator_CD...."
        }
      ],
      "jcl-file-location": "${s3-source}/catalog/jcl"
    },
    "cics-settings": {
      "binary-file-location": "${s3-source}/loadlib",
      "csd-file-location": "${s3-source}/rdef",
      "system-initialization-table": "CARDSIT"
    },
    "xa-resources": [
      {
```

```
        "name": "XASQL",
        "secret-manager-arn":
 "arn:aws:secretsmanager:Region:123456789012:secret:MicroFocus-Tutorial-RDS-Secret-
xxxxxx",
        "module": "${s3-source}/xa/ESPGSQLXA64.so"
      }
    ]
  }
}
```

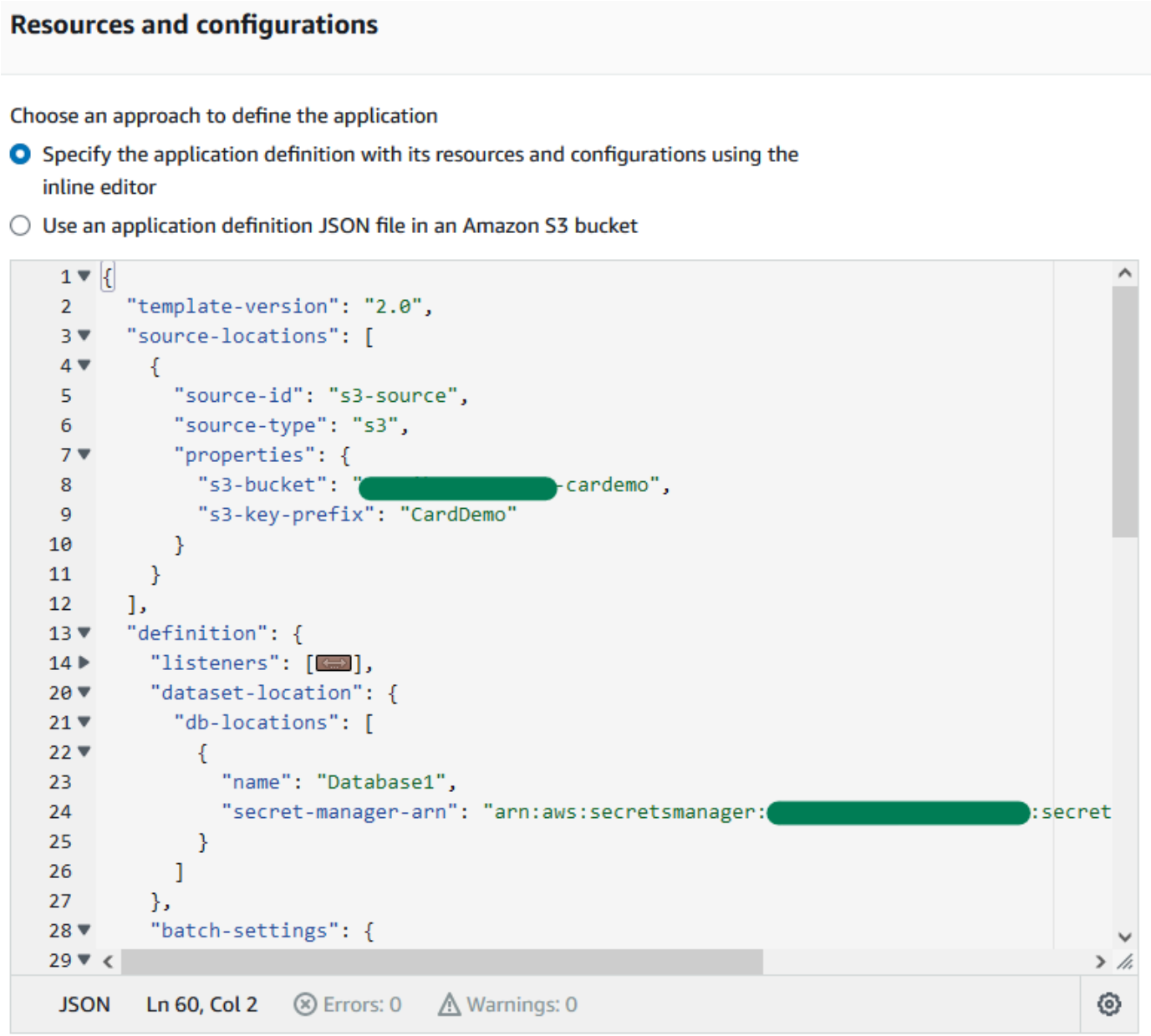
> **Note**
>
> File ini dapat berubah sewaktu-waktu.

4. Edit aplikasi JSON di objek properti sumber-lokasi sebagai berikut:

   a. Ganti nilainya `s3_bucket` dengan nama bucket Amazon S3 yang Anda buat di Langkah 1.

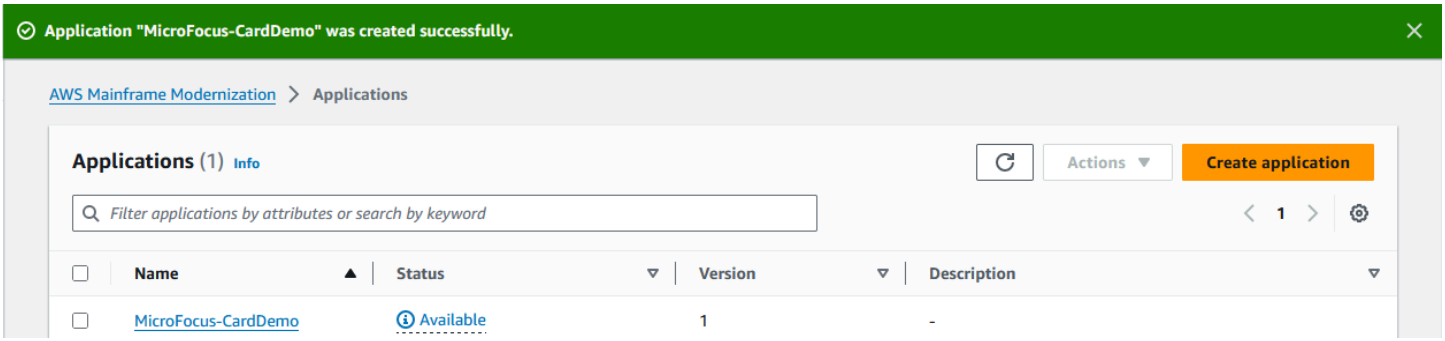
   b. Ganti nilai untuk `s3-key-prefix` dengan folder (key prefix) tempat Anda mengunggah file sampel. CardDemo Jika Anda mengunggah `CardDemo` direktori langsung ke bucket Amazon S3, maka `s3-key-prefix` tidak perlu diubah.

   c. Ganti kedua `secret-manager-arn` nilai dengan ARN untuk rahasia database yang Anda buat di Langkah 4.



Untuk informasi lebih lanjut tentang definisi aplikasi, lihatDefinisi aplikasi Fokus Mikro.

5. Pilih Next untuk melanjutkan.
6. Pada halaman Tinjau dan buat, tinjau informasi yang Anda berikan, lalu pilih Buat aplikasi.

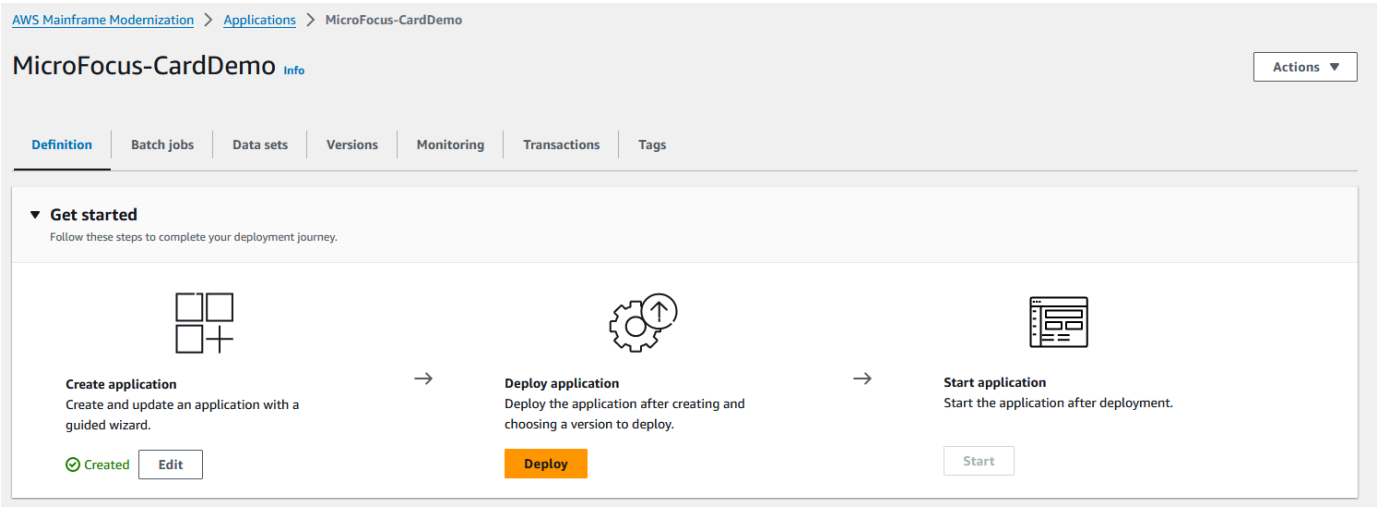


Ketika Anda telah membuat aplikasi Anda, sebuah spanduk muncul yang mengatakan`Application` *`name`* `was created successfully`. Dan bidang Status berubah menjadi Tersedia.

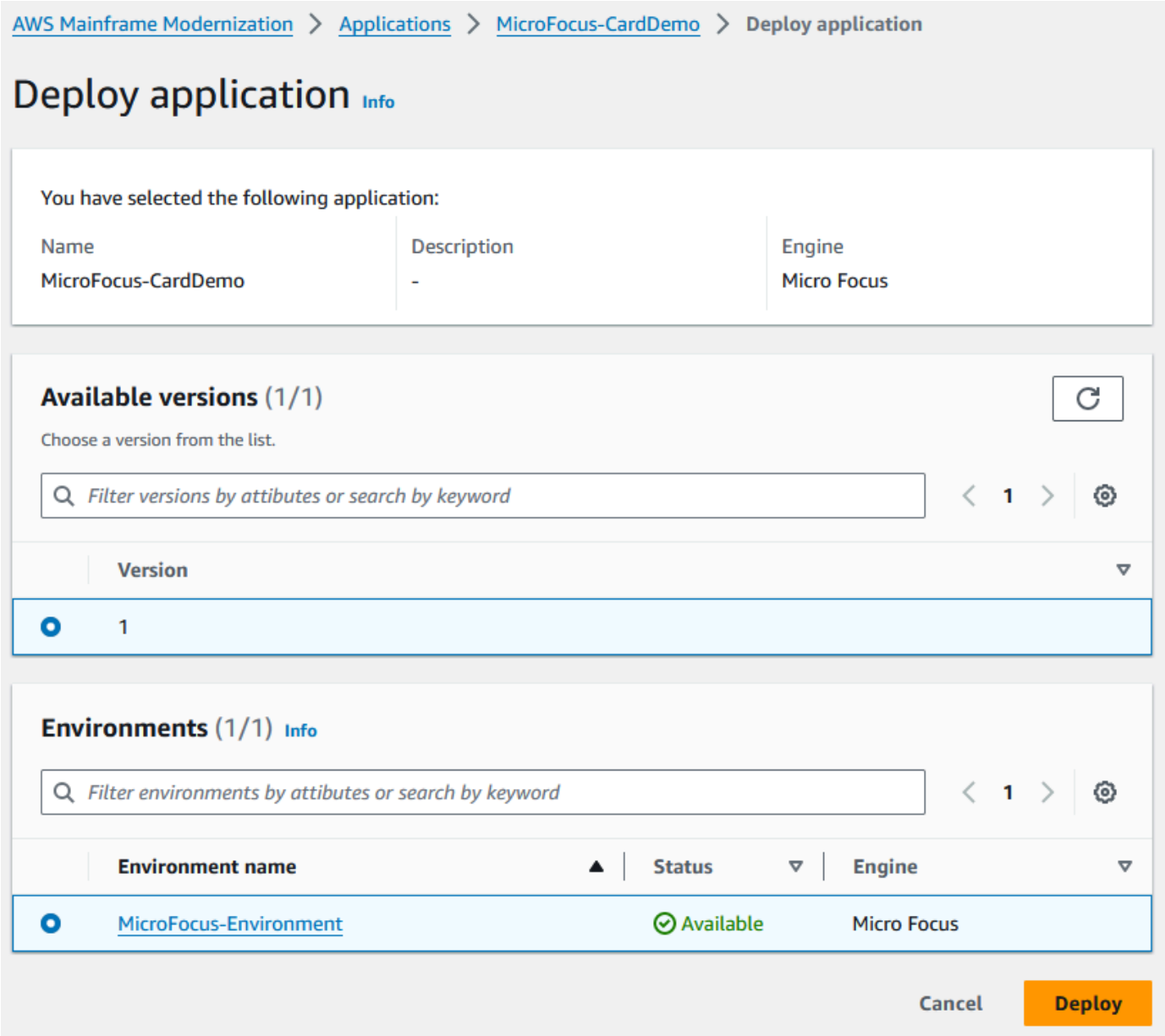
Langkah 7: Menyebarkan aplikasi

Untuk menyebarkan aplikasi

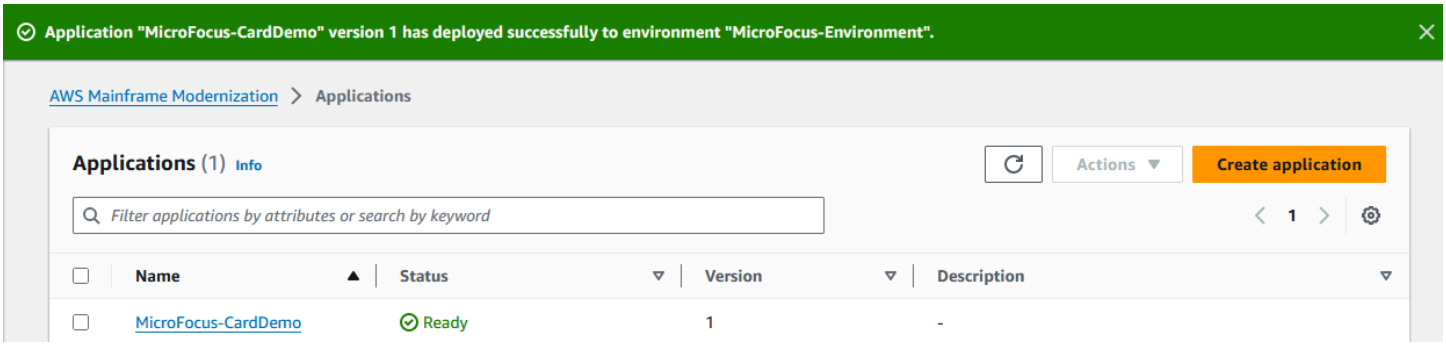
1. Di panel navigasi, pilih Aplikasi, lalu pilih`MicroFocus-CardDemo`.
2. Di bawah Menyebarkan aplikasi, pilih Deploy.



3. Pilih versi terbaru dari aplikasi dan lingkungan yang Anda buat sebelumnya, lalu pilih Deploy.



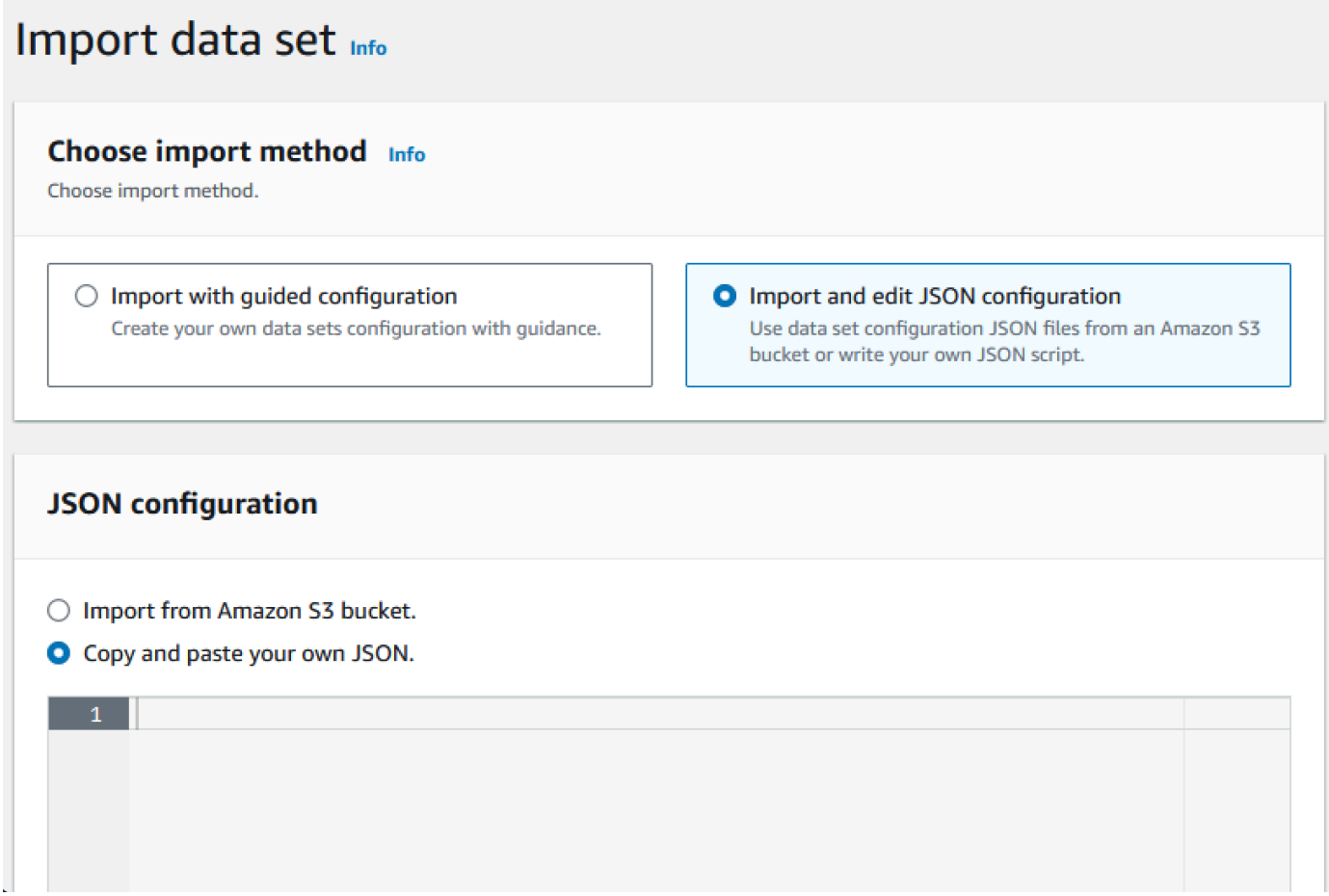
Ketika CardDemo aplikasi berhasil digunakan, status berubah menjadi Siap.



Langkah 8: Impor set data

Untuk mengimpor set data

1. Di panel navigasi, pilih Aplikasi, lalu pilih aplikasi.
2. Pilih tab Set data. Kemudian pilih Impor.
3. Pilih Impor dan Edit konfigurasi JSON, lalu pilih opsi Salin dan tempel JSON Anda sendiri.



4. Salin dan tempel JSON berikut tetapi jangan memilih “Kirim”. JSON ini berisi semua kumpulan data yang diperlukan untuk aplikasi demo tetapi membutuhkan detail bucket Amazon S3 Anda.

```
{
    "dataSets": [
        {
            "dataSet": {
                "storageType": "Database",
                "datasetName": "AWS.M2.CARDDEMO.ACCTDATA.VSAM.KSDS",
                "relativePath": "DATA",
                "datasetOrg": {
```

```
                    "vsam": {
                        "format": "KS",
                        "encoding": "A",
                        "primaryKey": {
                            "length": 11,
                            "offset": 0
                        }
                    }
                },
                "recordLength": {
                    "min": 300,
                    "max": 300
                }
            },
            "externalLocation": {
                "s3Location": "s3://<s3-bucket-name>/CardDemo/catalog/data/
AWS.M2.CARDDEMO.ACCTDATA.VSAM.KSDS.DAT"
            }
        },
        {
            "dataSet": {
                "storageType": "Database",
                "datasetName": "AWS.M2.CARDDEMO.CARDDATA.VSAM.AIX.PATH",
                "relativePath": "DATA",
                "datasetOrg": {
                    "vsam": {
                        "format": "KS",
                        "encoding": "A",
                        "primaryKey": {
                            "length": 11,
                            "offset": 16
                        }
                    }
                },
                "recordLength": {
                    "min": 150,
                    "max": 150
                }
            },
            "externalLocation": {
                "s3Location": "s3://<s3-bucket-name>/CardDemo/catalog/data/
AWS.M2.CARDDEMO.CARDDATA.VSAM.KSDS.DAT"
            }
        },
```

```
        {
            "dataSet": {
                "storageType": "Database",
                "datasetName": "AWS.M2.CARDDEMO.CARDDATA.VSAM.KSDS",
                "relativePath": "DATA",
                "datasetOrg": {
                    "vsam": {
                        "format": "KS",
                        "encoding": "A",
                        "primaryKey": {
                            "length": 16,
                            "offset": 0
                        }
                    }
                },
                "recordLength": {
                    "min": 150,
                    "max": 150
                }
            },
            "externalLocation": {
                "s3Location": "s3://<s3-bucket-name>/CardDemo/catalog/data/
AWS.M2.CARDDEMO.CARDDATA.VSAM.KSDS.DAT"
            }
        },
        {
            "dataSet": {
                "storageType": "Database",
                "datasetName": "AWS.M2.CARDDEMO.CARDXREF.VSAM.KSDS",
                "relativePath": "DATA",
                "datasetOrg": {
                    "vsam": {
                        "format": "KS",
                        "encoding": "A",
                        "primaryKey": {
                            "length": 16,
                            "offset": 0
                        }
                    }
                },
                "recordLength": {
                    "min": 50,
                    "max": 50
                }
```

```
            },
            "externalLocation": {
                "s3Location": "s3://<s3-bucket-name>/CardDemo/catalog/data/
AWS.M2.CARDDEMO.CARDXREF.VSAM.KSDS.DAT"
            }
        },
        {
            "dataSet": {
                "storageType": "Database",
                "datasetName": "AWS.M2.CARDDEMO.CUSTDATA.VSAM.KSDS",
                "relativePath": "DATA",
                "datasetOrg": {
                    "vsam": {
                        "format": "KS",
                        "encoding": "A",
                        "primaryKey": {
                            "length": 9,
                            "offset": 0
                        }
                    }
                },
                "recordLength": {
                    "min": 500,
                    "max": 500
                }
            },
            "externalLocation": {
                "s3Location": "s3://<s3-bucket-name>/CardDemo/catalog/data/
AWS.M2.CARDDEMO.CUSTDATA.VSAM.KSDS.DAT"
            }
        },
        {
            "dataSet": {
                "storageType": "Database",
                "datasetName": "AWS.M2.CARDDEMO.CARDXREF.VSAM.AIX.PATH",
                "relativePath": "DATA",
                "datasetOrg": {
                    "vsam": {
                        "format": "KS",
                        "encoding": "A",
                        "primaryKey": {
                            "length": 11,
                            "offset": 25
                        }
```

```
                    }
                },
                "recordLength": {
                    "min": 50,
                    "max": 50
                }
            },
            "externalLocation": {
                "s3Location": "s3://<s3-bucket-name>/CardDemo/catalog/data/
AWS.M2.CARDDEMO.CARDXREF.VSAM.KSDS.DAT"
            }
        },
        {
            "dataSet": {
                "storageType": "Database",
                "datasetName": "AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS",
                "relativePath": "DATA",
                "datasetOrg": {
                    "vsam": {
                        "format": "KS",
                        "encoding": "A",
                        "primaryKey": {
                            "length": 16,
                            "offset": 0
                        }
                    }
                },
                "recordLength": {
                    "min": 350,
                    "max": 350
                }
            },
            "externalLocation": {
                "s3Location": "s3://<s3-bucket-name>/CardDemo/catalog/data/
AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS.DAT"
            }
        },
        {
            "dataSet": {
                "storageType": "Database",
                "datasetName": "AWS.M2.CARDDEMO.USRSEC.VSAM.KSDS",
                "relativePath": "DATA",
                "datasetOrg": {
                    "vsam": {
```

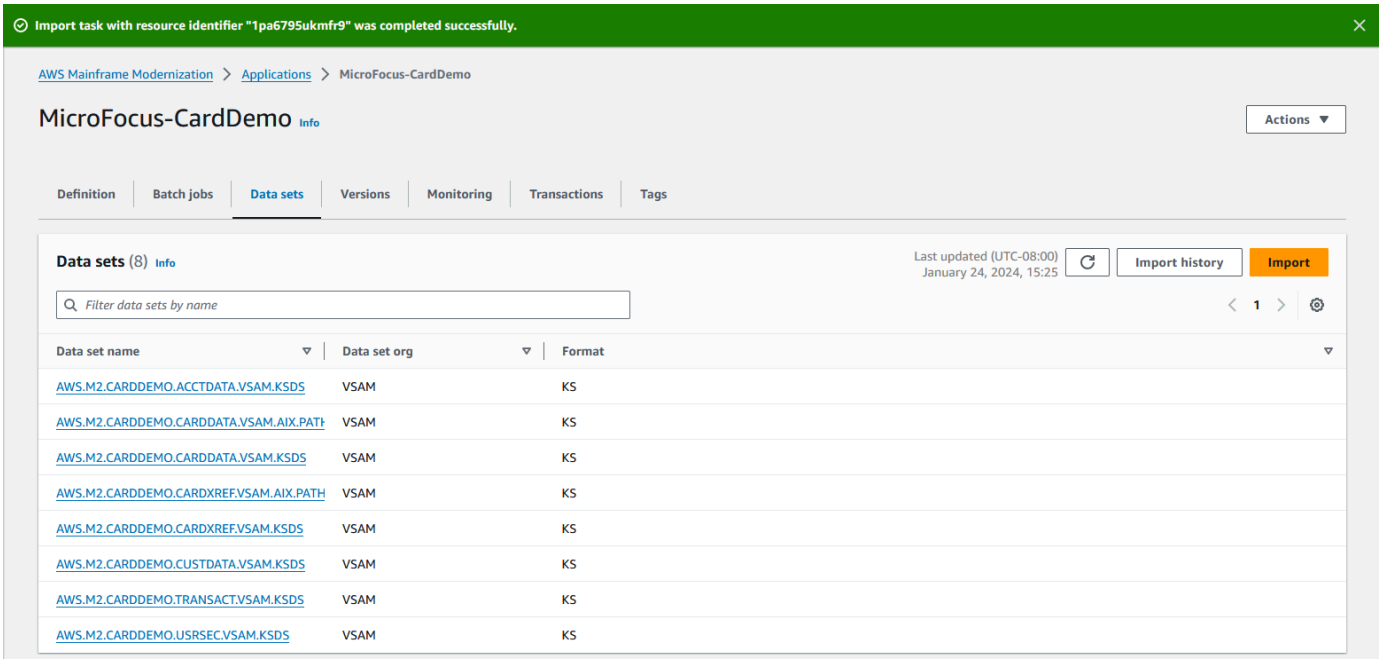
```
                        "format": "KS",
                        "encoding": "A",
                        "primaryKey": {
                            "length": 8,
                            "offset": 0
                        }
                    }
                },
                "recordLength": {
                    "min": 80,
                    "max": 80
                }
            },
            "externalLocation": {
                "s3Location": "s3://<s3-bucket-name>/CardDemo/catalog/data/
AWS.M2.CARDDEMO.USRSEC.VSAM.KSDS.DAT"
            }
        }
    ]
}
```

5. Ganti setiap kemunculan `<s3-bucket-name>` (ada delapan) dengan nama bucket Amazon S3 yang berisi CardDemo folder, misalnya,. `your-name-aws-region-carddemo`

> **Note**
>
> Untuk menyalin URI Amazon S3 untuk folder di Amazon S3, pilih folder, lalu pilih Salin Amazon S3 URI.

6. Pilih Kirim.

   Saat impor selesai, spanduk muncul dengan pesan berikut: `Import task with resource identifier` *`name`* `was completed successfully.` Daftar kumpulan data yang diimpor ditampilkan.

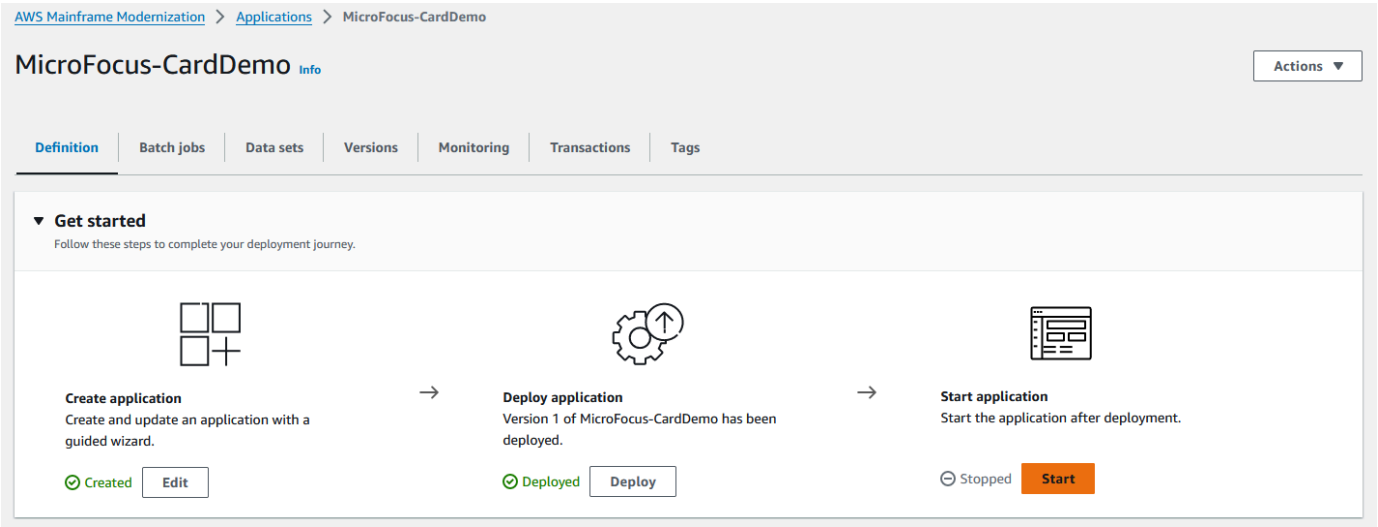


Anda juga dapat melihat status semua impor kumpulan data dengan memilih Impor Riwayat pada tab Set data.

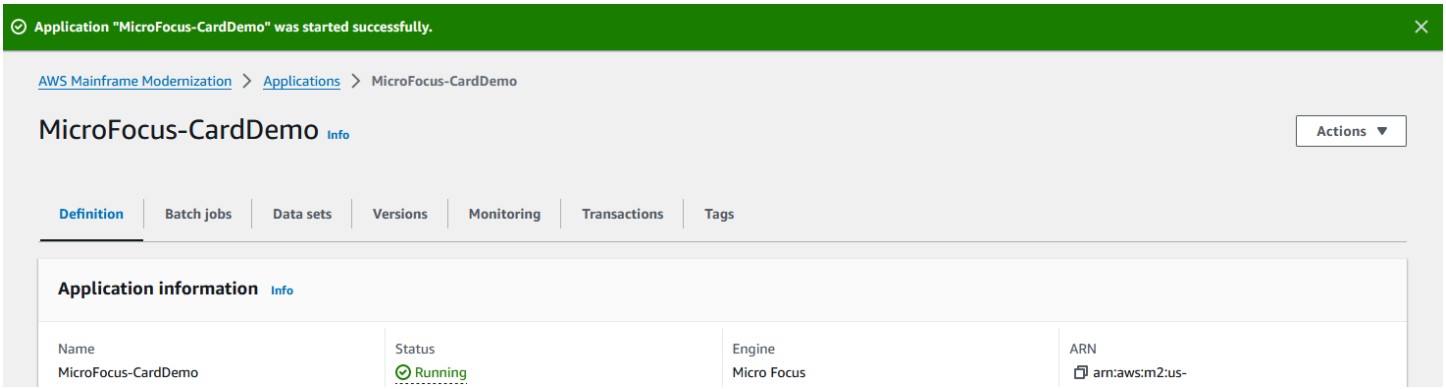
Langkah 9: Mulai aplikasi

Untuk memulai aplikasi

1. Di panel navigasi, pilih Aplikasi, lalu pilih aplikasi.
2. Pilih Mulai aplikasi.



Ketika CardDemo aplikasi mulai berjalan dengan sukses, sebuah spanduk muncul dengan pesan berikut: `Application` *`name`* `was started successfully` Bidang Status berubah menjadi Running.



Langkah 10: Connect ke aplikasi CardDemo CICS

Sebelum Anda terhubung, pastikan bahwa VPC dan grup keamanan yang Anda tentukan untuk aplikasi sama dengan yang Anda terapkan untuk antarmuka jaringan yang akan Anda sambungkan.

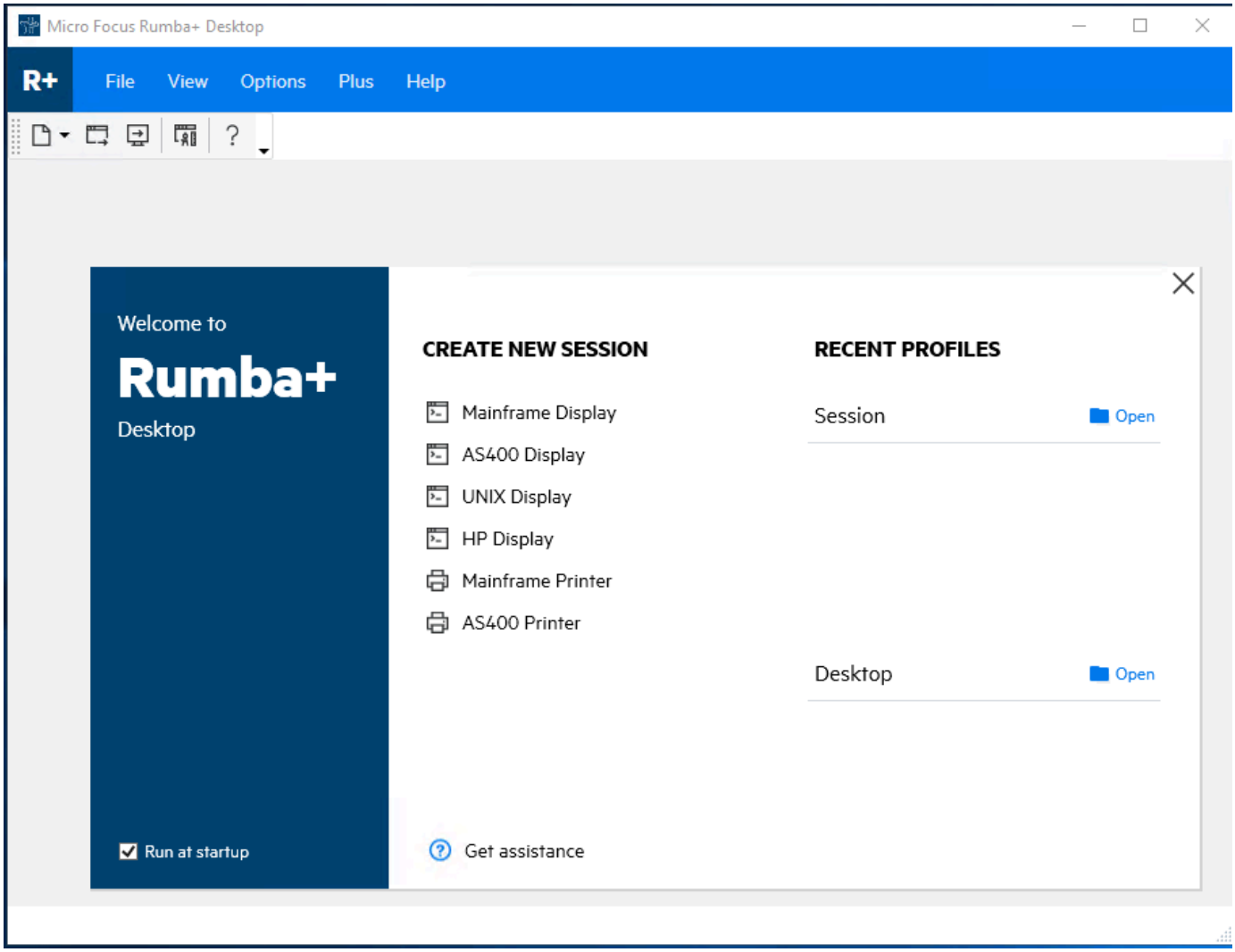
Untuk mengkonfigurasi koneksi TN3270, Anda juga memerlukan nama host DNS dan port aplikasi.

Untuk mengkonfigurasi dan menghubungkan aplikasi ke mainframe menggunakan emulator terminal

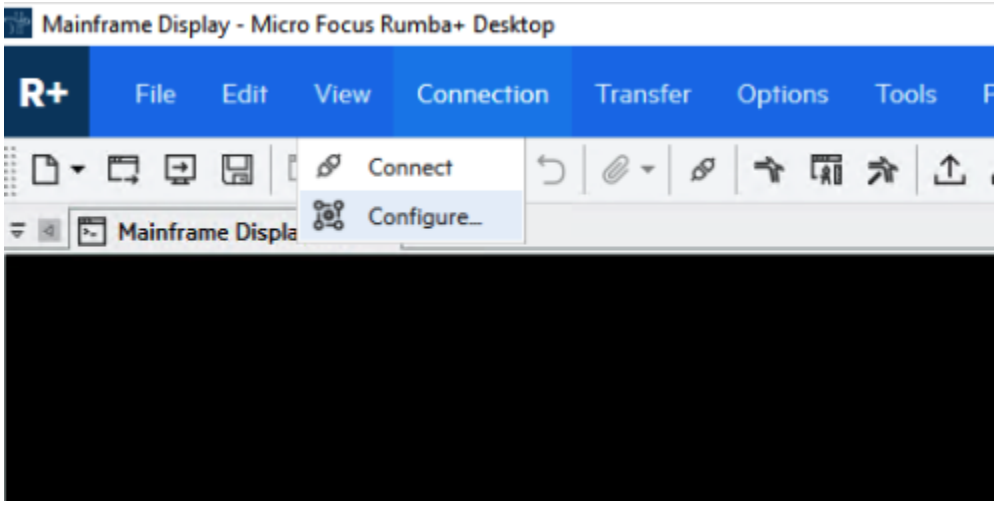
1. Buka konsol Modernisasi AWS Mainframe dan pilih Aplikasi, lalu pilih. `MicroFocus-CardDemo`
2. Pilih ikon salin untuk menyalin Nama Host DNS. Pastikan juga untuk mencatat nomor Port.
3. Mulai emulator terminal. Tutorial ini menggunakan Micro Focus Rumba+.

> **Note**
>
> Langkah-langkah konfigurasi bervariasi menurut emulator.

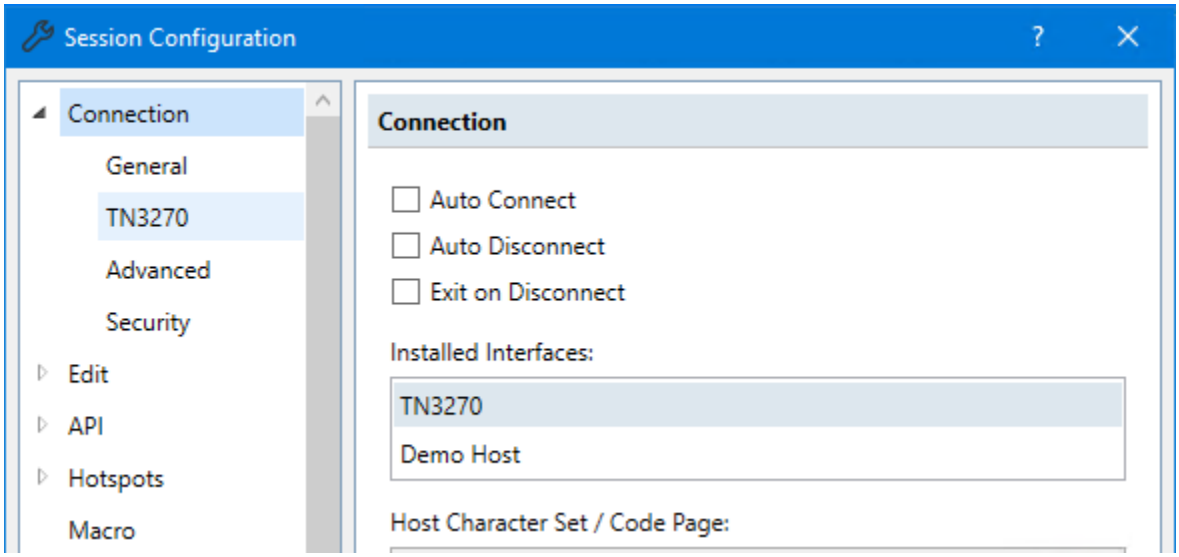
4. Pilih Tampilan Mainframe.



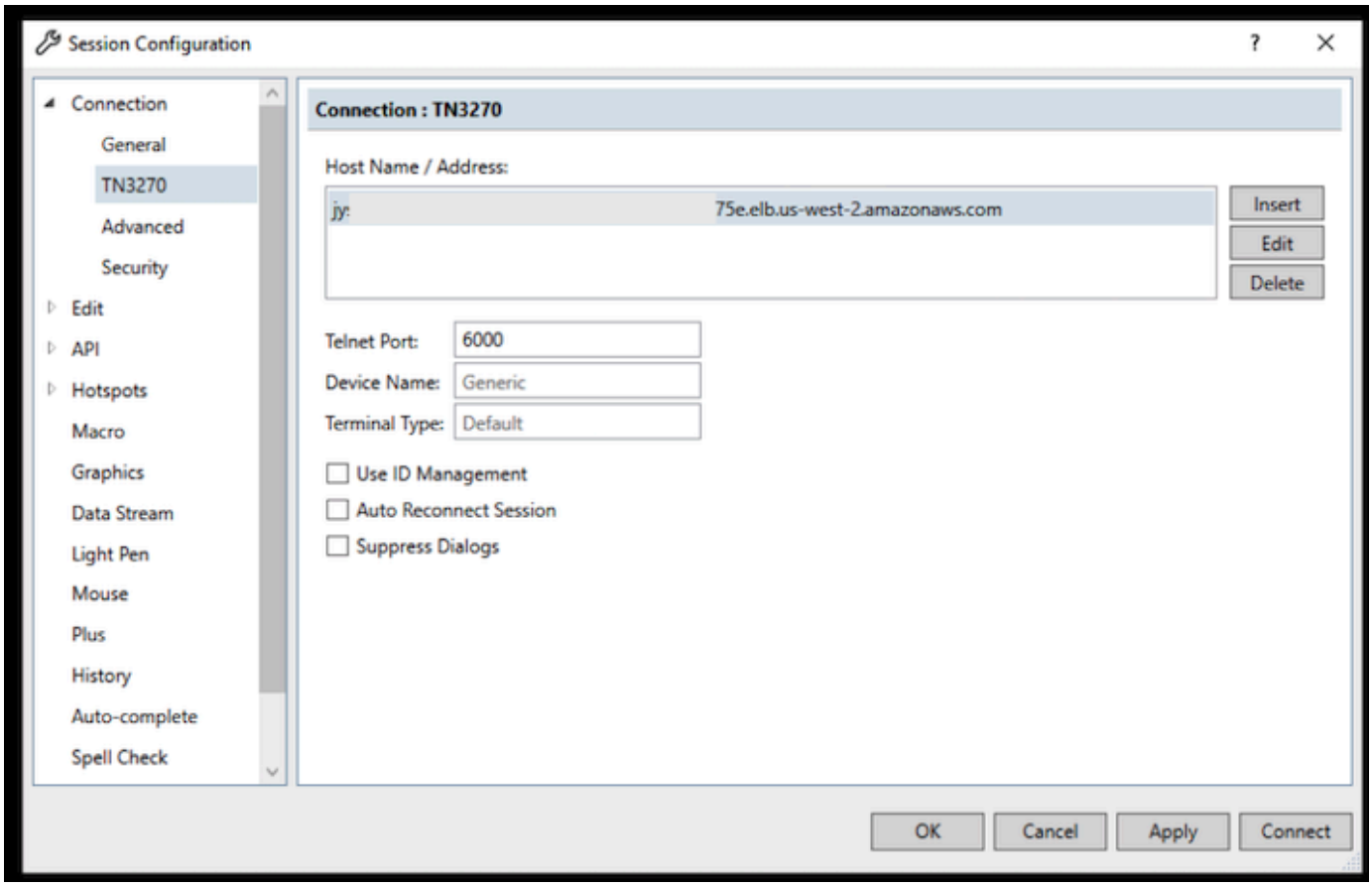
5. Pilih Koneksi, lalu pilih Konfigurasi.



6. Di bawah Antarmuka yang Dipasang, pilih`TN3270`, dan kemudian pilih `TN3270` lagi di bawah menu Koneksi.



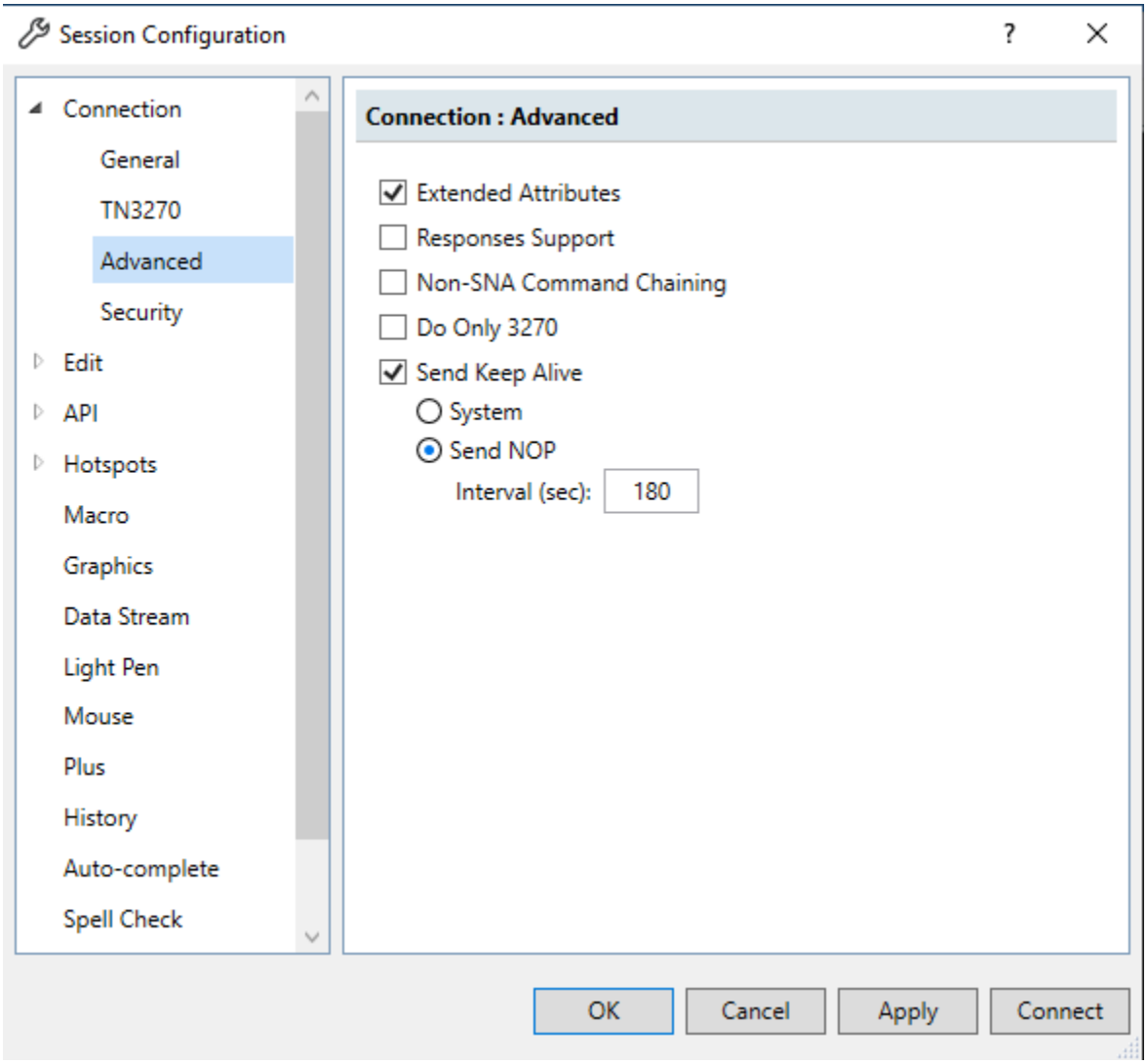
7. Pilih Sisipkan, dan tempel `DNS Hostname` untuk Aplikasi. Tentukan `6000` untuk Port Telnet.



> **Note**
>
> Jika Anda menggunakan AWS AppStream 2.0 di browser dan mengalami kesulitan dalam menempelkan nilai, silakan lihat Pemecahan Masalah Pengguna AppStream 2.0.

8. Di bawah Connection, pilih Advanced, lalu pilih Send Keep Alive dan Send NOP, dan masukkan 180 untuk Interval.

> **Note**
>
> Mengonfigurasi pengaturan keep alive pada terminal TN3270 Anda setidaknya 180 detik membantu memastikan bahwa Network Load Balancer tidak menghentikan koneksi Anda.



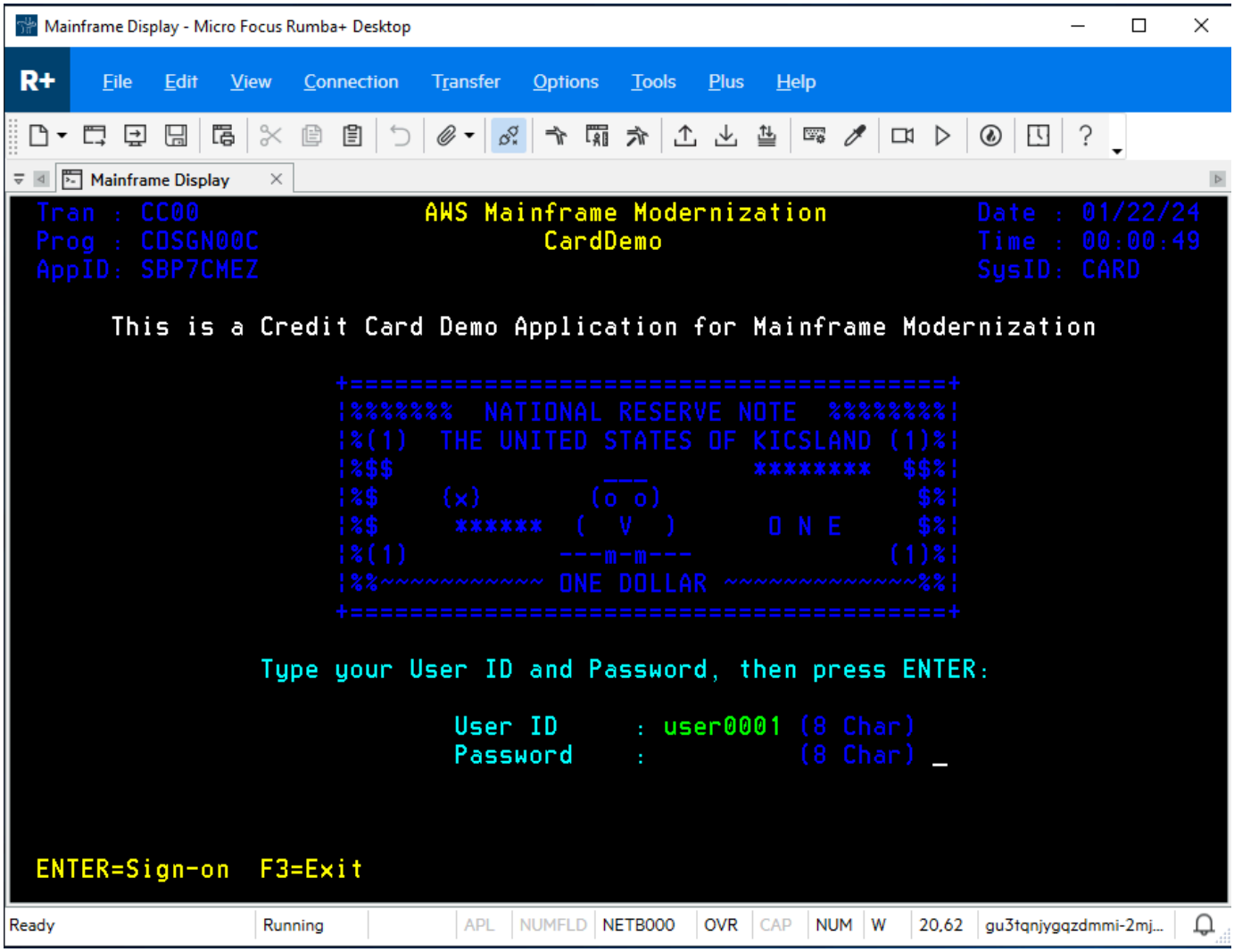
9. Pilih Hubungkan.

> **Note**
>
> Jika koneksi gagal:
>
> - Jika Anda menggunakan AppStream 2.0, konfirmasikan bahwa VPC dan grup keamanan yang ditentukan untuk lingkungan aplikasi sama dengan armada 2.0. AppStream
> - Gunakan VPC Reachability Analyzer untuk menganalisis koneksi. Anda dapat mengakses Reachability Analyzer melalui konsol.

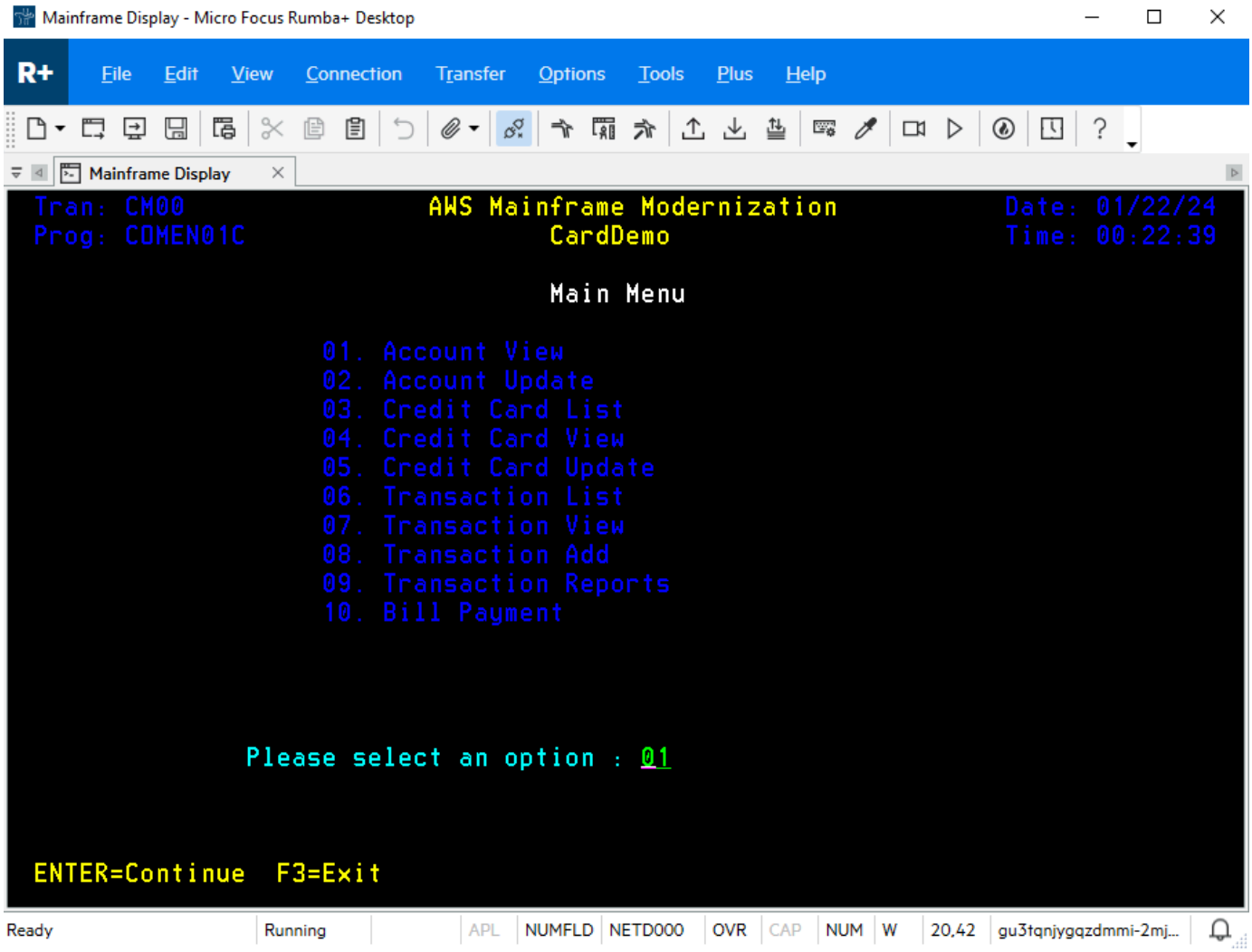
- Sebagai langkah diagnostik, coba tambahkan atau ubah aturan masuk Grup Keamanan untuk aplikasi untuk memungkinkan lalu lintas untuk port 6000 dari mana saja (yaitu CIDR Block 0.0.0.0/0). Jika Anda berhasil terhubung, maka Anda tahu grup keamanan memblokir lalu lintas Anda. Ubah sumber grup keamanan menjadi sesuatu yang lebih spesifik. Untuk informasi selengkapnya tentang grup keamanan, lihat Dasar-dasar grup keamanan.

10. Masukkan `USER0001` nama pengguna dan `password` kata sandi.

> **Note**
>
> Di Rumba, default untuk Clear adalah ctrl-shift-z, dan default untuk Reset adalah ctrl-r.

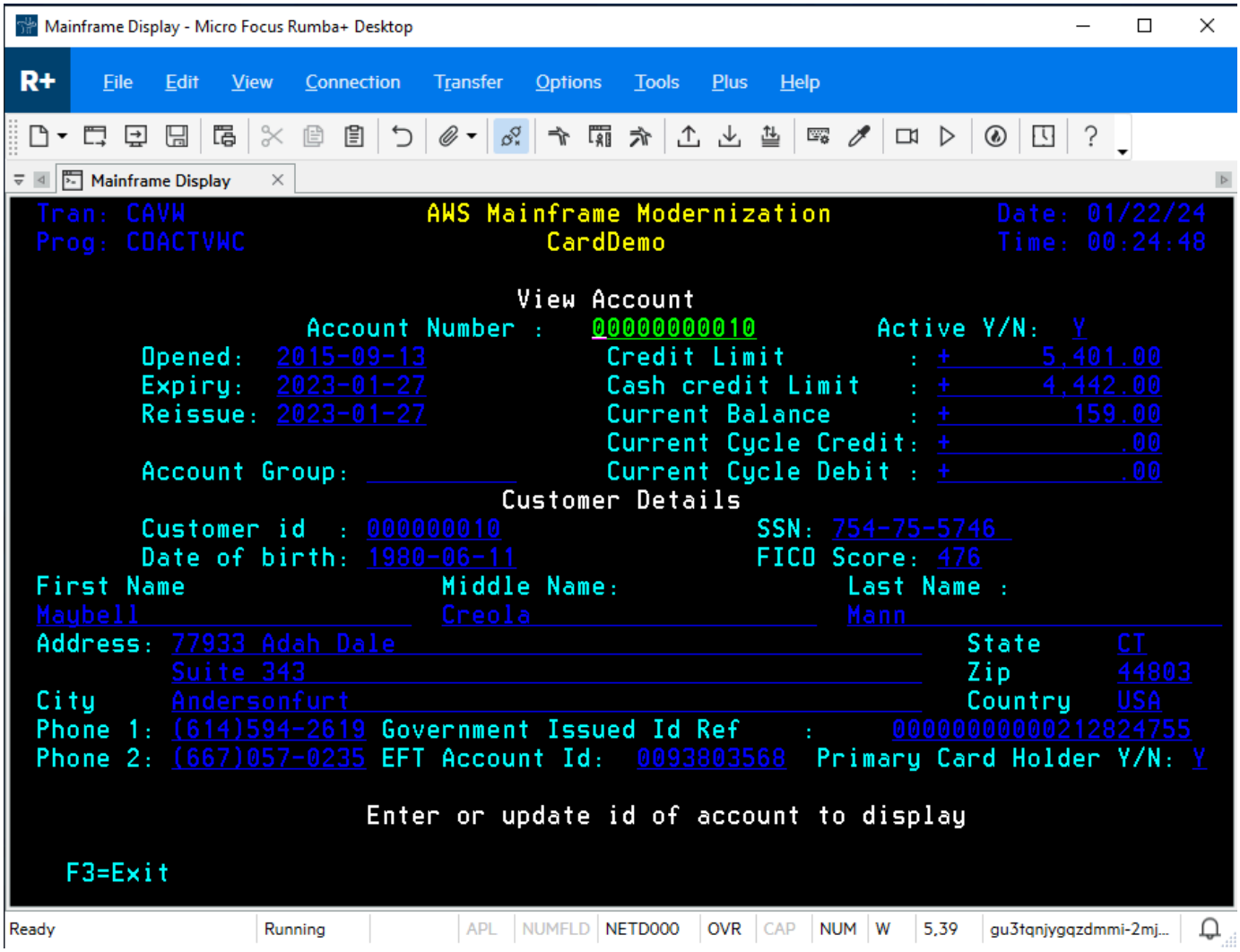


11. Setelah Anda berhasil masuk, Anda dapat menavigasi melalui CardDemo aplikasi.
12. Masuk `01` untuk Tampilan Akun.



13. Masukkan `00000000010` untuk Nomor Akun dan tekan Enter pada keyboard Anda.

> **Note**
>
> Akun valid lainnya adalah `0000000011` dan`00000000020`.



14. Tekan F3 untuk Keluar ke menu, dan F3 untuk keluar dari transaksi.

Pembersihan sumber daya

Jika Anda tidak lagi membutuhkan sumber daya yang Anda buat untuk tutorial ini, hapus untuk menghindari biaya tambahan. Untuk melakukannya, selesaikan langkah-langkah berikut:

- Jika perlu, hentikan aplikasi.
- Hapus aplikasi. Untuk informasi selengkapnya, lihat Hapus aplikasi Modernisasi AWS Mainframe.
- Hapus lingkungan runtime. Untuk informasi selengkapnya, lihat Menghapus lingkungan AWS runtime Modernisasi Mainframe.

- Hapus bucket Amazon S3 yang Anda buat untuk tutorial ini. Untuk informasi selengkapnya, lihat Menghapus bucket di Panduan Pengguna Amazon S3.
- Hapus AWS Secrets Manager rahasia yang Anda buat untuk tutorial ini. Untuk informasi selengkapnya, lihat Menghapus rahasia.
- Hapus tombol KMS yang Anda buat untuk tutorial ini. Untuk informasi selengkapnya, lihat Menghapus kunci AWS KMS.
- Hapus database Amazon RDS yang Anda buat untuk tutorial ini. Untuk informasi selengkapnya, lihat Menghapus instans EC2 dan instans DB di Panduan Pengguna Amazon RDS.
- Jika Anda menambahkan aturan Grup Keamanan untuk port 6000, hapus aturan.

Langkah selanjutnya

Untuk mempelajari cara menyiapkan lingkungan pengembangan untuk aplikasi modern Anda, lihat Tutorial: Mengatur AppStream 2.0 untuk digunakan dengan Micro Focus Enterprise Analyzer dan Micro Focus Enterprise Developer.

Pendekatan modernisasi

Migrasi itu kompleks dan memiliki banyak variabel. AWS Modernisasi Mainframe menawarkan pendekatan evolusioner yang memberikan beberapa kemenangan jangka pendek dengan meningkatkan kelincahan dengan banyak peluang untuk mengoptimalkan dan berinovasi di kemudian hari. Selain itu, Modernisasi AWS Mainframe membantu menyederhanakan perjalanan dan tetap menghormati rincian perusahaan dan bisnis klien Anda. Dua pendekatan utama yang didukung Modernisasi AWS Mainframe adalah refactoring otomatis atau replatforming. Yang harus dipilih tergantung pada situasi klien Anda.

Pemfaktoran ulang otomatis menggunakan alat AWS Blu Age untuk secara otomatis mengonversi kode, data, dan dependensi ke bahasa modern, datastore, dan kerangka kerja, sementara pada saat yang sama menjamin kesetaraan fungsional dengan fungsi bisnis yang sama.

Replatforming menggunakan alat Micro Focus untuk mengubah beban kerja mainframe menjadi layanan gesit. AWS

Anda dapat memikirkan perjalanan modernisasi secara bertahap. Tahap pertama mencakup tiga fase: menilai, memobilisasi, dan bermigrasi dan memodernisasi. Tahap selanjutnya mencakup fase mengoperasikan dan mengoptimalkan, di mana Anda dapat mengidentifikasi lebih banyak peluang untuk inovasi.

Topik

Menilai fase

Pada level tertinggi, fase Penilaian melihat apakah Anda siap untuk bermigrasi. Anda mendefinisikan kasus bisnis, dan kemudian mendidik tim Anda dengan lokakarya dan hari imersi (demo dan laboratorium) yang ditawarkan oleh. AWS Lokakarya dan hari pencelupan membahas berbagai topik. Tugas-tugas ini dilakukan di luar Modernisasi AWS Mainframe.

Memobilisasi fase

Pada fase Mobilisasi, Anda memulai proyek Anda dengan kickoff, dan kemudian menjalankan proses penemuan yang mengekstrak data dari aplikasi mainframe Anda dan memasukkannya ke alat migrasi. Anda mengidentifikasi aplikasi yang ingin Anda migrasi dan memilih beberapa aplikasi untuk pilot. Anda menyempurnakan kasus bisnis Anda, menulis rencana migrasi Anda, dan memutuskan bagaimana Anda ingin menangani keamanan dan kepatuhan, tata kelola akun, dan model operasional Anda. Anda mendirikan pusat cloud keunggulan dengan orang-orang yang tepat dari tim Anda. Anda menjalankan pilot dan mendokumentasikan apa yang Anda pelajari. Anda menyempurnakan rencana migrasi dan kasus bisnis Anda. Banyak dari tugas-tugas ini dilakukan di luar Modernisasi AWS Mainframe.

Migrasi dan modernisasi fase

Fase Migrasi dan Modernisasi berlaku untuk setiap aplikasi dan terdiri dari beberapa tugas, termasuk menugaskan orang, menjalankan penemuan mendalam, mencari tahu arsitektur aplikasi yang tepat, menyiapkan lingkungan runtime aplikasiAWS, mereplatforming atau refactoring kode Anda, mengintegrasikan dengan sistem lain, dan, tentu saja, pengujian. Di akhir fase, Anda menerapkan aplikasi yang direplatformed atau refactored ke produksi dan memotong ke sistem baru di AWS. Sebagian besar atau semua tugas ini dilakukan dalam Modernisasi AWS Mainframe, di AWS layanan lain, atau dalam alat yang menyediakan akses Modernisasi AWS Mainframe.

Jika Anda ingin menggunakan refactoring otomatis, lihat Blu Insights. AWS Blu Insights sekarang tersedia dari AWS Management Console melalui single sign-on. Anda tidak perlu lagi mengelola kredensil AWS Blu Insights yang terpisah. Anda dapat mengakses fitur AWS AWS Blu Age Codebase dan Transformation Center langsung dari file. AWS Management Console

Untuk memigrasikan data dari mainframe keAWS, kami merekomendasikan AWS SCT dan file. AWS Database Migration Service Untuk informasi selengkapnya, lihat Apa itu AWS Schema Conversion Tool? dalam Panduan Pengguna Alat Konversi AWS Skema dan Apa itu AWS Database Migration Service? dalam AWS Database Migration ServiceUser Guide.

Mengoperasikan dan mengoptimalkan fase

Pada fase Operasikan dan Optimalkan, Anda fokus pada pemantauan aplikasi yang digunakan, mengelola sumber daya, dan memastikan bahwa keamanan dan kepatuhan mutakhir. Anda juga menilai peluang untuk mengoptimalkan beban kerja yang dimigrasi.

Konsep

AWS Modernisasi Mainframe menyediakan alat dan sumber daya untuk membantu Anda memigrasi, memodernisasi, dan menjalankan beban kerja mainframe. AWS

Topik

Aplikasi

Beban kerja mainframe yang berjalan di Modernisasi AWS Mainframe. Satu set pekerjaan batch, transaksi interaktif (CICS atau IMS), atau komponen lain terdiri dari aplikasi. Anda mendefinisikan ruang lingkup. Anda harus menentukan dan menentukan komponen atau sumber daya apa pun yang dibutuhkan beban kerja, seperti transaksi CICS atau pekerjaan batch.

Definisi aplikasi

Definisi atau spesifikasi komponen dan sumber daya yang dibutuhkan oleh aplikasi (beban kerja mainframe) yang berjalan di Modernisasi AWS Mainframe. Memisahkan definisi dari aplikasi

itu sendiri penting karena dimungkinkan untuk menggunakan kembali definisi yang sama untuk beberapa tahap (Pra-produksi, Produksi), yang diwakili oleh lingkungan runtime yang berbeda.

Pekerjaan Batch

Program terjadwal yang dikonfigurasi untuk berjalan tanpa memerlukan interaksi pengguna. Dalam Modernisasi AWS Mainframe, Anda harus menyimpan file JCL batch job dan batch job binari dalam bucket Amazon S3, dan menyediakan lokasi keduanya dalam file definisi aplikasi. Saat Anda menjalankan pekerjaan batch, Modernisasi AWS Mainframe melaporkan nilai status berikut:

Mengirimkan

Pekerjaan batch sedang dalam proses diserahkan.

Menunggu

Pekerjaan batch ditunda.

Mengirim

Pekerjaan batch sedang dalam proses dikirim.

Berjalan

Pekerjaan batch saat ini sedang berjalan.

Membatalkan

Pekerjaan batch sedang dalam proses dibatalkan.

Dibatalkan

Pekerjaan batch dibatalkan.

Berhasil

Pekerjaan batch selesai berjalan dengan sukses.

Failed

Pekerjaan batch gagal.

Berhasil Dengan Peringatan

Pekerjaan batch selesai berjalan dengan sukses dengan kesalahan kecil yang dilaporkan. Kode kondisi pekerjaan yang dikembalikan sebagai bagian dari GetBatchJobExecution respons menunjukkan penyebab kesalahan.

Konfigurasi

Karakteristik lingkungan atau aplikasi. Konfigurasi lingkungan terdiri dari jenis mesin, versi mesin, pola ketersediaan, konfigurasi sistem file opsional, dan banyak lagi.

Konfigurasi aplikasi bisa statis atau dinamis. Konfigurasi statis hanya berubah saat Anda memperbarui aplikasi dengan menerapkan versi baru. Konfigurasi dinamis, yang biasanya merupakan aktivitas operasional seperti mengaktifkan atau menonaktifkan penelusuran, berubah segera setelah Anda memperbaruinya.

Kumpulan data

File yang berisi data untuk digunakan oleh aplikasi.

Environment

Kombinasi bernama sumber daya AWS komputasi, mesin runtime, dan detail konfigurasi yang dibuat untuk meng-host satu atau beberapa aplikasi.

Modernisasi mainframe

Proses migrasi aplikasi dari lingkungan mainframe lama ke. AWS

Perjalanan migrasi

end-to-end Proses migrasi dan modernisasi aplikasi lama, biasanya dibuat dari tahap-tahap berikut: Menilai, Memobilisasi, Migrasi dan memodernisasi, dan Mengoperasikan dan mengoptimalkan.

Titik gunung

Direktori dalam sistem file yang menyediakan akses ke file yang disimpan dalam sistem itu.

Refactoring Otomatis

Proses modernisasi artefak aplikasi lama untuk berjalan di lingkungan cloud modern. Ini dapat mencakup kode dan konversi data. Untuk informasi selengkapnya, lihat Modernisasi AWS Mainframe Automated Refactor.

Pembuatan ulang

Proses memindahkan artefak aplikasi dan aplikasi dari satu platform komputasi ke platform komputasi yang berbeda. Untuk informasi lebih lanjut, lihat AWS Mainframe Modernization Replatform

Sumber daya

Komponen fisik atau virtual dalam sistem komputer.

Mesin runtime

Perangkat lunak yang memfasilitasi menjalankan aplikasi.

Aplikasi refactoring secara otomatis dengan Blu Age AWS

Pemfaktoran ulang otomatis dengan AWS Blu Age memberikan end-to-end solusi untuk memigrasi dan memodernisasi aplikasi mainframe Anda. Langkah-langkah dalam proses refactoring adalah sebagai berikut:

- Menganalisis inventaris
- Menganalisis dependensi
- Secara otomatis mengubah kode
- Tangkap dan kelola skenario pengujian

Anda dapat menyelesaikan langkah-langkah sebelumnya di alat Blu Insights, tersedia melalui sistem masuk tunggal dari konsol Modernisasi Mainframe. AWS Untuk informasi lebih lanjut tentang Blu Insights, lihat dokumentasi Blu Insights.

Ketika Anda puas dengan kode sumber yang diubah, saatnya untuk pindah ke AWS, di mana Anda akan menyelesaikan langkah-langkah berikut:

- Bangun dan terapkan aplikasi refactored.
- Terapkan dan pantau aplikasi Anda di Modernisasi AWS Mainframe.

AWS Blu Age Runtime (non-managed) adalah salah satu penawaran layanan Modernisasi AWS Mainframe bersama dengan Blu Age yang dikelola. AWS Dengan AWS Blu Age managed, Anda dapat menerapkan aplikasi modern Anda ke lingkungan AWS-managed yang menyederhanakan pengalaman Anda, sehingga Anda tidak perlu mengelola infrastruktur dasar yang menjalankan aplikasi modern Anda. Sebaliknya, dengan AWS Blu Age Runtime (tidak dikelola) Anda dapat menerapkan aplikasi modern Anda di AWS akun Anda sendiri, sehingga Anda dapat mengelola infrastruktur Anda sendiri. Dengan AWS Blu Age Runtime (non-managed) Anda memiliki fleksibilitas untuk mengoperasikan semua komponen teknis yang diperlukan untuk menjalankan aplikasi modern Anda seperti yang Anda inginkan.

AWS Blu Age Runtime (tidak dikelola) tersedia untuk penerapan di Amazon EC2 dan di Amazon ECS aktif. AWS Fargate

Topik

- AWS Catatan Rilis Blu Age

- AWS Konsep Runtime Blu Age
- AWS Konfigurasi Blu Age Runtime dan file konfigurasi
- AWS Blu Age Runtime API
- AWS Pengaturan Blu Age Runtime (tidak dikelola)
- Ubah kode sumber dengan Blu Age Developer IDE

AWS Catatan Rilis Blu Age

Bagian ini berisi catatan rilis AWS Blu Age Runtime dan Modernization Tools dari versi 3.5.0 dan seterusnya, yang terbaru pertama, diatur berdasarkan nomor versi.

> **Note**
>
> Untuk catatan rilis yang mendahului dokumen ini, hubungi layanan pengiriman AWS Blu Age. Untuk informasi tentang fitur Blu Insights terbaru, lihat rilis Blu Insights.

Topik

- Catatan rilis 3.10.0
- Rilis runtime 3.10.0
- Alat modernisasi rilis 3.10.0
- Catatan rilis 3.9.0
- Rilis runtime 3.9.0
- Alat modernisasi rilis 3.9.0
- Catatan rilis 3.8.0
- Rilis runtime 3.8.0
- Alat modernisasi rilis 3.8.0
- Catatan rilis 3.7.0
- Rilis runtime 3.7.0
- Alat modernisasi rilis 3.7.0
- Catatan rilis 3.6.0
- Rilis runtime 3.6.0

- Alat modernisasi rilis 3.6.0
- Catatan rilis 3.5.0
- Rilis runtime 3.5.0
- Alat modernisasi rilis 3.5.0

Catatan rilis 3.10.0

Rilis AWS Blu Age Runtime dan Modernization Tools ini difokuskan pada peningkatan dan peningkatan dasar inti di seluruh produk yang berusaha meningkatkan kinerja dan ketahanan dalam semua langkah transformasi dan eksekusi. Beberapa fitur utama dan perubahan dalam rilis ini adalah:

- Peningkatan versi dari Java 8 ke Java 17, meningkatkan keamanan dan kinerja, dan memungkinkan pelanggan untuk menyebarkan dan menjalankan aplikasi yang diimplementasikan dalam bahasa yang lebih modern dan menggunakan versi kerangka kerja pihak ketiga terbaru.
- Dukungan tambahan untuk mengelola ruang memori bersama yang besar antara pengguna atau pekerjaan, menyimpan data yang dapat digunakan kembali setelah aplikasi atau instance restart.
- Akses lebih cepat ke kumpulan data besar di Blusam menggunakan mekanisme pagination yang memungkinkan untuk mengambil subset catatan secara bertahap.

Untuk informasi selengkapnya tentang perubahan yang disertakan dalam rilis ini, lihat bagian berikut.

Rilis runtime 3.10.0

Runtime ini didasarkan pada Java17, Spring2.7, dan Angular16.

ZoS

Fitur baru

- Blusam - Menambahkan dukungan untuk kumpulan data besar melalui mekanisme paginasi di mana indeks disimpan dan dimuat menggunakan halaman

Perbaikan

- DataUtilsDitingkatkan.compare untuk menangani konversi prioritas yang lebih rendah dari string ke angka

- Menambahkan dukungan untuk memeriksa bahwa no dibuat dengan nilai ByteRange yang tidak tepat melalui properti YMLDataSimplifier. byteRangeBoundsPeriksa
- Enhanced removeSosi () untuk mendukung inisialisasi dengan karakter kosong GraphicAlphanumericType
- Menambahkan ketahanan untuk operasi pekerjaan dan pembacaan status GDG yang aman
- Blusam - Menambahkan dukungan untuk membersihkan Ehcache dari kumpulan data Blusam melalui metode baru bernama .removeCache () CoreBluesamManager
- Blusam - Peningkatan perilaku hapus/ganti nama untuk kumpulan data Blusam biasa
- Redis - Dukungan yang ditingkatkan untuk membuka kunci set data dan membersihkan kunci rekaman
- JICS - Memperbaiki pesan kesalahan untuk permintaan yang gagal
- JCL - Menambahkan dukungan untuk rangkaian variabel ControlM berdasarkan karakter titik
- JCL - Menambahkan dukungan untuk Write ADVANCING (ADV) untuk file GDG
- JCL - Dukungan yang ditingkatkan untuk nomor generasi saat ini setelah menghapus semua file GDG
- JCL - Dukungan yang ditingkatkan untuk pembacaan RDW/RecordSize dari katalog pada pembuatan dataset
- JCL - Menambahkan dukungan untuk memperbarui objek sumber daya (dari AbstractSequentialFile) saat membuka file dengan ukuran catatan keluaran data
- JCL - Peningkatan kinerja IDCAMS
- JCL - Peningkatan dukungan untuk PERNYATAAN CETAK dengan menambahkan “CHAR” sebagai alias “KARAKTER”
- SORT - Dukungan yang ditingkatkan untuk operasi penyalinan dari kumpulan data panjang tetap Blusam ke kumpulan data dengan panjang variabel
- SORT - Tata bahasa pengurutan yang disempurnakan untuk menangani beberapa pernyataan tertentu

AS400

Fitur baru

- Menambahkan dukungan untuk Ruang Pengguna dan API terkait
- Ditambahkan dukungan untuk parameter TOMSGQ SNDPGMMSG dan mengimplementasikan antrian pesan

- CL - Menambahkan dukungan untuk parameter FILE dan SPLFNAME untuk perintah OVRPRTF
- CL - Menambahkan dukungan untuk menangani perpustakaan untuk tabel partisi yang sesuai dengan perintah CPYF
- CL - Menambahkan dukungan untuk menangani perintah CHGCURLIB dan mempertimbangkan perpustakaan saat ini saat membangun kueri
- CL - Menambahkan dukungan untuk menangani perintah cl sebagai bagian dari panggilan stacktrace

Perbaikan

- Ditingkatkan MessageHandlingBuilder untuk penanganan entri jejak tumpukan panggilan yang lebih baik
- Peningkatan eksekusi paralel dari fitur ContextPreConstruct
- Atribut tampilan yang ditingkatkan saat catatan dibuat oleh SFLINZ
- Peningkatan SAVOBJ untuk memungkinkan penanganan beberapa file output
- Peningkatan penanganan program groovy dengan menambahkannya programCallStack ketika mereka dipanggil dari program Java
- Peningkatan deteksi posisi atas modal bantuan
- Peningkatan fungsionalitas TopGMQ saat parameter TomSGQ disediakan untuk SNDPGMMSG
- Peningkatan pengambilan pesan yang telah ditentukan dan fungsionalitas pemuat pesan
- Peningkatan penanganan CPYTOIMPF karakter pembatas dalam konten
- Kunci rilis yang ditingkatkan pada catatan BACA

Kemampuan transversal

Fitur baru

- Ditambahkan terjemahan untuk pesan sistem di Front-End
- Menambahkan metode baru ExecutionContext untuk mengembalikan tumpukan panggilan program
- Tetapkan pemisah garis (untuk penyederhanaan data) terlepas dari lingkungan sebenarnya
- Ditambahkan kemungkinan untuk mengkonfigurasi jalur JSON model SQL

Perbaikan

- Memperbaiki metode perbandingan DataUtils. compareAlphInt() saat bantalan terlibat
- Pembuatan bendera untuk mengizinkan perilaku khusus pada pengecualian dalam kueri kursor
- Peningkatan konversi db LOWVALUES grafis

Pihak ketiga

- Tingkatkan untuk mengurangi CVE-2024-21634, CVE-2023-34055, CVE-2023-34462, IN1-JAVA-ORGSPRINGFRAMEWORKSECURITY-5905484, CVE-2023-46120, CVE-2023-6481, CVE-2023-6378, CVE-2023-5072)

Alat modernisasi rilis 3.10.0

ZoS

Perbaikan

- COBOL - Menambahkan dukungan untuk fungsi ABS
- JCL - Cakupan variabel yang ditingkatkan: dilampirkan ke STEP alih-alih JOB
- Injeksi parameter kursor yang ditingkatkan untuk nilai rendah/tinggi
- Penguraian CSD yang ditingkatkan, terutama untuk TRANSAKSI jarak jauh

AS400

Perbaikan

- Cek kosong yang dihapus untuk Indikator Tingkat Kontrol
- Ditambahkan dukungan untuk nama eksternal untuk kata kunci IMPORT/EKSPOR
- Menambahkan dukungan untuk %LEN pada bidang
- CL - Menambahkan dukungan untuk operator baru untuk bahasa CLLE
- CL - Menambahkan dukungan untuk IF bersarang
- COBOL - Peningkatan penanganan perintah START saat digunakan dengan beberapa tombol
- DSPF - Peningkatan penanganan posisi kursor dengan nomor catatan
- DSPF - Meningkatkan format untuk bidang numerik, numerik saja yang ditandatangani, dan bidang dengan skala besar

- DSPF - Meningkatkan penentuan judul untuk Screen General Help
- DSPF - Peningkatan dukungan spesifikasi Input/Output
- DSPF - Peningkatan penanganan pemisah pengelompokan selama validasi bidang numerik
- Peningkatan output pemetaan/catatan DDS
- Kemampuan kata kunci REFFLT file printer yang ditingkatkan untuk menyelesaikan bidang yang direferensikan
- RPG - Dukungan yang ditingkatkan untuk pernyataan “SEMUA gratis”
- RPG - Peningkatan parsing kondisi dan menambahkan dukungan untuk menangani CABXX tanpa TAG hasil
- RPG - Peningkatan spesifikasi input penanganan bidang numerik
- RPG - Peningkatan penanganan panggilan prosedur dalam kondisi IF/ELSEIF/WHEN
- RPG - Peningkatan penanganan perintah READ saat dipanggil pada file dspf
- RPG - Meningkatkan dukungan untuk file yang mengacu pada DDS yang tidak ada
- Meningkatkan penanganan REFFLD ketika melewati nama format rekaman fisik
- Ditambahkan dukungan untuk menggunakan 'return' sebagai nama kolom db

Kemampuan transversal

Fitur baru

- Oracle - Memungkinkan untuk mendefinisikan pengguna daripada SYS untuk menyimpan fungsi bawaan

Perbaikan

- Versi Java yang ditingkatkan dari v8 ke v17
- Peningkatan kondisi SQL dengan nama kolom Cluster
- Ditambahkan dukungan untuk ORDER BY klausa dari tampilan

Catatan rilis 3.9.0

Rilis AWS Blu Age Runtime dan Modernization Tools ini difokuskan pada beberapa peningkatan transversal di seluruh produk yang berusaha meningkatkan kinerja dalam arsitektur ketersediaan

tinggi, bersama dengan kemampuan baru untuk meningkatkan eksekusi pekerjaan ke tingkat berikutnya. Beberapa fitur utama dan perubahan dalam rilis ini adalah:

- Versi upgrade dari Angular 13 ke Angular 16, meningkatkan keamanan dan memberikan akses ke fitur baru yang meningkatkan kinerja dalam aplikasi online pelanggan.
- Tambahkan dukungan fitur lintas pekerjaan di AS400, dengan cahaya utama bahwa pekerjaan dapat mengirim pesan pertanyaan secara serempak di antara mereka, memungkinkan pemisahan dalam pekerjaan modern.
- Peningkatan kinerja pada penggunaan Redis, termasuk optimasi kumpulan koneksi, keamanan tinggi pada koneksi, dan mekanisme penguncian kumpulan data yang ditingkatkan.

Untuk informasi selengkapnya tentang perubahan yang disertakan dalam rilis ini, lihat bagian berikut.

Rilis runtime 3.9.0

ZoS

Fitur baru

- Urutkan program: Input VSAM yang diperbarui dengan panjang tetap
- JHDB DB: Menambahkan batas waktu yang dapat dikonfigurasi

Perbaikan

- Dukungan yang ditingkatkan untuk pemisah baris untuk streaming jika digunakan dalam rangkaian file
- Dukungan yang ditingkatkan untuk membuka file sekuensial gabungan. Inisialisasi DataSetIndex setelah membuka file
- Dukungan yang ditingkatkan untuk pemisah desimal virtual ketika a NumericEditedType dipengaruhi ke nilai numerik
- Dukungan yang ditingkatkan untuk NumericEditedType nilai negatif
- IDCAMS: Kartu SYSIN sekarang dibaca menggunakan properti “pengkodean” yang ditentukan dalam.yl application-utility-pgm
- IDCAMS: Tata bahasa yang diperbarui untuk mendukung argumen FILE (..) dalam pernyataan DEFINE CLUSTER

- INFUTILB: Menambahkan dukungan untuk argumen DFSIGDCB untuk mengganti parameter DCB dari DD SYSREC
- INFUTIL: Peningkatan dukungan untuk parameter “DFSIGDCB YES”
- Peningkatan SPLICE untuk menangani file input besar
- DFSORT: Peningkatan penanganan bidang komentar
- DFSORT: Menambahkan dukungan untuk format numerik formulir gratis (ditandatangani /tidak ditandatangani) (SFF/UFF)
- SORT: Menambahkan dukungan parsing untuk pernyataan OPTION PRINT dan OPTION ROUTE
- SORT/ICEMAN: Menambahkan dukungan untuk operasi divisi tertutup (lapangan dengan operator DIV)
- Dukungan yang disempurnakan untuk CICS READ menggunakan tombol generik
- StringUtilsFungsi.chargraphic tetap untuk menghapus SOSI dari tipe grafis
- Tingkatkan kinerja pada DataUtils. isDoubleBytePengkodean
- JCL: Dukungan yang ditingkatkan untuk mode disposisi KEEP untuk kumpulan data sementara. Sistem mengubah disposisi menjadi PASS
- JCL: Menangani parameter DCB secara dinamis
- JCL: Output SUM FIELDS yang ditingkatkan untuk nilai yang salah
- JCL: CommondDutils: :getContent sekarang mencari recordSize di katalog
- JCL: Baca atribut RDW/RecordSize dari katalog pada pembuatan dataset
- JCL: Menambahkan dukungan untuk DCB=.MYDD untuk menyalin parameter DCB dari DD ke yang lain dalam langkah pekerjaan yang sama
- JCL: Peningkatan sistem pewarisan ukuran catatan
- JCL: Menambahkan kunci set data eksklusif (Redis)
- Redis: Menambahkan dukungan SSL untuk mode mandiri
- Redis: Menambahkan jumlah kunci Redis yang disinkronkan dengan kunci
- Redis: Parameter Pool yang didukung untuk kunci Redis
- Redis: Penyegaran metadata yang dioptimalkan dengan Redis
- Redis: Peningkatan dukungan cluster redis
- Peningkatan pada kunci terbuka dengan mode IO
- Set data yang ditingkatkan mengunci kinerja dan membersihkan kunci yang tidak digunakan

- Jalur yang disempurnakan dari kumpulan data selama membatalkan pendaftaran file
- Peningkatan pembatalan cache jendela pra-pengambilan
- Menambahkan dukungan untuk penggunaan penyedia sumber data utilitas aman utas
- Pemeriksaan nullity DatasetState yang disempurnakan
- Dukungan yang ditingkatkan untuk tidak membuka kembali set data yang sudah dibuka
- Menambahkan ketahanan untuk operasi akhir pekerjaan
- Dukungan yang ditingkatkan untuk urutan indeks untuk kunci yang memungkinkan duplikat
- Dukungan yang ditingkatkan untuk urutan serialisasi daftar lewati
- Menambahkan dukungan untuk fitur dump debug untuk membantu mendiagnosis masalah urutan indeks
- Dukungan yang ditingkatkan untuk penyegaran metadata
- Dukungan yang ditingkatkan untuk pembacaan massal Blusam

AS400

Fitur baru

- Membuat registri konteks aplikasi
- Support untuk kata kunci DSPF CLRL (NO) Support pemantauan kunci catatan
- Support untuk keyed DataQueue
- Support untuk pesan INQUIRY untuk pekerjaan batch
- Menambahkan dukungan untuk file Printer yang dijelaskan Program untuk AS400 COBOL
- Menangani perintah cl RMVJOBSCDE
- Perbaikan untuk RUNSQL/DLYJOB
- CHKOBJ: Meningkatkan kode kesalahan lama untuk parameter LIB
- SNDPGMMSG: Mendukung parameter string
- RTVDTAARA: Peningkatan substring di LDA
- DSPFD: Param FILE didukung ditambahkan untuk nama file tertentu
- RUNQRY: Dukungan untuk file sql di QRY PARAM
- CRTDUPOB: Support untuk menyalin data antar area data
- SBMJOB: Mengonversi instruksi untuk digunakan JobQueueManager

- OPNQRYF: Menambahkan dukungan untuk perpustakaan Qtemp
- CRTDUPOBJ: Peningkatan logika untuk menyalin konten partisi
- CRTDUPOBJ: Menambahkan dukungan untuk Qtemp untuk tampilan
- RTVSYSVAL: Support untuk nilai SYSVAL, QDATFMT dalam perintah CL
- CHKOBJ: Menambahkan dukungan untuk OUTQ
- RTVJOBA: Mendukung SWS param
- SNDPGMMSG dan RCVMSG: Parameter tambahan didukung MSGF, MSGFLIB, MSGDTA, MSGTYPE, KEYVAR, MSGKEY, MSGID

Perbaikan

- Peningkatan kartu I/O WORKSTATION mendukung
- Peningkatan penanganan pesan set yang melapisi pesan sebelumnya
- Mendukung informasi pesan tambahan pada array-messageline
- Peningkatan akses pembungkus array mandiri di dalam EVAL, SortA, figuratives
- Tingkatkan pembersihan DAO saat aplikasi online berakhir
- Menambahkan dukungan untuk format tanggal tambahan dan meningkatkan penanganan input string
- Peningkatan penanganan CVTDAT SYSVAL dengan menambahkan nilai sistem kelas pembantu Decode dan membangun parameter dari perintah CL SbmJob
- Paket yang dihapus com.netfective.bluage.gapwalk.rt.blu4iv dari pemindaian komponen gapwalk-cl-command
- Meningkatkan dukungan pesan yang telah ditentukan untuk API antrian pesan
- Meningkatkan dukungan retrieveSubfileRecord untuk catatan yang ditulis dalam program lain
- Meningkatkan dukungan pesan langsung untuk API antrian pesan
- Peningkatan penanganan area data lokal saat mengirimkan pekerjaan
- Mulai JobQueues secara otomatis saat server dimulai
- Menggunakan konfigurasi ApplicationContext untuk memecahkan kode parameter untuk SBMJOB
- Peningkatan pesan kesalahan yang disediakan sistem
- Memungkinkan RTVMSG untuk mencari file.properties di sub-direktori bersarang
- Menangani reset entitas yang terikat pada pointer buruk/tidak valid

- Ditingkatkan MessageHandlingBuilder untuk menampilkan MSgID dan MsgFile nama sebagai string untuk RCVMSG
- Metode withMsgFile Nama yang disempurnakan dari API antrian pesan
- Mekanisme kunci area data yang ditingkatkan
- RTVMBRD: Support untuk huruf kecil dan besar untuk parameter FILE
- CRTDUPOBJ: Peningkatan penanganan tampilan
- CPYTOSTMF: Peningkatan penanganan koneksi
- CPYF: Peningkatan dalam menangani nama direktori saat menyalin dari file datar
- RCVF: Menangani parameter DEV/RCDFMT dengan benar dan transformasi RCDFMT untuk groovy dan java
- RCVF: Menangani panggilan berikutnya dan menghindari mengatur ulang kursor
- CPYF: Menambahkan dukungan untuk menulis dari file datar
- CRTDUPOBJ: Menambahkan penanganan obj baru dengan perpustakaan Qtemp
- CHGDTAARA: Peningkatan panjang maksimum area data dari 256 menjadi 2000
- SAVOBJ: Pastikan catatan yang disimpan dalam urutan penyisipan
- RTVDTAARA: Nilai diambil (tidak akan dipangkas)
- CHKOBJ: Mengembalikan pesan monitor yang benar ketika anggota tidak ada
- RTVDTAARA: Menambahkan dukungan substring LDA
- RTVDTAARA: Mengembalikan spasi putih hingga panjang variabel yang ditentukan dalam parameter RTNVAR
- RTVDTAARA: Mendukung parameter integer untuk awal dan panjang dan mendukung format transformasi terbaru
- CHGDTAARA: Menambahkan dukungan untuk parameter yang mencakup batas bawah dan atas
- CHKOBJ: Menangani nilai VIEW untuk tipe objek parameter
- CHKOBJ: hasil disetel ke true terlepas dari anggota jika tampilan ada

Kemampuan transversal

Fitur baru

- Menangani pembuatan laporan ke file.txt
- Menambahkan properti sumber data CurrentSchema XA ke manajer rahasia

- Tambahkan database.cursor.raise.already.opened.error properti yl untuk mengaktifkan kerangka kerja untuk meningkatkan kesalahan SQLCODE 502 saat kursor yang sudah dibuka terbuka

Perbaikan

- Menambahkan gapwalk pom ke AWS Blu Age pada kemasan Amazon EC2
- Menggunakan paradigma handler sinyal baru secara default
- Tambahkan dukungan untuk kunci ketika disposisi adalah MOD atau LAMA
- Ditambahkan cache untuk menyimpan pola waktu tanggal database
- Peningkatan fungsi pemeriksaan PackedType
- Meningkatkan DataUtils fungsi.setTo untuk Rekaman dengan VariableSizeArray
- Menangani opsi MQ SYNCPOINT sehubungan dengan unit run
- Kerangka kerja yang diaktifkan untuk mengatur SQLCODE pada transaksi rollback
- Menambahkan nama kelas driver otomatis sesuai dengan rahasia kunci mesin
- Batas waktu Program/Transaksi
- Kembalikan posisi kursor setelah Rollback saat mengakses kursor

Pihak ketiga

- Tingkatkan SnakeYAML, Redisson, dan Amazon SDK, YamlBeans hapus (kurangi CVE-2022-25857, CVE-2023-24621, CVE-2023-42809, CVE-2023-44487)

Alat modernisasi rilis 3.9.0

ZoS

Perbaikan

- Dukungan yang ditingkatkan untuk XML-TEXT sebagai sumber untuk target tipe String
- Peningkatan alur kerja STM ke UMP untuk mendukung pola pembagian X/ (Y/Z)
- JHDB DB: Menerima panggilan ROLLBACK sebelum pembaruan basis data apa pun
- JHDB DB: Menerima ROLLBACK bahkan jika transaksi dihentikan (NOP)
- JCL: Peningkatan fungsi validasi langkah

- SORT: Menangani fungsi SUM dengan nilai negatif desimal zona
- COBOL: Menambahkan dukungan untuk lolos kutipan tunggal/ganda dalam literal string

AS400

Perbaikan

- Peningkatan fungsi bawaan %editc penanganan kode edit X dengan menambahkan angka nol di depan
- Peningkatan penanganan input hanya bidang nilai awal
- Menambahkan tombol tindakan untuk membantu dialog
- Catatan footer tabel dinamis muncul di bagian bawah
- Menangani perintah START tanpa FASE KUNCI untuk file yang menentukan RECORD-KEY yang sebenarnya
- Menambahkan nilai default untuk float dan NumberUtils: :pow type
- Ditambahkan dukungan mendefinisikan variabel menggunakan LIKE (IN)
- Diperbarui UNTUK penanganan loop untuk mendukung menghilangkan elemen opsional
- Penguraian RPG yang diperbarui untuk mengaitkan catatan dengan nama array CTDATA
- Peningkatan penanganan indikator untuk pernyataan CABxx
- Mendukung parameter opsional pada kata kunci COMMIT
- Peningkatan dukungan Format Kata Kunci di LF
- Kode operasi LOOKUP terkelola dengan indikator tinggi dan sama (atau rendah dan sama)
- Nama kunci PF yang ditangani dinyatakan dalam tanda kutip ganda
- Meningkatkan penanganan EDTCDE X untuk tidak menekan angka nol terkemuka
- Peningkatan dukungan untuk MSGCON dalam file printer tidak menghasilkan label yang tidak disebutkan namanya
- KONTEN bidang dibagikan oleh beberapa struktur data
- Parameter ERRSFL yang ditangani dalam kombinasi dengan SFLMSG/SFLMSGID
- Peningkatan kode utama sebelum cakupan deklarasi proc rpg gratis penuh
- Menambahkan spesifikasi kontrol terkondisi parsing
- Peningkatan dukungan untuk setErrSfl () metode di dataholdermapper
- Resolusi tipe yang ditingkatkan untuk variabel yang dibuat secara internal

- Peningkatan dukungan untuk opcode Z-ADD
- Meningkatkan penanganan bidang konstan dengan nilai DFT
- Meningkatkan dukungan bidang integer di dalam status program ds
- Penugasan indikator yang ditangani di parameter ENTRY
- Peningkatan filter kata kunci disebarkan melalui kata kunci ref/reffield
- Struktur DataArea data tanpa nama yang didukung
- Peningkatan penanganan tipe data pointer
- Elemen array yang ditangani digunakan untuk mendefinisikan variabel dengan akses array dukungan kata kunci LIKE di bidang output
- Peningkatan dukungan untuk numerik yang ditandatangani, hanya menampilkan digit
- Hubungan logis yang didukung pada kartu O
- Kasus uji untuk %CHAR pada alfanumerik
- Kata kunci spesifikasi kontrol yang didukung utama
- EDTCDE dengan dua parameter dalam file printer
- Penguraian FullFree RPG yang ditingkatkan
- Meningkatkan tabel dinamis untuk memastikan footer diposisikan dengan benar
- Menambahkan dukungan untuk menginisialisasi tipe numerik dengan SEMUA konstanta figuratif
- Peningkatan penanganan beberapa file logis RPG yang mereferensikan file fisik yang sama
- Tingkatkan deteksi bidang yang dimodifikasi di layar modern
- Sinkronisasi modal dengan bidang dinamis
- Meningkatkan penanganan output hanya bidang numerik yang ditandatangani
- Meningkatkan dukungan kartu I/O WORKSTATION

Kemampuan transversal

Fitur baru

- Alat Migrator Data: Menambahkan properti ebcdicFilesWith VarcharIn VB untuk memungkinkan memperhitungkan panjang 2-byte VARCHAR saat membaca byte
- Menerapkan API umum untuk mencatat kesalahan
- Implementasi BluAgeErrorDictionaryUtils dan penggunaan API umum untuk mencatat kesalahan dan/atau info di Cobol2Model, CycleBuilder RPG, Definitions2Model dan FieldsProcessor

- Tata bahasa SQL yang ditingkatkan untuk mendukung definisi klausa isolasi yang berbeda

Perbaikan

- Versi Angular yang ditingkatkan ke v16
- Angular: Versi ajv yang ditingkatkan dari 6 menjadi 8,9

Pihak ketiga

- Upgrade Groovy ke versi 2.4.15

Catatan rilis 3.8.0

Rilis AWS Blu Age Runtime dan Modernization Tools ini difokuskan pada beberapa peningkatan transversal di seluruh produk untuk meningkatkan kualitas dan keamanannya, bersama dengan peningkatan kinerja untuk caching dan penyatuan dukungan perintah dalam satu distribusi. Beberapa fitur utama dan perubahan dalam rilis ini adalah:

- Versi upgrade dari Spring 2.5 ke Spring 2.7, meningkatkan dukungan pemeliharaan, kinerja, dan keamanan platform.
- Penyatuan lebih dari 82 perintah CL mendukung sebagai bagian dari over-the-counter distribusi untuk memfasilitasi penggunaan dan penyebaran aplikasi modern yang sebelumnya menggunakan skrip CL.
- API baru tersedia untuk beroperasi dan berinteraksi lebih baik dengan kumpulan data BluSam, seperti impor terintegrasi ke layanan terkelola dan kemampuan untuk mencantumkan informasi metadata kumpulan data.
- Peningkatan kinerja dan perluasan penggunaan Redis, termasuk ketersediaan dalam mode cluster, pengambilan data ketersediaan tinggi, standarisasi penggunaan rahasia.

Untuk informasi selengkapnya tentang perubahan yang disertakan dalam rilis ini, lihat bagian berikut.

Rilis runtime 3.8.0

ZoS

Fitur baru

- Menangani definisi kunci sebagai string untuk DynamicFileBuilder
- DFSORT: Menambahkan dukungan untuk multi-item di OUTFIL TRAILER1 + inisialisasi tata bahasa DFSORT
- Alat CommondDutils: menangani ukuran rekaman dalam data dalam aliran
- File yang diindeks: menangani opsi GENKEY

Perbaikan

- Layanan pemuatan BluSam eksternal dalam toples terpisah
- Ditambahkan dukungan untuk mengatur lokasi untuk menyimpan file sementara
- Peningkatan mekanisme cache bersama untuk kasus multi-node
- Penggunaan cache bersama: IDCAMS memverifikasi optimasi
- Meningkatkan injeksi ROWID untuk pilihan tertanam
- JCL: Setiap prosedur pekerjaan in-stream sekarang dibuat dalam file Groovy yang berbeda
- Pastikan card-demo-v 2 cakupan pada kartu IDCAMS JCL
- BluSam: Hindari duplikat WarmUp saat menggunakan beberapa instance
- Mengurangi jejak memori pada hidrasi cache
- Dukungan konfigurasi kolam Jedis
- Menambahkan pemisah baris untuk streaming jika digunakan dalam rangkaian file
- Support untuk kartu EBCDIC+komentar blok (/*.../) di utilitas IDCAMS
- Kueri dukungan basis data: dukungan untuk string byte ganda dalam konversi level49 menuju SQL
- Tata bahasa DFSORT: mengimplementasikan 17 pernyataan kontrol+integrasi 2 di antaranya (OMIT/INCLUDE)
- Tingkatkan kolom GRAFIS ambil INFUTILB
- Support untuk membaca file dengan tabel Ukuran variabel
- Support for ZonedType with nibble signed di mana bit pertama byte terakhir adalah 'E'
- DFSORT/ICETOOL menambahkan dukungan untuk argumen NOMATCH =(..) jika record tidak cocok dengan salah satu konstanta CHANGE find
- Kompatibilitas Redis Cluster
- Menangani Status Job (Gagal) berdasarkan kode keluar yang asyik
- Peningkatan dukungan CICS SYNCPOINT ROLLBACK.
- Jendela pra-ambil untuk mengoptimalkan penggunaan cache Redis

- JCL/GROOVY: Mewarisi properti isRDW dari kumpulan data langkah sebelumnya saat DISP = (, PASS)
- Menangani salinan sebagian data dengan array ukuran variabel

AS400

Fitur baru

- Support untuk kartu I/O untuk file tampilan
- Support untuk informasi pesan tambahan untuk kata kunci DSPF ERRMSGID dan CHKMSGID
- Support untuk beberapa pesan kesalahan di layar frontend
- Menambahkan atau meningkatkan dukungan 82 perintah CL dalam gapwalk-cl-command aplikasi

Perbaikan

- Peningkatan dukungan untuk DELETE dan READ di bawah kontrol komitmen
- ConvertDate di dalam %dec bawaan
- Header keamanan XSS yang diberlakukan
- Peningkatan ketahanan dan konsistensi generasi STM (penanganan yang lebih baik dari: garis kelanjutan dalam bentuk bebas rpg, koma untuk bagian desimal, blok bentuk bebas dalam definisi/deklarasi)
- DataHolderMapper Generasi yang ditingkatkan
- Menambahkan kekokohan dan mengubah ruang lingkup DataAreaFactory
- Meningkatkan pergeseran fokus pada tombol tab
- Peningkatan kinerja pada pembuatan laporan Jasper
- Tampilan desimal yang ditingkatkan dengan padding 0s
- Peningkatan dukungan untuk bidang ROW/COL di INFDS
- Meningkatkan dukungan untuk bidang yang dimodifikasi dari layar
- Menambahkan getter untuk nama dan jalur laporan yang dihasilkan
- Peningkatan pada panjang Dataqueue
- Peningkatan konfigurasi otomatis dari Job Queues agar sesuai dengan standar baru di Spring Boot 2.7
- Pembaruan workstation yang ditingkatkan untuk beberapa sesi bersamaan

Kemampuan transversal

Fitur baru

- Support untuk Toleransi Data Tidak Valid untuk Dikemas
- Ditambahkan paginasi/penyaringan ke daftar titik akhir dataset

Perbaikan

- Strategi transformasi kueri ORACLE yang disempurnakan dalam perbandingan kolom terhadap string kosong
- Menangani BLOB DB2 dengan program utilitas DSNTEP dan INFUTILB. BLOB DB2 sekarang dimodernisasi menjadi postgres tipe BYTEA.
- Peningkatan penghapusan item terakhir kursor
- Dukungan yang ditingkatkan untuk menghapus file RRDS
- Peningkatan kinerja AWS rahasia Blusam
- Peningkatan penanganan koneksi database dalam kerangka SQL
- Kunci manajer rahasia AWS multi-datasource standar
- Perbaikan regresi kinerja
- Peningkatan fungsi pemeriksaan untuk PackedType
- Peningkatan penanganan LOW-VALUE untuk PackedType
- Kemasan keamanan pegas yang ditingkatkan untuk koneksi cognito
- Tidak menerapkan pengkodean dan decoding codeshiftpoint pada database yang ditargetkan DB2

Pihak ketiga

- Upgrade Spring Boot dari 2,5 menjadi 2,7

Alat modernisasi rilis 3.8.0

ZoS

Fitur baru

- JCL: Menangani aliran dengan carriage return “\ r”

Perbaikan

- Peningkatan logging untuk mencegah pembagian dengan nol saat memodernisasi klausa DIVIDE dengan ON SIZE ERROR
- JCL: Dukungan yang ditingkatkan untuk memanggil prosedur dalam suatu prosedur
- Support untuk kata kunci OF dalam perintah FORMATTIME CICS ketika ada bidang yang ambigu
- JCL: dukungan untuk karakter â¥ dalam variabel
- JCL: komputasi RC berdasarkan langkah sebelumnya
- Membandingkan byte alih-alih string saat PL1 SUBSTR digunakan
- Peningkatan inisialisasi array multidimensi dari sumber tunggal
- Peningkatan parsing COBOL ketika melibatkan query SQL tunggal di blok IF

AS400

Fitur baru

- Support untuk pernyataan IF bersarang di CL
- Peningkatan dukungan untuk pernyataan ENDDO dalam bentuk bebas RPG

Perbaikan

- Peningkatan dukungan untuk pengkondisian Tingkat Kontrol
- Pengembalian prototipe yang ditingkatkan dengan LIKE
- Peningkatan dukungan untuk menangani fungsi %months, %year, %days
- Dukungan untuk fitur bantuan untuk seluruh layar
- Penanganan BLANKS figuratif diteruskan sebagai parameter
- Peningkatan ekspresi EVAL dengan operator “”
- Menangani perintah START tanpa FASE KUNCI
- Peningkatan penanganan Keyword LIKEREC
- Peningkatan pada subbidang yang tidak disebutkan namanya
- Perbaikan prosedur mengembalikan tipe yang tidak ditandatangani
- Peningkatan dukungan untuk operasi RESET (RPG Gratis), %CHAR dan% DEC bawaan

- Peningkatan fungsi bawaan %LOOKUPXX
- Peningkatan dukungan untuk kata kunci LIKEDS pada prosedur tanpa prototipe
- Menangani tipe array kata kunci Dim (VAR, AUTO)
- Peningkatan dukungan untuk XFOOT
- COBOL: peningkatan dukungan untuk bidang RENAMES
- CL: mendukung kondisi sementara (benar)
- Meningkatkan penanganan array mandiri dengan kata kunci LIKE
- Peningkatan fungsi bawaan %INT
- Peningkatan RPG Full Free parsing
- Peningkatan dukungan untuk array di linkage
- CL2GROOVY: Support Select Statement
- Peningkatan kata kunci DSPF “ERRMSGID”
- Meningkatkan penanganan inisialisasi byte dengan angka nol terkemuka
- Peningkatan AuthorizedValues untuk bidang numerik
- Menangani extender H untuk pernyataan EVAL formulir Gratis
- CL ke Groovy: Support substring dari LDA
- Peningkatan dukungan untuk RESET pada catatan
- Meningkatkan penanganan EDTCDE dan EDTWRD dengan referensi
- Peningkatan pemetaan input-field dengan bidang DDS
- Peningkatan dukungan untuk karakter MOVEA ke dalam array IN
- Peningkatan prototipe dengan kata kunci LIKEDS
- Peningkatan dukungan untuk kata kunci DSPF DSPATR
- Peningkatan parsing D-card dengan +/-
- Ditambahkan kekokohan dalam panggilan program
- Menambahkan kekokohan dalam proses penyelesaian lapangan

Kemampuan transversal

Perbaikan

- FrontEnd: Simulasikan acara tempel untuk input IME

Pihak ketiga

- Upgrade Spring Boot dari 2,5 menjadi 2,7

Catatan rilis 3.7.0

Rilis AWS Blu Age Runtime dan Modernization Tools ini terutama mencakup penyempurnaan untuk mendukung perintah dan utilitas yang lebih baik, kemampuan untuk berintegrasi dengan AWS Secrets Manager dan fitur pemantauan baru. Beberapa perubahan utama dalam rilis ini adalah:

- Beberapa komponen runtime sekarang dapat menggunakan AWS Secrets Manager untuk meningkatkan pengaturan keamanan aplikasi modern, sebagian besar terkait dengan sumber data utilitas, Redis untuk TS Queues, cache dan kunci. BluSam
- Memantau titik akhir yang memungkinkan untuk mengambil metrik transaksi, batch, dan JVM untuk optimalisasi penggunaan sumber daya dan manajemen operasional, seperti status, durasi, volume, dan lainnya.
- Fitur baru untuk mendukung panggilan IBM MQ dalam RPG, dan peningkatan cakupan transformasi JCL SORT dan IDCAMS.

Untuk informasi selengkapnya tentang perubahan yang disertakan dalam rilis ini, lihat bagian berikut.

Rilis runtime 3.7.0

Topik

- ZoS
- AS400
- Kemampuan transversal

ZoS

Fitur baru

- Tingkatkan kueri parsing yang terlibat dalam aplikasi utilitas program dengan menggunakan SQL seperti tata bahasa. (V7-9401)
- Menangani Array Ukuran Variabel yang diindeks saat offset (V7-9904)
- Support INSERT SQL TIME kolom ke DB2 dengan format 24:00:00 jam (V7-10023)

- Support INSERT SQL query dari array dengan FOR ROWS dan ATOMIC options (V7-10105)
- JCL SORT - tingkatkan TranscodeTool untuk mendukung OUTREC dengan IFTHEN (V7-10124)
- JCL SORT - menambahkan dukungan untuk kata kunci DATE dalam perintah OUTREC (V7-10125)
- JCL - menambahkan dukungan prosedur In-Stream (V7-10223)

Perbaikan

- Kumpulan data yang ditandai dengan disposisi “PASS” harus tersedia di semua langkah pekerjaan (V7-9504)
- Support JCL atribut SCHENV (V7-9570)
- Support KIRIM dengan opsi CTLCHAR (V7-9714)
- COBOL - Menangani charset pemisah garis yang berbeda dalam pernyataan ACCEPT (V7-9875)
- Hindari beberapa rollback (V7-9958)
- Izinkan penggunaan disposisi MOD untuk ditambahkan di akhir file GDG (V7-10031)
- Optimasi: PutAll refactoring (V7-10063)
- PutAll refactoring: menambahkan pagination (V7-10063)
- Jadikan batas waktu baca klien Jedis dapat dikonfigurasi (V7-10063)
- UseSsl dukungan untuk mode mandiri (V7-10114)
- Support EIBDS setelah berhasil membuka file (V7-10147)
- Support EIBDS setelah permintaan kontrol file (V7-10147)
- Tingkatkan dukungan CICS SYNCPOINT (V7-10187)
- BluesamRedisSerializer: masalah dengan MetadataPersistence (V7-10202)
- Support Redis AWS Secrets Manager untuk antrian TS (V7-10204)
- Support JCLBCICS untuk menyesuaikan ukuran nama DD (V7-10224)
- Menambahkan dukungan untuk jalur absolut dalam pernyataan IDCAMS DELETE (V7-10308)

AS400

Fitur baru

- Implementasi fitur bantuan untuk layar AS400 (V7-9673)

Perbaikan

- Jumlah catatan dalam INFDS (V7-9377)

Kemampuan transversal

Fitur baru

- Support for Runtime di EC2 untuk mengirim log ke Amazon CloudWatch (D87990246)
- Menambahkan titik akhir baru untuk mengambil metrik tentang batch, transaksi, dan JVM (D88393832)

Perbaikan

- Support sumber data AWS Secrets Manager untuk utilitas pgm (V7-9570)
- Menambahkan dukungan Db2 untuk DSNUTILB DISCARD (V7-9798)
- Support untuk menulis ke logger alih-alih aliran output sistem default dalam file SYSPRINT dan SYSPUNCH default (V7-10098)
- Support BluSam Redis cache dan mengunci properti koneksi di AWS Secrets Manager (V7-10238)
- Support untuk koneksi SSL pada rahasia AWS Db2 XA (V7-10258)
- Metadata yang diperbarui untuk IDCAMS REPRO dan VERIFY (V7-10281)
- Peningkatan IDCAMS Abend Manajemen Kode Pengembalian (V7-10307)

Alat modernisasi rilis 3.7.0

Topik

ZoS

Fitur baru

- PLI - Peningkatan penugasan untuk penampang array dan array dua dimensi (V7-9830)

AS400

Fitur baru

- Penanganan indikator tingkat kontrol (V7-9227)
- Support untuk parameter EXTNAME* INPUT (V7-9897)
- Enhanced Goto Rewriting: Support untuk tag yang terletak di pernyataan SELECT OTHER (V7-9973)
- Support REFSHIT DSPF kata kunci (V7-10049)

Perbaikan

- Peningkatan penanganan kata kunci deskripsi file EXTIND (* iNUX) (V7-7404)
- Peningkatan transformasi file SQLDDS (V7-7687)
- Objek file tidak lagi dihasilkan untuk file AS400 (V7-9062)
- Peningkatan penanganan kata kunci deskripsi file EXTDESC (V7-9268)
- Peningkatan penanganan %CHAR bawaan (V7-9311)
- Peningkatan dukungan untuk pagedown pada catatan terakhir tanpa SFLEND (V7-9322)
- Peningkatan dukungan untuk struktur data awalan (V7-9436)
- Support untuk dimensi didefinisikan dengan% SIZE (V7-9472)
- Support untuk menangani nama bidang PF dideklarasikan dalam tanda kutip ganda (V7-9557)
- Peningkatan operasi file - case insensitive (V7-9785)
- Support untuk bidang yang diinisialisasi ke*USER (V7-9806)
- Support untuk tipe COMP di AS400 (V7-9840)
- Peningkatan penguraian COBOL400 pada (Tidak) (V7-9922) InvalidKey
- Peningkatan penanganan operasi SCAN (V7-9971)
- Peningkatan dukungan dari GOTO opcode (V7-9973)
- Peningkatan penanganan operasi KECUALI (V7-9977)
- Dukungan awalan yang ditingkatkan (V7-10000)
- Support untuk panggilan MQ dalam RPG (V7-10007)
- Peningkatan %LOOKUP builtin (struktur data array yang dikunci) (V7-10022)
- Support untuk Close * Semua operasi (V7-10036)

- Support untuk UPDATE AS ROW CHANGE pernyataan SQLDDS (V7-10051)
- Peningkatan untuk menangani tipe nilai literal Panjang (V7-10073)
- Tata bahasa RPG yang ditingkatkan (penggunaan kata kunci INZ sebagai nama subrutin) (V7-10074)
- Tata bahasa RPG yang ditingkatkan untuk mendukung nilai numerik dengan bagian pecahan kosong (V7-10077)
- Peningkatan dukungan untuk bidang yang dibagikan antara CL dan file eksternal (V7-10081)
- Peningkatan dukungan untuk indikator kondisional DDS (V7-10084)
- Support untuk tipe biner DDS dengan program COBOL (V7-10100)
- Peningkatan tabrakan nama dengan linkage (V7-10109)
- Support untuk pencampuran prosedur utama dan ekspor (V7-10112)
- Peningkatan dukungan untuk DataStructure dalam sub-prosedur (V7-10113)
- Peningkatan dukungan CLEAR (V7-10126)
- Peningkatan dukungan DO loop (V7-10134)
- Support SQLTYPE dalam RPG Bebas Penuh (V7-10151)
- Peningkatan parsing kondisi pada kata kunci DDS (V7-10155)
- Peningkatan generasi DSL (V7-10163)
- Perbaikan untuk ProcessIndicators ketika kondisinya adalah ekspresi biner. (V7-10164)
- Peningkatan GoTos dengan kondisi Lain (V7-10168)
- Support untuk tipe Waktu dan Timestamp di DSPF (V7-10173)
- Peningkatan parsing garis kelanjutan untuk DDS (V7-10183)
- Dukungan COBOL untuk RENAMES FLD OF RECORD (V7-10195)
- Peningkatan penguraian indikator bersyarat pada bidang DSPF (V7-10221)
- Support parsing kata kunci DDS NOALTSEQ (V7-10288)
- Menu Bantuan Dukungan dan bidang tersembunyi (V7-10314)
- Peningkatan DSPF bantuan kata kunci pemeriksaan kewarasan (V7-10328)
- Tidak lagi menyebarkan semua kata kunci di bidang Ref (V7-10347)

Kemampuan transversal

Fitur baru

- Migrator Data - Menangani data CLOB (V7-9665)

Perbaikan

- Menyebarkan properti JCL SCHENV dari JOB ke definisi PROC GROOVY melalui (V7-10225) JobContext
- FrontEnd - Menyesuaikan ukuran jendela jika tidak ada batas (V7-10358)

Catatan rilis 3.6.0

Rilis AWS Blu Age Runtime dan Modernization Tools ini menyediakan fitur baru untuk migrasi lama ZoS dan AS400, terutama berorientasi pada perluasan mekanisme dukungan CICS, melengkapi kemampuan JCL, mengoptimalkan kinerja dalam fitur bersamaan dan volume tinggi, dan menambahkan kemampuan. multi-data-source Beberapa perubahan utama dalam rilis ini adalah:

- Peningkatan penanganan file dinamis JCL, perluasan pernyataan saat ini dan pengelolaan kumpulan data gabungan, eksekusi beberapa pernyataan dalam satu blok, dan transfer data dari batch ke program.
- Dukungan yang ditingkatkan dari beberapa perintah CICS, termasuk penyelidikan untuk beberapa jenis sumber daya CICS.
- Kemampuan untuk memiliki database yang berbeda saat menggunakan Blu Age Runtime Utilities, paling cocok untuk skenario ketika data bisnis didistribusikan di berbagai sumber.

Untuk informasi selengkapnya tentang perubahan yang disertakan dalam rilis ini, lihat bagian berikut.

Rilis runtime 3.6.0

Topik

ZoS

Fitur baru

- JCL - DynamicFileBuilder - Manajemen penanganan file yang ditingkatkan (V7-9408)
- Konversi format yang ditingkatkan pada beberapa fungsi SQL DB2 bawaan saat memanggil utilitas INFUTILB UNLOAD (V7-9554)
- Penugasan array multi-dimensi PLI yang ditingkatkan (V7-9592)
- Penanganan sysout redirect ke file (V7-9992)

Perbaikan

- Tambahkan pemicu prosedur tersimpan untuk DB2 RDBMS (V7-9155)
- SORT menangani konversi ke format PDF (V7-9286)
- JCL/GROOVY - Tingkatkan pernyataan REPRO untuk mendukung kumpulan data DUMMY (V7-9424)
- Tingkatkan dukungan CICS UNLOCK (V7-9606)
- Menangani ukuran nilai default untuk Union (V7-9648)
- JCL/GROOVY menangani terminasi/disposisi yang berbeda dalam kumpulan data gabungan (V7-9653)
- Jadikan PageSize dapat dikonfigurasi untuk kumpulan data blusam (V7-9680)
- DSNUTIL - memungkinkan pemuatan 24:00:00 sebagai WAKTU yang valid di DB2LUW (V7-9697)
- Mendukung perbandingan NILAI TINGGI (0xff) NumberUtils di.ne ()/ NumberUtils.eq () (V7-9731)
- JCL/GROOVY - mendukung DO... KEMUDIAN kata kunci dalam klausa IDCAMS IF-THEN-ELSE untuk mengeksekusi beberapa pernyataan dalam satu blok (V7-9750)
- JHDB tidak valid disebut program di luar JHDB (V7-9782) BatchRunner
- Mendukung karakter spasi putih di kartu kontrol SORT OUTFIL (V7-9808)
- Tingkatkan dukungan CICS READ PREV (V7-9845)
- Meningkatkan akses bersamaan untuk indeks dataset (V7-9864)
- Meningkatkan dukungan CICS REWRITE (V7-9873)
- COBOL - dukungan untuk SYSIN multiline dalam pernyataan ACCEPT untuk meneruskan data dari batch (JCL) ke program (COBOL) (V7-9875)
- Groovy - Penanganan yang lebih baik ConcatenatedFileConfiguration pada langkah pembuatan file (V7-9876)
- IDCAMS UTILITY - Penanganan pernyataan DEFINE PATH (V7-9878)

- SORT BUILD - Sesuaikan opsi TRAN dan tangani blanko implisit (V7-9925)
- Tingkatkan CICS DELETE dengan dukungan opsi GENERIC (V7-9939)
- Meningkatkan dukungan CICS STARTBR dan ENDBR (V7-9952)
- Meningkatkan kinerja dekat pada akses bersamaan (V7-9953)
- Tingkatkan penanganan status file saat mulai (V7-9991)
- Groovy - Izinkan panggilan getDisposition ()/()/getNormalTermination() on getAbnormalTermination ConcatenatedFileConfiguration (V7-10012)

AS400

Fitur baru

- Mendukung indikator eksternal pada kata kunci COMMIT (V7-6035)
- Setel ulang loop ReadC setelah penulisan SFLCTL (V7-8061)
- Support Indikator LR di CALL (V7-9250)
- Tambahkan tipe baru bidang dinamis (split) untuk menangani bidang input pada beberapa baris (V7-9370)
- Support file primer/sekunder (V7-9390)
- Area Data Lokal sekarang diteruskan ke pekerjaan yang disebut saat mengirimkan pekerjaan (V7-9775)
- Support QTEMP untuk area data dan dukungan penciptaan nilai datarea. (V7-9916)
- Kontrol Komitmen - dukungan untuk mengaktifkan/menonaktifkan kontrol komitmen (V7-9956)
- Mendukung indikator eksternal pada kata kunci COMMIT

Perbaikan

- Tingkatkan tampilan nilai 0 dan EDTWRD (V7-8933)
- Support dari kata kunci DSPF “CHKMSGID” (V7-9125)
- Transaksi komit SQL setelah penghentian batch (V7-9232)
- Meningkatkan dukungan kata kunci EKSPOR dan IMPOR untuk bidang dan struktur data (V7-9265)
- Support huruf kecil di DateHelper (V7-9461)

- Dukungan konversi * CYMD ke* ISO (numerik) (V7-9488)
- Tingkatkan pegangan %len bawaan untuk bidang yang bervariasi (sisi kiri dan kanan ekspresi) (V7-9733)
- Meningkatkan dukungan untuk fungsi bawaan '%LOOKUPXX' XX (“LE”, “LT”, “GE”, “GT”) (V7-10064)

Kemampuan transversal

Fitur baru

- CICS - Tingkatkan transaksi Inquire untuk status opsi (V7-9712)
- JCL - Meningkatkan Beban untuk sysprint dengan file keluar sistem (V7-9797)
- CICS - Meningkatkan INQUIRE TSQUEUE (V7-9823)
- CICS - Tingkatkan terminal Inquire untuk userid opsi (V7-9906)

Perbaikan

- Tingkatkan pegangan perbandingan dengan blank (V7-8047)
- Tingkatkan logging untuk Jics dan BluSam (V7-8847)
- Support BMS atribut diperpanjang SOSI dan simbol terprogram F8 untuk bidang dinamis (V7-8857)
- Menangani buffer overflow dalam parameter program (V7-9138)
- Tingkatkan konkurensi penulisan utas untuk registri kunci Blusam (V7-9505)
- Mendukung beberapa konfigurasi sumber data untuk Utility-PGM (V7-9570)
- Mode penguncian tingkat rekam Blusam saja (V7-9626)
- Pastikan persistensi metadata menolak restart server (V7-9748)
- Tingkatkan pembersihan DAO pada pengecualian (Tutup Browser) (V7-9790)
- Support DummyFile untuk INFUTILB SYSPUNCH (V7-9799)
- Tingkatkan dukungan untuk nilai negatif pada NumericEditedType (V7-9935)

Alat modernisasi rilis 3.6.0

Topik

- ZoS

- AS400
- Kemampuan transversal

ZoS

Fitur baru

- JCL - Meningkatkan logging untuk akhir prosedur (V7-8509)
- PL1 - Tingkatkan pembuatan tas untuk tipe data PakedLong (V7-8917)
- JCL - Tingkatkan logging untuk akhir prosedur ketika file berisi penanda “akhir”//(V7-9509)
- PL1 - Meningkatkan dukungan untuk GET EDIT dengan Fixed-point dan SYSIN stream (V7-9593)
- DB2 - Meningkatkan dukungan untuk tipe VARGRAPHIC DB2 (V7-9809)
- CICS - Meningkatkan perintah QUERY SECURITY untuk opsi LOGMESSAGE (V7-9969)
- PL1 - Meningkatkan pembuatan tas untuk charg/Chargraphic built-in (V7-9989)

Perbaikan

- PL1- Meningkatkan dukungan untuk kata kunci INCLUDEX (V7-9588)
- PL/I - Menangani kata kunci CHARGRAPHIC sebagai parameter yang valid dari setiap panggilan metode (V7-9589)
- Meningkatkan resolusi variabel host PL1 bila dinamai dengan karakter tertentu @ # $ §. (V7-9654)
- COBOL - Support dari C01... C12 & S01... S05 kata kunci sebagai parameter pernyataan WRITE ADVANCING pada langkah parsing (V7-9669)

AS400

Fitur baru

- Mendukung transformasi SQL-DDS di Analyzer (V7-7687)
- Mengotomatiskan deteksi file SQL-DDS (V7-7687)
- Implementasi preprocessing SQL-DDS (V7-7687)
- Support ALIGN kata kunci (V7-9254)
- Dukungan ExtName untuk DSPF dan multi-dim array (V7-9663)

- InvalidKey Pernyataan Support pada COBOL WRITE (V7-9793)

Perbaikan

- Peningkatan pada opcode TESTB (V7-8865)
- Meningkatkan dukungan DECFMT pada fokus (V7-8933)
- Penanganan indikator yang dihasilkan pada MOVE (V7-9224)
- Meningkatkan dukungan template kata kunci untuk bidang dan struktur data (V7-9278)
- Peningkatan LIKEDS (DS didefinisikan menggunakan LIKEDS secara otomatis memenuhi syarat) (V7-9302)
- COBOL - Meningkatkan generasi struktur indikator (V7-9423)
- Parameter Const dalam prototipe tidak hanya-baca (V7-9437)
- Tingkatkan kata kunci EDTCDE dengan kode edit “Y” (V7-9443)
- Support generasi bidang* RUTINE di PSDS dan INFDS (V7-9487)
- Tingkatkan bidang penulisan ulang XXX menjadi mandiri (nilai default hilang saat menulis ulang) (V7-9522)
- Meningkatkan Support dari kata kunci DSPF (V7-9658)
- Menangani nilai default ZEROES pada biner (V7-9666)
- Mendukung penunjuk implisit (V7-9719)
- Meningkatkan penanganan panggilan bawaan %size dengan satu parameter (V7-9730)
- Meningkatkan penanganan referensi struktur data dalam panggilan bawaan (% ELEM) (V7-9736)
- Tingkatkan penanganan panjang yang ditandatangani untuk bidang dengan referensi LIKE dalam spesifikasi definisi (V7-9738)
- Perbaikan pada REWRITE (V7-9791)
- Peningkatan generasi indeks dari file DDS (V7-9803)
- Tingkatkan ketahanan mappers dengan nilai numerik yang tidak valid (V7-9813)
- Tingkatkan pembuatan file SQLModel dan AllIndexes (V7-9818)
- Tingkatkan dukungan DS yang memenuhi syarat (V7-9863)
- Meningkatkan dukungan LOOKUP (dengan bidang mandiri SEPERTI DS dalam parameter) (V7-9961)
- Tingkatkan LIKE pada indikator (V7-9985)
- Penanganan indikator yang dihasilkan pada MVR (V7-9995)

- Support karakter N dengan tilde (V7-10021)
- Tingkatkan pembuatan file DDL modern dari file lama SQLDDS (V7-10067)

Kemampuan transversal

Fitur baru

- Sesuaikan lokasi sumber daya dengan properti yl (D88816105)
- COBOL - Support pernyataan EXIT PERFORM untuk keluar dari PERFORM inline tanpa menggunakan GO TO /PERFORM... MELALUI (V7-9582)
- Menentukan pengkodean warisan default untuk dipertimbangkan ke dalam metadata global. (V7-9883)

Perbaikan

- Tingkatkan pembuatan topeng (V7-9602)
- Tingkatkan pemanasan konteks (V7-9621)
- Jadikan utas Charset CUSTOM930 aman. (V7-9674)
- Perbaikan pada MOVEA (V7-9773)

Catatan rilis 3.5.0

Rilis AWS Blu Age Runtime dan Modernization Tools ini menyediakan fitur baru untuk migrasi warisan ZoS dan AS400, terutama berorientasi pada kumpulan data dan optimasi pesan, serta kemampuan Java yang diperluas sebagai aset yang dihasilkan dari proses transformasi. Beberapa perubahan utama dalam rilis ini adalah:

- Kemampuan migrasi program CL ke Java selain fitur skrip asyik yang sudah ada sebelumnya, untuk memfasilitasi integrasi dengan program modern lainnya, dan untuk menyederhanakan kurva pembelajaran pelanggan dengan menyatukan bahasa pemrograman yang dihasilkan.
- Pengurangan waktu dan optimalisasi kinerja beban kumpulan data di Redis dengan fitur massal data baru.
- Kemampuan untuk mengoperasikan dan meneruskan kumpulan data dalam langkah-langkah pekerjaan untuk memodernisasi perilaku kumpulan data tradisional.
- Perpanjangan migrasi SQL untuk mendukung file input VB dan migrasi sederhana Java 11.

- Beberapa mekanisme baru untuk integrasi yang lebih cepat dengan IBM MQ termasuk header tambahan, dukungan GET/PUT yang diperluas, dan pengambilan metadata antrian secara otomatis.
- REST Endpoint untuk metadata dataset dan dataset impor dari bucket S3.

Untuk informasi selengkapnya tentang perubahan yang disertakan dalam rilis ini, lihat bagian berikut.

Rilis runtime 3.5.0

Topik

ZoS

Fitur baru

- JCL SORT - Menangani hamparan kata kunci baru (V7-9409)
- ZOS COBOL - meningkatkan dukungan arang mengambang (V7-9404)
- Port RedisJics TSqueue ke RedisTemplate & ListOperations (V7-9212)
- ZOS JCL - tingkatkan jalur direktori sementara dengan direktori file jika didefinisikan melalui UserDefinedParameters (V7-9012)
- Tangani FUNGSI ORD-MAX dengan SEMUA (semua item array) (V7-9366)
- Kunci awalan dan dapat dibaca manusia sekarang digunakan saat menyimpan TS Queues di Redis (V7-9212)
- Tambahkan titik akhir set data untuk blusam API
- JCL - TAMBAHKAN dukungan untuk pekerjaan batch dengan nama yang melibatkan karakter khusus seperti # (V7-9136)
- Pengambilan TSModel sekarang dilakukan dengan kuat sesuai permintaan (V7-9212)

Perbaikan

- Dukungan INCLUDE non-versi dalam file LNK (V7-6022)

- MQ - Meningkatkan dukungan encoding (V7-9652)
- Meningkatkan dukungan untuk byte ganda atau charset campuran untuk berbagai jenis karakter (V7-9596)
- JCL - Support konfigurasi FilesDirectory di IDCAMS menghapus pernyataan NONVSAM (V7-9609)
- Mendukung mode massal untuk kumpulan data ESDS dan RRDS yang dimuat dari file (V7-8639)
- Tangani pembukaan ESDS kosong dalam mode input. (V7-9287)
- Tingkatkan pernyataan DEFINE CLUSTER dengan dukungan singkatan ORD/UNORD (V7-9451)
- BluSam Peningkatan kinerja kunci Redis (V7-8639)
- Tingkatkan pernyataan DEFINE CLUSTER untuk mendukung RECORDSIZE yang disediakan dalam lingkup argumen DATA () (V7-9337)
- Menambahkan dukungan atribut BUFFERSPACE/UNIQUE pada pernyataan DEFINE CLUSTER (V7-9419)
- Tingkatkan operasi BluSam baca untuk kumpulan data catatan panjang variabel. (V7-9391)
- ALAMAT CICS dengan benar mewakili CWA yang hilang sebagai nol (V7-9491)
- Hapus Tulis yang tidak perlu di kunci akhir (V7-8639)
- Tangani injeksi template cache Redis dalam cache (V7-9510)
- Dekode dengan benar parameter BPXWDYN (V7-9417)
- Peningkatan konsumsi ekspor LISTCAT (V7-9201)
- Dukungan karakter yang tidak dapat dicetak dalam nama BluSam TS Queues (V7-9212)
- Tangani menerima bangunan Peta untuk bidang dengan mapset null (V7-9486)
- Tingkatkan operasi BluesamRelativeFile hapus dan tulis ulang untuk mode akses dinamis. (V7-8989)

AS400

Fitur baru

- Tambahkan fitur untuk menghasilkan file CL sebagai program Java melalui pivot DS/STM standar (V7-9427)
- Support Input File dengan mode ADD (V7-9378)
- Peningkatan urutan pengurutan dan manajemen pengambilan untuk mendukung perintah cl OPNQRYF (Open Query File) dan menambahkan dukungan parameter SHARE di. OverrideItem (V7-9364)

Perbaikan

- Support SFLNXTCHG aktif (V7-8061) UpdateSubfile
- Ubah ruang lingkup konteks CL saat menjalankan perintah CL (V7-9624)
- Menangani kode pengembalian untuk program BPXWDYN (V7-9417)
- Hapus monitor lokal. (V7-9624)
- Support dari kata kunci DSPF RTNCSRLOC (V7-9389)
- setOnGreaterOrEqual() tidak mengatur Sama dengan 1 (V7-9342)
- Perbarui cache bidang pada UpdateSubfileRecord (V7-9376)
- Meningkatkan Support SFLNXTCHG (V7-8061)

Kemampuan transversal

Fitur baru

- Abaikan awalan G pada string grafis literal. (V7-9420)
- ZOS COBOL - Meningkatkan dukungan FIEDL.initialize () untuk beberapa struktur khusus (V7-9485)
- Izinkan inisialisasi konteks secara asinkron untuk meningkatkan kinerja startup program (V7-9446)
- SQL Release secara eksplisit pernyataan persiapan terbuka dan. ResulSet (V7-9422)
- Meningkatkan JMS MQ - mendukung MQRFH2 untuk MQ PUT/V7-7085 - dukungan manajer antrian default (V7-9400)
- Manajemen SQL - Aktifkan konversi Lambda pada parameter untuk perintah SET (V7-9492)
- ZOS MQ JMS - Tambahkan dukungan ke MQCOMIT dan MQBACK (V7-9399)
- ZOS IBMMQ - Meningkatkan dukungan untuk MQINQ (V7-9544)
- Tangani operasi CONCAT dengan byte alih-alih string saat menggunakan pengkodean byte ganda. (V7-8932)
- ZOS IBMMQ - Meningkatkan dukungan perintah PUT dengan opsi SET_ALL_CONTEXT (V7-9544)

Perbaikan

- Tangani nama file gdg dengan karakter $ (V7-9066)
- SQL Diagnostic mengembalikan 1 sebagai klausa NUMBER ketika pernyataan SQL sebelumnya berhasil. (V7-9410)

- Garis besar untuk bidang dengan panjang non nol (V7-7536)
- Mendukung fungsi GRAFIS PL1 bawaan (V7-9245)
- MQ - Tambahkan dukungan versi untuk pengaturan bidang MQGMO (V7-9500)
- JMS MQ GET - Pesan mengembalikan peningkatan DataLength (V7-9502)
- Atur sqlerrd (3) dengan jumlah item yang diambil dalam konteks ROWSET. (V7-9371)

Alat modernisasi rilis 3.5.0

Topik

ZoS

Fitur baru

- ZOS PLI - Support indeks asterisk dalam penugasan dengan ekspresi biner (V7-9178)
- JCL to BatchScript - A “//” menandai akhir eksekusi pekerjaan (V7-9304)
- ZOS PLI - tingkatkan dukungan karakter mengambang dan masuk tipe yang diedit numerik (V7-8982)
- COBOL - Support fungsi SUM bawaan (V7-9367)
- JCL- opsional, komentari kode mati setelah pernyataan nol (//) (V7-9202)
- JCL- Support operator '|' dalam pernyataan kondisi (V7-9499)
- PL/I - Komentar arahan prakompilasi pada langkah preprocessing untuk mencegah pengecualian parsing (V7-9507)

Perbaikan

- Menangani definisi Stream dengan pembatas (V7-9615)
- Meningkatkan penanganan ekspor LISTCAT. (V7-9201)
- PL/I - Peningkatan untuk mendukung argumen 'null' implisit (V7-9204)

AS400

Fitur baru

- Support dari kata kunci DDS CONCAT (V7-9439)
- Refactor kode java yang dihasilkan untuk kata kunci DSPF. (V7-7700)
- Support Memvariasikan kata kunci pada bidang dalam definisi struktur data (V7-9029)

Perbaikan

- Meningkatkan parsing hubungan logis DAN/ATAU (V7-9352)
- COBOL Meningkatkan pemetaan antara vo dan DSentity (V7-9449)
- Menampilkan nilai kosong jika input numerik difokuskan (V7-9374)
- Variabel lokal di SQL Declare Cursor (V7-9456)
- Masalah lingkup dengan DS kosong (V7-9466)
- Memangkas garis setelah col 80 sebelum parsing (V7-9632)
- Tingkatkan pegangan referensi lapangan dan panggilan bawaan dalam kata kunci (DIM, LIKE,...) dalam spesifikasi definisi (V7-9358)
- Mendukung komentar SQL (--) (V7-9632)
- FullFree penguraian, ketik Tanggal/Waktu/Stempel Waktu (V7-9542)
- Sertakan SQLCA dari FullFree parsing (V7-9333)
- Meningkatkan Support of Control Level. (V7-9610)
- Tangani perbandingan DS dengan* BLANKS (V7-9668)
- Meningkatkan dukungan beberapa indikator di DDS (V7-9318)
- Meningkatkan dukungan beberapa program DSPF (V7-9657)
- Tingkatkan pegangan bidang dengan LIKE (kasus struktur data yang disukai dan kasus struktur data yang disukai dalam array) (V7-9213)
- RPG gratis, Menangani kelanjutan secara literal (V7-9686)
- Meningkatkan Support dari catatan akhir program (V7-9452)
- Support frase LINKAGE dalam pernyataan CALL. (V7-9685)
- Kode operasi CASXX (CASBB tanpa grup CASXX) (V7-9357)
- Tingkatkan penguraian FullFree RPG (V7-9457)
- Built-in %LEN tidak mendukung DS sebagai argumen (V7-9267)

- Perbaikan MOVEA ketika faktor 2 adalah *ALL'X... ' (V7-9228)
- Support menetapkan dengan bidang RENAME (V7-9385)

Kemampuan transversal

Fitur baru

- Alat SQL Migrator - Tambahkan opsi OID untuk panjang catatan variabel pada langkah pemuatan ebcdic. (V7-9380)
- Alat SQL Migrator - Dukungan untuk Java 11 pada opsi OID (V7-9599)

Perbaikan

- Meningkatkan dukungan untuk array bersarang (V7-9595)
- Ganti karakter â¬ dengan! dalam kasus â¬ didukung oleh pengkodean asli. (V7-9465)
- JCL - Dukungan penghentian normal PASS untuk berbagi kumpulan data antara langkah-langkah pekerjaan (V7-9504)
- Terapkan ON NULL ke definisi kolom pada ORACLE ketika berurusan dengan VARCHAR dan tipe kolom db nullable. (V7-9681)
- Meningkatkan kepatuhan injeksi Spring (V7-9635)

AWS Konsep Runtime Blu Age

Memahami konsep dasar AWS Blu Age Runtime dapat membantu Anda memahami bagaimana aplikasi Anda dimodernisasi dengan refactoring otomatis.

Topik

- AWS Arsitektur Tingkat Tinggi Blu Age Runtime
- AWS Struktur Usia Blu dari Aplikasi Modernisasi
- Penyederhanaan Data

AWS Arsitektur Tingkat Tinggi Blu Age Runtime

Sebagai bagian dari solusi AWS Blu Age untuk memodernisasi program warisan ke Java, AWS Blu Age Runtime menyediakan titik masuk berbasis REST terpadu untuk aplikasi modern, dan kerangka

eksekusi untuk aplikasi semacam itu, melalui perpustakaan yang menyediakan konstruksi warisan dan standarisasi organisasi kode program.

Aplikasi modern tersebut adalah hasil dari proses AWS Blu Age Automated Refactor untuk memodernisasi program mainframe dan midrange (disebut dalam dokumen berikut sebagai "warisan") ke arsitektur berbasis web.

Tujuan AWS Blu Age Runtime adalah reproduksi perilaku program warisan (isofungsionalitas), kinerja (sehubungan dengan waktu eksekusi program dan konsumsi sumber daya), dan kemudahan pemeliharaan program modern oleh pengembang Java, meskipun penggunaan lingkungan dan idiom yang sudah dikenal seperti tomcat, Spring, getters/setter, API fasih...

Topik

AWS Komponen runtime Blu Age

Lingkungan AWS Blu Age Runtime terdiri dari dua jenis komponen:

- Satu set perpustakaan java (file jar) sering direferensikan sebagai "folder bersama", dan menyediakan konstruksi dan pernyataan warisan.
- Satu set aplikasi web (file perang) yang berisi aplikasi web berbasis Spring yang menyediakan seperangkat kerangka kerja dan layanan umum untuk program modern.

Bagian berikut merinci peran kedua komponen ini.

AWS Perpustakaan Blu Age

Pustaka AWS Blu Age adalah satu set file jar yang disimpan dalam `shared/` subfolder yang ditambahkan ke classpath tomcat standar, sehingga membuatnya tersedia untuk semua program Java modern. Tujuan mereka adalah untuk menyediakan fitur yang tidak asli atau mudah tersedia di lingkungan pemrograman Java, tetapi khas lingkungan pengembangan warisan. Fitur-fitur tersebut diekspos dengan cara yang seakrab mungkin bagi pengembang Java (getter/setter, berbasis kelas, API yang lancar). Contoh penting adalah pustaka Data Simplifier, yang menyediakan tata letak

memori lama dan konstruksi manipulasi (ditemui dalam bahasa COBOL, PL1 atau RPG) ke program Java. Guci tersebut adalah ketergantungan inti dari kode Java modern yang dihasilkan dari program lama. Untuk informasi selengkapnya tentang Penyederhanaan Data, lihatPenyederhanaan Data.

Aplikasi web

Web Application Archives (Wars) adalah cara standar untuk menyebarkan kode dan aplikasi ke server aplikasi tomcat. Yang disediakan sebagai bagian dari runtime AWS Blu Age bertujuan menyediakan satu set kerangka kerja eksekusi yang mereproduksi lingkungan lama dan monitor transaksi (batch JCL, CICS, IMS...), dan layanan yang diperlukan terkait.

Yang paling penting adalah `gapwalk-application` (sering disingkat sebagai “Gapwalk”), yang menyediakan satu set titik masuk berbasis REST terpadu untuk memicu dan mengontrol transaksi, program, dan eksekusi batch. Untuk informasi selengkapnya, lihat AWS Blu Age Runtime API.

Aplikasi web ini mengalokasikan thread eksekusi Java dan sumber daya untuk menjalankan program modern dalam konteks yang dirancang. Contoh lingkungan yang direproduksi tersebut dirinci di bagian berikut.

Aplikasi web lain menambah lingkungan eksekusi (lebih tepatnya, ke “Program Registry” yang dijelaskan di bawah) program meniru yang tersedia untuk, dan dapat dipanggil dari, program warisan. Dua kategori penting seperti itu adalah:

- Emulasi program yang disediakan OS: Batch yang digerakkan oleh JCL terutama berharap dapat memanggil berbagai program manipulasi file dan database sebagai bagian dari lingkungan standar mereka. Contohnya termasuk`SORT`/`DFSORT`atau`IDCAMS`. Untuk tujuan ini, program Java disediakan yang mereproduksi perilaku tersebut, dan dapat dipanggil menggunakan konvensi yang sama dengan yang lama.
- “Driver”, yang merupakan program khusus yang disediakan oleh kerangka eksekusi atau middleware sebagai titik masuk. Contohnya adalah`CBLTDLI`, program COBOL mana yang dijalankan di lingkungan IMS bergantung pada untuk mengakses layanan terkait IMS (IMS DB, dialog pengguna melalui MFS, dll.).

Registri program

Untuk berpartisipasi dan memanfaatkan konstruksi, kerangka kerja, dan layanan tersebut, program Java yang dimodernisasi dari yang lama mematuhi struktur tertentu yang didokumentasikan. AWS Struktur Usia Blu dari Aplikasi Modernisasi Saat startup, AWS Blu Age Runtime akan mengumpulkan

semua program semacam itu dalam “Registry Program” yang umum sehingga mereka dapat dipanggil (dan saling memanggil) sesudahnya. Registry Program menyediakan kopling longgar dan kemungkinan dekomposisi (karena program memanggil satu sama lain tidak harus dimodernisasi secara bersamaan).

Lingkungan eksekusi

Lingkungan warisan dan koreografi yang sering ditemui tersedia:

- Batch berbasis JCL, setelah dimodernisasi ke program Java dan skrip Groovy, dapat dimulai dengan cara sinkron (pemblokiran) atau asinkron (terpisah). Dalam kasus terakhir, eksekusi mereka dapat dipantau melalui titik akhir REST.
- Subsistem AWS Blu Age menyediakan lingkungan eksekusi yang mirip dengan CICS melalui:
  - titik masuk yang digunakan untuk memulai transaksi CICS dan menjalankan program terkait sambil menghormati koreografi “run level” CICS,
  - penyimpanan eksternal untuk Definisi Sumber Daya,
  - satu set pernyataan mereproduksi `EXEC CICS` API lancar Java yang homogen,
  - satu set kelas pluggable mereproduksi layanan CICS, seperti Antrian Penyimpanan Sementara, Antrian Data Sementara atau akses file (beberapa implementasi biasanya tersedia, seperti Amazon Managed Service untuk Apache Flink, Amazon Simple Quue Service, atau RabbitMQ untuk TD Queues),
  - untuk aplikasi yang dihadapi pengguna, format deskripsi layar BMS dimodernisasi ke aplikasi web Angular, dan dialog “pseudo-percakapan” yang sesuai didukung.
- Demikian pula, subsistem lain menyediakan koreografi berbasis pesan IMS, dan mendukung modernisasi layar UI dalam format MFS.
- Selain itu, subsistem ketiga memungkinkan eksekusi program dalam lingkungan seperti iSeries, termasuk modernisasi layar yang ditentukan DSPF (Display File).

Semua lingkungan tersebut dibangun di atas layanan tingkat OS umum seperti:

- emulasi alokasi dan tata letak memori lama (Penyederhanaan Data),
- Reproduksi berbasis benang Java dari eksekusi “run unit” COBOL dan parameter passing mechanism (`CALL`statement),
- emulasi flat, gabungan, VSAM (melalui kumpulan perpustakaan Bluesam), dan organisasi Kumpulan Data GDG,

- akses ke penyimpanan data, seperti RDBMS (`EXEC SQL`pernyataan).

Tanpa kewarganegaraan dan penanganan sesi

Fitur penting dari AWS Blu Age Runtime adalah mengaktifkan Ketersediaan Tinggi (HA) dan skenario skalabilitas horizontal saat menjalankan program modern.

Landasan untuk ini adalah tanpa kewarganegaraan, contoh pentingnya adalah penanganan sesi HTTP.

Penanganan sesi

Tomcat berbasis web, mekanisme penting untuk ini adalah penanganan sesi HTTP (seperti yang disediakan oleh tomcat dan Spring) dan desain stateless. Karena desain tanpa kewarganegaraan tersebut didasarkan pada hal-hal berikut:

- pengguna terhubung melalui HTTPS,
- server aplikasi digunakan di belakang penyeimbang Beban,
- ketika pengguna pertama kali terhubung ke aplikasi itu akan diautentikasi dan server aplikasi akan membuat pengenal (biasanya dalam cookie)
- identifier ini akan digunakan sebagai kunci untuk menyimpan dan mengambil konteks pengguna ke/dari cache eksternal (penyimpanan data).

Manajemen cookie dilakukan secara otomatis oleh kerangka AWS Blu Age dan server tomcat yang mendasarinya, ini transparan bagi pengguna. Browser internet pengguna akan mengelola ini secara otomatis.

Aplikasi web Gapwalk dapat menyimpan status sesi (konteks) di berbagai penyimpanan data:

- Amazon ElastiCache untuk Redis
- Gugus Redis
- di peta memori (hanya untuk pengembangan dan lingkungan mandiri, tidak cocok untuk HA).

Ketersediaan Tinggi dan tanpa kewarganegaraan

Secara lebih umum, prinsip desain kerangka AWS Blu Age adalah tanpa kewarganegaraan: sebagian besar status non-sementara yang diperlukan untuk mereproduksi perilaku program

warisan tidak disimpan di dalam server aplikasi, tetapi dibagikan melalui “sumber kebenaran tunggal” eksternal yang umum.

Contoh status tersebut adalah Antrian Penyimpanan Sementara atau Definisi Sumber Daya CICS, dan penyimpanan eksternal yang khas untuk mereka adalah server yang kompatibel dengan REDIS atau database relasional.

Desain ini, dikombinasikan dengan load balancing dan sesi bersama, mengarah ke sebagian besar dialog yang dihadapi pengguna (OLTP, “Online Transactional Processing”) untuk didistribusikan antara beberapa “node” (di sini, contoh tomcat).

Memang pengguna dapat melakukan transaksi di server mana pun dan tidak peduli jika panggilan transaksi berikutnya dilakukan di server yang berbeda. Kemudian ketika server baru muncul (karena penskalaan otomatis, atau untuk mengganti server yang tidak sehat), kami dapat menjamin bahwa setiap server yang dapat dijangkau dan sehat dapat menjalankan transaksi seperti yang diharapkan dengan hasil yang tepat (nilai pengembalian yang diharapkan, perubahan data yang diharapkan dalam database, dll.).

AWS Struktur Usia Blu dari Aplikasi Modernisasi

Dokumen ini memberikan rincian tentang struktur aplikasi modern (menggunakan alat refactoring Modernisasi AWS Mainframe), sehingga pengembang dapat menyelesaikan berbagai tugas, seperti:

- menavigasi ke aplikasi dengan lancar.
- mengembangkan program kustom yang dapat dipanggil dari aplikasi modern.
- dengan aman memfaktorkan ulang aplikasi modern.

Kami berasumsi bahwa Anda sudah memiliki pengetahuan dasar tentang hal-hal berikut:

- konsep pengkodean umum yang lama, seperti catatan, kumpulan data, dan mode aksesnya ke catatan -- diindeks, berurutan --, VSAM, run unit, skrip jcl, konsep CICS, dan sebagainya.
- pengkodean java menggunakan kerangka Spring.
- Sepanjang dokumen, kami gunakan `short class names` untuk keterbacaan. Untuk informasi selengkapnya, lihat AWS Pemetaan nama Blu Age sepenuhnya memenuhi syarat untuk mengambil nama yang memenuhi syarat untuk elemen runtime AWS Blu Age dan Pemetaan nama pihak ketiga yang sepenuhnya memenuhi syarat untuk mengambil nama yang memenuhi syarat untuk elemen pihak ketiga.

- Semua artefak dan sampel diambil dari output proses modernisasi dari sampel aplikasi COBOL/CICS. CardDemo

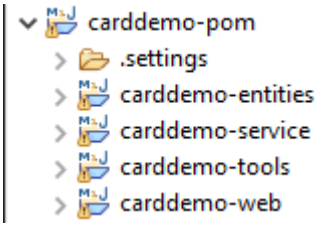
Topik

- Organisasi artefak
- Menjalankan dan memanggil program
- Tulis program Anda sendiri
- Pemetaan nama yang sepenuhnya memenuhi syarat

Organisasi artefak

AWS Aplikasi modern Blu Age dikemas sebagai aplikasi web java (.war), yang dapat Anda gunakan di server JEE. Biasanya, server adalah instance Tomcat yang menyematkan runtime AWS Blu Age Velocity, yang saat ini dibangun di atas kerangka kerja Springboot dan Angular (untuk bagian UI).

Perang mengumpulkan beberapa artefak komponen (.jar). Setiap jar adalah hasil kompilasi (menggunakan alat maven) dari proyek java khusus yang elemennya merupakan hasil dari proses modernisasi.



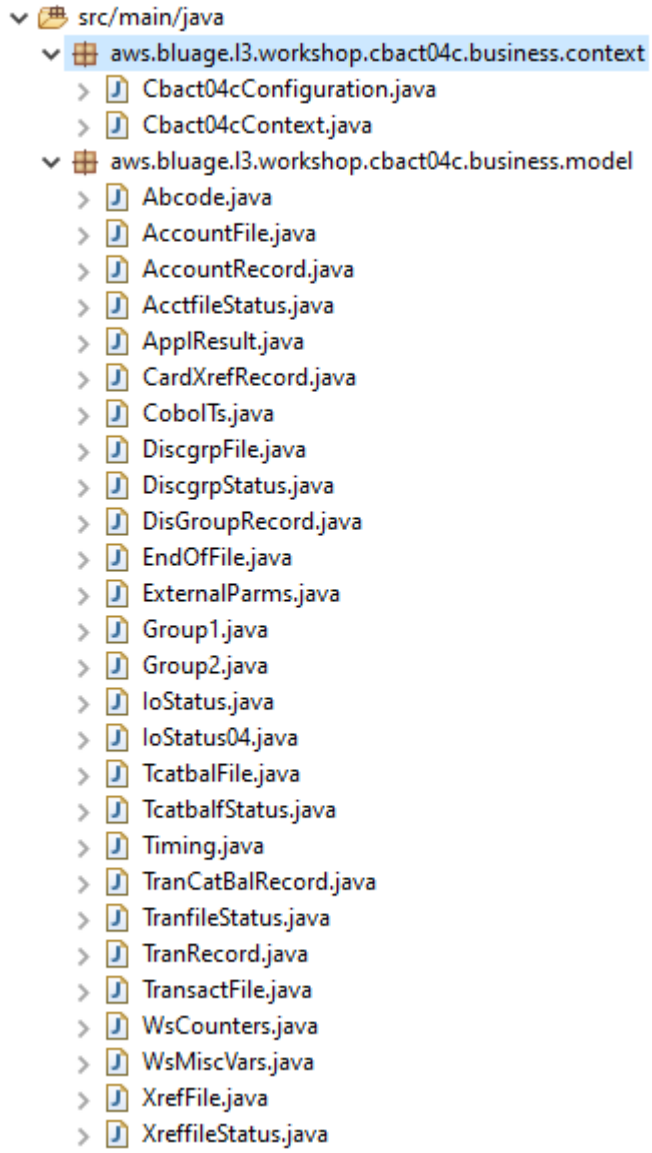
Organisasi dasar bergantung pada struktur berikut:

- Proyek entitas: berisi model bisnis dan elemen konteks. Nama proyek umumnya diakhiri dengan “-entity”. Biasanya, untuk program COBOL warisan tertentu, ini sesuai dengan modernisasi bagian I/O (kumpulan data) dan pembagian data. Anda dapat memiliki lebih dari satu proyek entitas.
- Proyek layanan: berisi elemen modernisasi logika bisnis warisan. Biasanya, pembagian prosedur program COBOL. Anda dapat memiliki lebih dari satu proyek layanan.
- Proyek utilitas: berisi alat dan utilitas umum bersama, yang digunakan oleh proyek lain.
- Proyek web: berisi modernisasi elemen terkait UI bila berlaku. Tidak digunakan untuk proyek modernisasi batch saja. Elemen UI ini bisa berasal dari peta CICS BMS, komponen IMS MFS, dan sumber UI mainframe lainnya. Anda dapat memiliki lebih dari satu proyek Web.

Entitas isi proyek

> **Note**
>
> Deskripsi berikut hanya berlaku untuk output modernisasi COBOL dan PL/I. Output modernisasi RPG didasarkan pada tata letak yang berbeda.

Sebelum refactoring, organisasi paket dalam proyek entitas terkait dengan program modern. Anda dapat mencapai ini dalam beberapa cara yang berbeda. Cara yang lebih disukai adalah dengan menggunakan kotak alat Refactoring, yang beroperasi sebelum Anda memicu mekanisme pembuatan kode. Ini adalah operasi lanjutan, yang dijelaskan dalam BluAge pelatihan. Untuk informasi lebih lanjut, lihat lokakarya Refactoring. Pendekatan ini memungkinkan Anda untuk mempertahankan kemampuan untuk membuat ulang kode java nanti, untuk mendapatkan keuntungan dari perbaikan lebih lanjut di masa depan, misalnya). Cara lain adalah dengan melakukan refactoring java biasa, langsung pada kode sumber yang dihasilkan, menggunakan pendekatan refactoring java apa pun yang mungkin ingin Anda terapkan - dengan risiko Anda sendiri.



Kelas terkait program

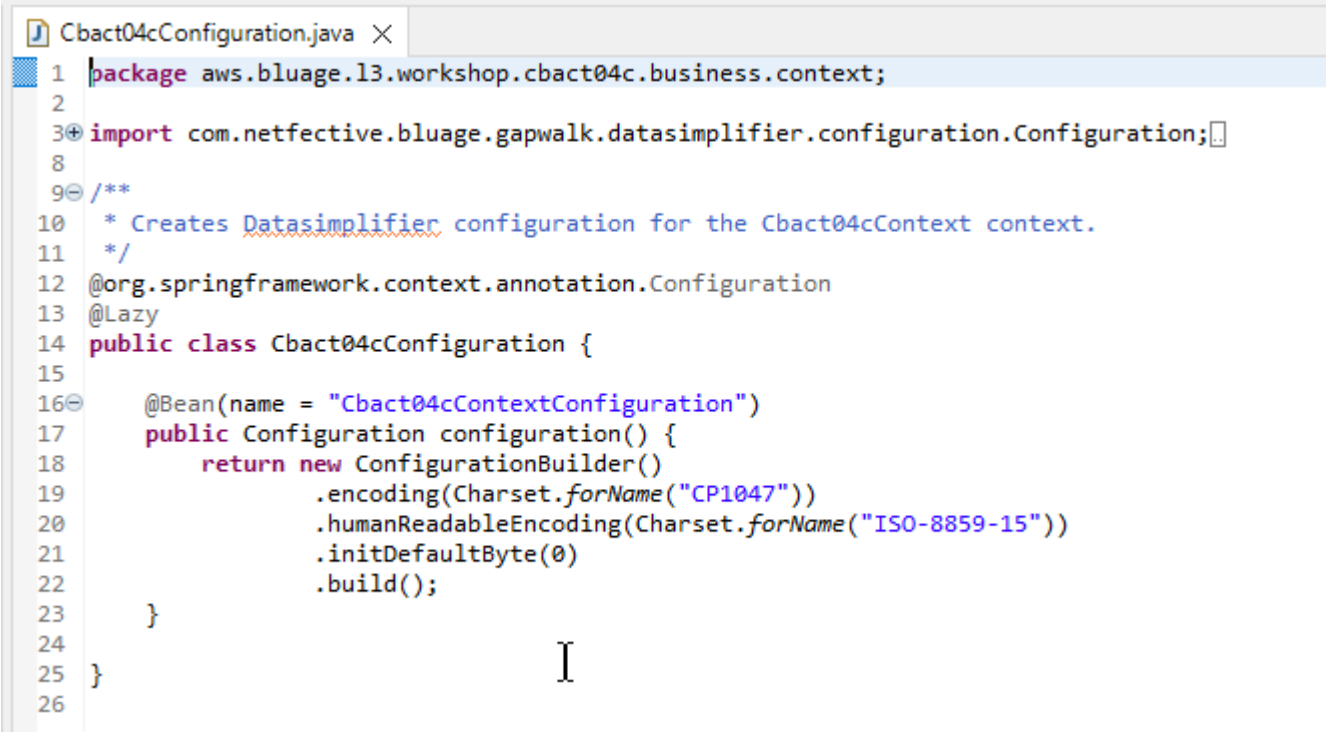
Setiap program modern terkait dengan dua paket, paket business.context dan business.model.

- *`base package`*`.`*`program`*`.business.context`

  Sub-paket business.context berisi dua kelas, kelas konfigurasi dan kelas konteks.

  - Satu kelas konfigurasi untuk program, yang berisi rincian konfigurasi spesifik untuk program yang diberikan, seperti set karakter yang akan digunakan untuk mewakili elemen data berbasis karakter, nilai byte default untuk elemen struktur data padding dan sebagainya. Nama kelas diakhiri dengan “Konfigurasi”. Hal ini ditandai dengan

`@org.springframework.context.annotation.Configuration` anotasi dan berisi metode tunggal yang harus mengembalikan `Configuration` objek setup dengan benar.



- Satu kelas konteks, yang berfungsi sebagai jembatan antara kelas layanan program (lihat di bawah) dan struktur data (`Record`) dan kumpulan data (`File`) dari sub-paket model (lihat di bawah). Nama kelas diakhiri dengan “Context” dan merupakan subclass dari `RuntimeContext` kelas.

Cbact04cContext.java

```
@Component("aws.bluage.l3.workshop.cbact04c.business.context.Cbact04cContext")
@Import({
    aws.bluage.l3.workshop.cbact04c.business.model.TcatbalFile.class
,   aws.bluage.l3.workshop.cbact04c.business.model.XrefFile.class
,   aws.bluage.l3.workshop.cbact04c.business.model.DiscgrpFile.class
,   aws.bluage.l3.workshop.cbact04c.business.model.AccountFile.class
,   aws.bluage.l3.workshop.cbact04c.business.model.TransactFile.class
})
@Lazy
@Scope("prototype")
public class Cbact04cContext extends JicsRuntimeContext {

    @Autowired
    private TcatbalFile tcatbalFile;

    @Autowired
    private XrefFile xrefFile;

    @Autowired
    private DiscgrpFile discgrpFile;

    @Autowired
    private AccountFile accountFile;

    @Autowired
    private TransactFile transactFile;

    private IndexedFile tcatbalFileFile;

    private IndexedFile xrefFileFile;

    private IndexedFile discgrpFileFile;

    private IndexedFile accountFileFile;

    private SequentialFile transactFileFile;

    private TranCatBalRecord tranCatBalRecord;
    private TcatbalfStatus tcatbalfStatus;
    private CardXrefRecord cardXrefRecord;
```

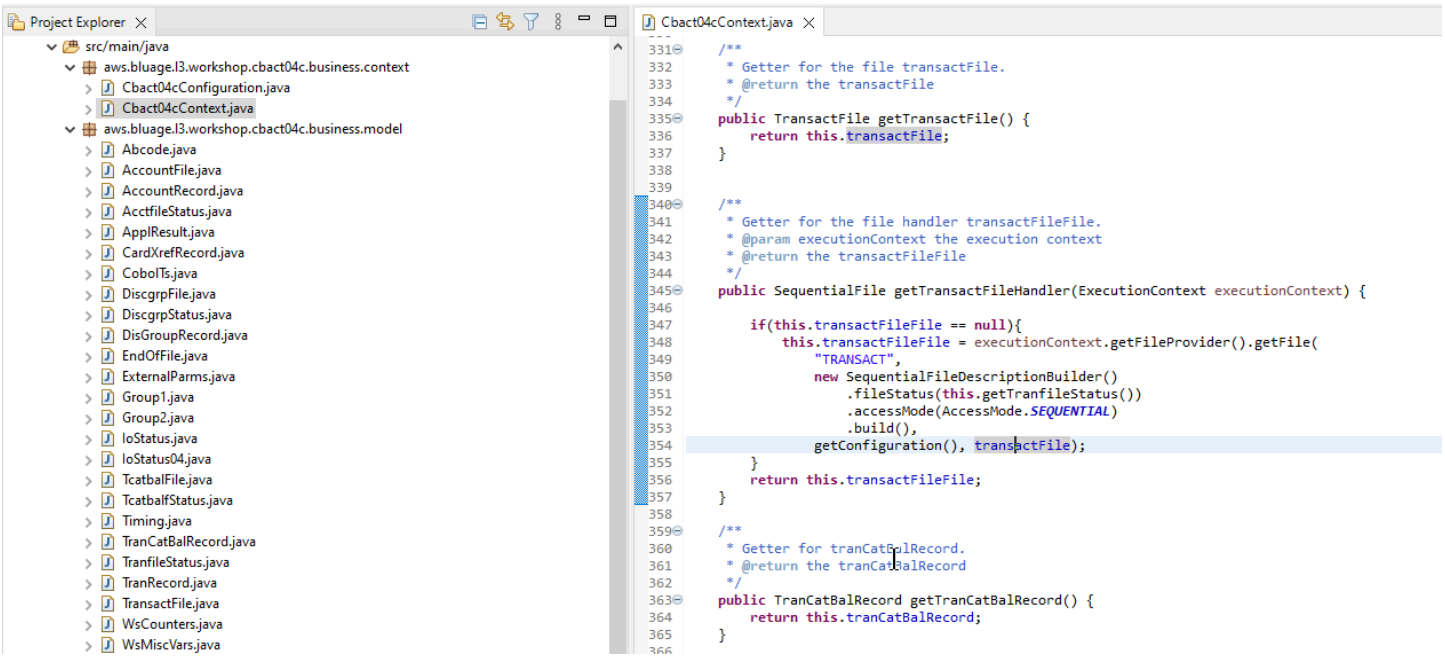
- *base package*.*program*.business.model

  Sub-paket model berisi semua struktur data yang dapat digunakan oleh program yang diberikan. Misalnya, setiap struktur data COBOL tingkat 01 sesuai dengan kelas dalam sub-paket model (struktur data tingkat bawah adalah properti dari struktur tingkat 01 yang mereka miliki).
  Untuk informasi lebih lanjut tentang bagaimana kita memodernisasi 01 struktur data, lihat. Penyederhanaan Data

DiscgrpFile.java

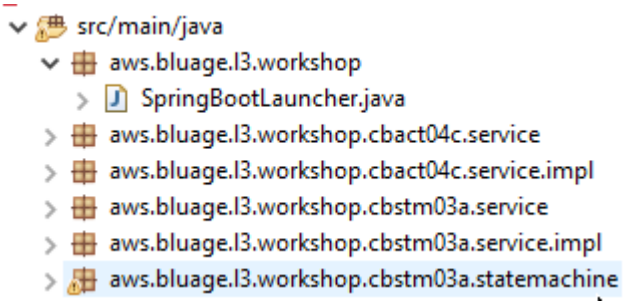
```java
package aws.bluage.l3.workshop.cbact04c.business.model;

import com.netfective.bluage.gapwalk.datasimplifier.configuration.Configuration;
import com.netfective.bluage.gapwalk.datasimplifier.data.structure.Elementary;
import com.netfective.bluage.gapwalk.datasimplifier.data.structure.Group;
import com.netfective.bluage.gapwalk.datasimplifier.entity.ElementaryRangeReference;
import com.netfective.bluage.gapwalk.datasimplifier.entity.RangeReference;
import com.netfective.bluage.gapwalk.datasimplifier.entity.RecordEntity;
import com.netfective.bluage.gapwalk.datasimplifier.metadata.type.AlphanumericType;
import com.netfective.bluage.gapwalk.datasimplifier.metadata.type.ZonedType;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.context.annotation.Lazy;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;

/**
 * Data simplifier file DiscgrpFile.
 *
 * <p>About 'fdDiscgrpRec' field, <br>uml entity: aws.bluage.l3.workshop.cbact04c.business.model.FdDiscgrpRec
 * <br></p>
 *
 */
@Component("aws.bluage.l3.workshop.cbact04c.business.model.DiscgrpFile")
@Lazy
@Scope("prototype")
public class DiscgrpFile extends RecordEntity {

    private final Group root = new Group(getData());
    private final Group fdDiscgrpRec = new Group(root);
    private final Group fdDiscgrpKey = new Group(fdDiscgrpRec);
    private final Elementary fdDisAcctGroupId = new Elementary(fdDiscgrpKey,new AlphanumericType(10));
    private final Elementary fdDisTranTypeCd = new Elementary(fdDiscgrpKey,new AlphanumericType(2));
    private final Elementary fdDisTranCatCd = new Elementary(fdDiscgrpKey,new ZonedType(4, 0, false));
    private final Elementary fdDiscgrpData = new Elementary(fdDiscgrpRec,new AlphanumericType(34));
```

Semua kelas memperluas `RecordEntity` kelas, yang mewakili akses ke representasi catatan bisnis. Beberapa catatan memiliki tujuan khusus, karena mereka terikat pada a`File`. Pengikatan antara a `Record` dan a `File` dibuat dalam FileHandler metode * yang sesuai yang ditemukan di kelas konteks saat membuat objek file. Misalnya, daftar berikut menunjukkan bagaimana terikat ke TransactFile `Record` (dari sub-paket model). TransactfileFile `File`



Isi proyek layanan

Setiap proyek layanan dilengkapi dengan aplikasi Springboot khusus, yang digunakan sebagai tulang punggung arsitektur. Hal ini diwujudkan melalui kelas bernama`SpringBootLauncher`, terletak di paket dasar dari layanan sumber java:



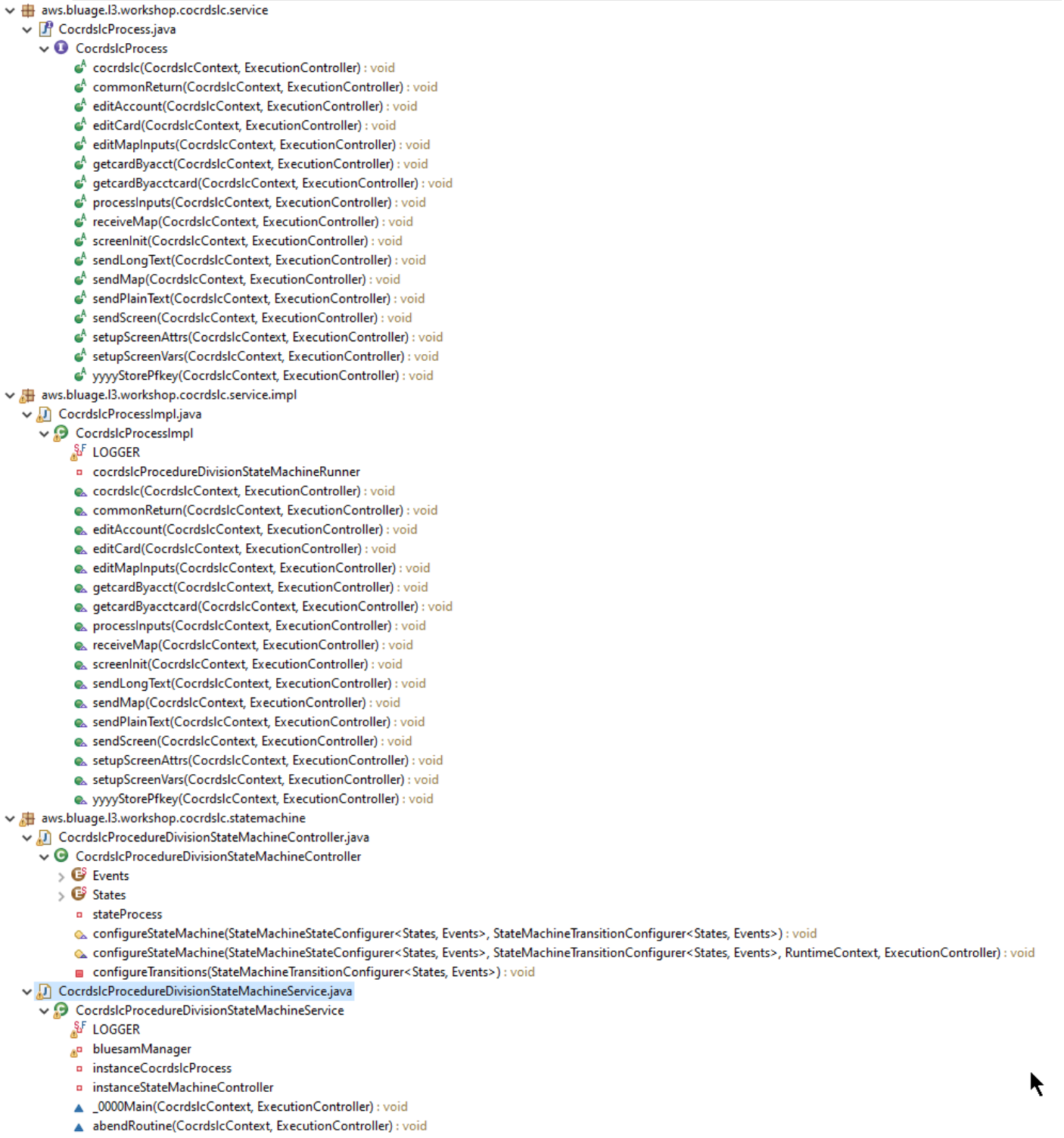
Kelas ini terutama bertanggung jawab untuk:

- membuat perekat antara kelas program dan sumber daya yang dikelola (sumber data/manajer transaksi/pemetaan kumpulan data/dll...).
- menyediakan `ConfigurableApplicationContext` untuk program.
- menemukan semua kelas yang ditandai sebagai komponen pegas (`@Component`).
- memastikan program terdaftar dengan benar di `ProgramRegistry` - lihat metode inisialisasi yang bertanggung jawab atas pendaftaran ini.

```
/**
 * Initialization method called when the spring application is ready.
 * Register all programs and services to the gapwalk shared context.
 * @param event the application ready event
 */
@EventListener
public void initialize(ApplicationReadyEvent event) {
    Map<String, ProgramContainer> programContainers = event.getApplicationContext().getBeansOfType(ProgramContainer.class);
    programContainers.values().forEach(ProgramRegistry::registerProgram);
    Map<String, ServiceContainer> serviceContainers = event.getApplicationContext().getBeansOfType(ServiceContainer.class);
    serviceContainers.values().forEach(ServiceRegistry::registerService);
}
```

Artefak terkait program

Tanpa refactoring sebelumnya, output modernisasi logika bisnis diatur pada dua atau tiga paket per basis program warisan:



Kasus yang paling lengkap akan memiliki tiga paket:

- *`base package`*`.`*`program`*`.service`: berisi antarmuka bernama Program Process, yang memiliki metode bisnis untuk menangani logika bisnis, melestarikan aliran kontrol eksekusi warisan.

- *base package*`.`*program*`.service.impl`: berisi kelas bernama ProgramProcessImpl, yang merupakan implementasi dari antarmuka Proses yang dijelaskan sebelumnya. Di sinilah pernyataan warisan “diterjemahkan” ke pernyataan java, mengandalkan kerangka AWS Blu Age:

CocrdslcProcessImpl.java

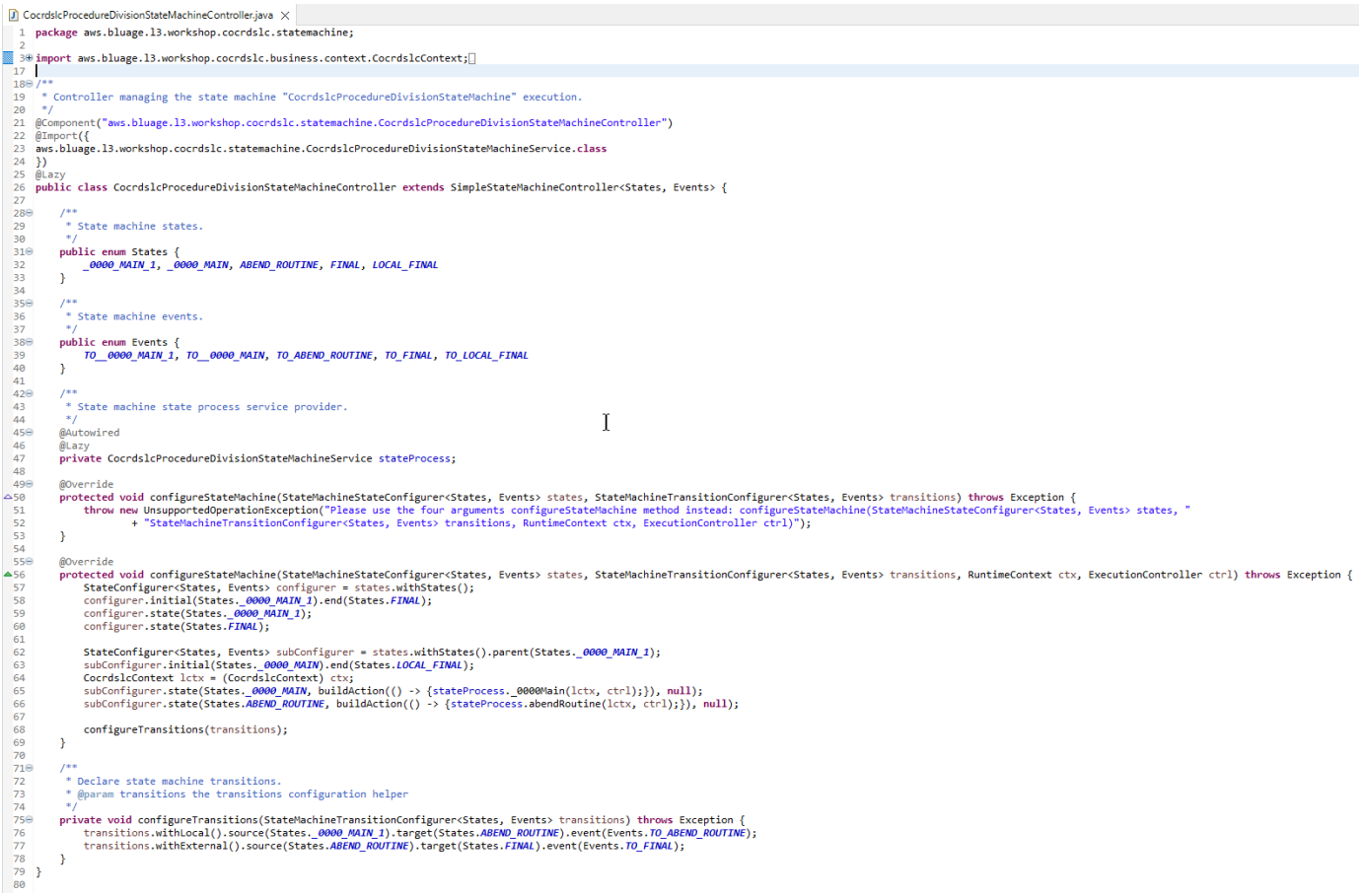
```
210     /**
211      * Process operation sendScreen.
212      *
213      * @param ctx
214      * @param ctrl
215      */
216     @Override
217     public void sendScreen(final CocrdslcContext ctx, final ExecutionController ctrl) {
218         ctx.getCcWorkAreas().setCcardNextMapset(ctx.getWsLiterals().getLitThismapset());
219         ctx.getCcWorkAreas().setCcardNextMap(ctx.getWsLiterals().getLitThismap());
220         ctx.getCarddemoCommarea().setCdemoPgmReenter(true);
221         SendMapBuilder.newInstance(ctx.getDfheiblk(), ctx)
222         .withMap(ctx.getCcWorkAreas().getCcardNextMap())
223         .withMapset(ctx.getCcWorkAreas().getCcardNextMapset())
224         .withData(ctx.getGroup1().getCcrdslaoReference())
225         .withCursor()
226         .withErase()
227         .withFreeKB()
228         .execute();
229         ctx.getWsMiscStorage().setWsRespCd(ctx.getDfheiblk().getEibresp());
230     }
231
232     /**
233      * Process operation processInputs.
234      *
235      * @param ctx
236      * @param ctrl
237      */
238     @Override
239     public void processInputs(final CocrdslcContext ctx, final ExecutionController ctrl) {
240         receiveMap(ctx, ctrl);
241         editMapInputs(ctx, ctrl);
242         ctx.getCcWorkAreas().setCcardErrorMsg(ctx.getWsMiscStorage().getWsReturnMsg());
243         ctx.getCcWorkAreas().setCcardNextProg(ctx.getWsLiterals().getLitThispgm());
244         ctx.getCcWorkAreas().setCcardNextMapset(ctx.getWsLiterals().getLitThismapset());
245         ctx.getCcWorkAreas().setCcardNextMap(ctx.getWsLiterals().getLitThismap());
246     }
247
```

- *base package*`.`*program*`.statemachine`: paket ini mungkin tidak selalu ada. Ini diperlukan ketika modernisasi aliran kontrol lama harus menggunakan pendekatan mesin negara (yaitu menggunakan StateMachine kerangka Spring) untuk menutupi aliran eksekusi lama dengan benar.

  Dalam hal ini, sub-paket statemachine berisi dua kelas:

  - *Program*`ProcedureDivisionStateMachineController`: kelas yang memperluas kelas yang mengimplementasikan antarmuka `StateMachineController` (mendefinisikan operasi yang diperlukan untuk mengontrol eksekusi mesin status) dan `StateMachineRunner`

(mendefinisikan operasi yang diperlukan untuk menjalankan mesin status) antarmuka, yang digunakan untuk menggerakkan mekanisme mesin status Spring; misalnya, `SimpleStateMachineController` seperti dalam kasus sampel.



```
CocrdslcProcedureDivisionStateMachineController.java
package aws.bluage.l3.workshop.cocrdslc.statemachine;

import aws.bluage.l3.workshop.cocrdslc.business.context.CocrdslcContext;

/**
 * Controller managing the state machine "CocrdslcProcedureDivisionStateMachine" execution.
 */
@Component("aws.bluage.l3.workshop.cocrdslc.statemachine.CocrdslcProcedureDivisionStateMachineController")
@Import({
aws.bluage.l3.workshop.cocrdslc.statemachine.CocrdslcProcedureDivisionStateMachineService.class
})
@Lazy
public class CocrdslcProcedureDivisionStateMachineController extends SimpleStateMachineController<States, Events> {

    /**
     * State machine states.
     */
    public enum States {
        _0000_MAIN_1, _0000_MAIN, ABEND_ROUTINE, FINAL, LOCAL_FINAL
    }

    /**
     * State machine events.
     */
    public enum Events {
        TO__0000_MAIN_1, TO__0000_MAIN, TO_ABEND_ROUTINE, TO_FINAL, TO_LOCAL_FINAL
    }

    /**
     * State machine state process service provider.
     */
    @Autowired
    @Lazy
    private CocrdslcProcedureDivisionStateMachineService stateProcess;

    @Override
    protected void configureStateMachine(StateMachineStateConfigurer<States, Events> states, StateMachineTransitionConfigurer<States, Events> transitions) throws Exception {
        throw new UnsupportedOperationException("Please use the four arguments configureStateMachine method instead: configureStateMachine(StateMachineStateConfigurer<States, Events> states, "
                + "StateMachineTransitionConfigurer<States, Events> transitions, RuntimeContext ctx, ExecutionController ctrl)");
    }

    @Override
    protected void configureStateMachine(StateMachineStateConfigurer<States, Events> states, StateMachineTransitionConfigurer<States, Events> transitions, RuntimeContext ctx, ExecutionController ctrl) throws Exception {
        StateConfigurer<States, Events> configurer = states.withStates();
        configurer.initial(States._0000_MAIN_1).end(States.FINAL);
        configurer.state(States._0000_MAIN_1);
        configurer.state(States.FINAL);

        StateConfigurer<States, Events> subConfigurer = states.withStates().parent(States._0000_MAIN_1);
        subConfigurer.initial(States._0000_MAIN).end(States.LOCAL_FINAL);
        CocrdslcContext lctx = (CocrdslcContext) ctx;
        subConfigurer.state(States._0000_MAIN, buildAction(() -> {stateProcess._0000Main(lctx, ctrl);}), null);
        subConfigurer.state(States.ABEND_ROUTINE, buildAction(() -> {stateProcess.abendRoutine(lctx, ctrl);}), null);

        configureTransitions(transitions);
    }

    /**
     * Declare state machine transitions.
     * @param transitions the transitions configuration helper
     */
    private void configureTransitions(StateMachineTransitionConfigurer<States, Events> transitions) throws Exception {
        transitions.withLocal().source(States._0000_MAIN_1).target(States.ABEND_ROUTINE).event(Events.TO_ABEND_ROUTINE);
        transitions.withExternal().source(States.ABEND_ROUTINE).target(States.FINAL).event(Events.TO_FINAL);
    }
}
```

Pengontrol mesin negara mendefinisikan kemungkinan status yang berbeda dan transisi di antara mereka, yang mereproduksi aliran kontrol eksekusi lama untuk program yang diberikan.

Saat membangun mesin status, pengontrol mengacu pada metode yang didefinisikan dalam kelas layanan terkait yang terletak di paket mesin negara dan dijelaskan di bawah ini:

```
subConfigurer.state(States._0000_MAIN, buildAction(() ->
 {stateProcess._0000Main(lctx, ctrl);}), null);
subConfigurer.state(States.ABEND_ROUTINE, buildAction(() ->
 {stateProcess.abendRoutine(lctx, ctrl);}), null);
```

- *Program*ProcedureDivisionStateMachineService: kelas layanan ini mewakili beberapa logika bisnis yang diperlukan untuk terikat dengan mesin status yang dibuat oleh pengontrol mesin negara, seperti yang dijelaskan sebelumnya.

  Kode dalam metode kelas ini menggunakan Peristiwa didefinisikan dalam pengontrol mesin negara:

CocrdslcProcedureDivisionStateMachineService.java

```
59    /**
60     * State process operation _0000Main.
61     *
62     * @param ctx
63     * @param ctrl
64     */
65    void _0000Main(CocrdslcContext ctx, ExecutionController ctrl) {
66        ctx.getDfheiblk().bind(ArgUtils.get(ctx, 0));
67        ctx.getDfhcommarea().bind(ArgUtils.get(ctx, 1));
68
69        /*
70        ****************************************************************
71        Program:     COCRDSLC.CBL                                      *
72        Layer:       Business logic                                    *
73        Function:    Accept and process credit card detail request     *
74        ****************************************************************
75        Copyright Amazon.com, Inc. or its affiliates.
76        All Rights Reserved.
77        Licensed under the Apache License, Version 2.0 (the \"License\").
78        You may not use this file except in compliance with the License.
79        You may obtain a copy of the License at
80        http://www.apache.org/licenses/LICENSE-2.0
81        Unless required by applicable law or agreed to in writing,
82        software distributed under the License is distributed on an
83        \"AS IS\" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND,
84        either express or implied. See the License for the specific
85        language governing permissions and limitations under the License
86        ****************************************************************
87        Ver: CardDemo_v1.0-15-g27d6c6f-68 Date: 2022-07-19 23:16:00 CDT */
88        instanceStateMachineController.registerSignalHandler(Events.TO_ABEND_ROUTINE, "!ABEND");
89        HandleAbendBuilder.newInstance(ctx.getDfheiblk(), ctx).execute().handleException();
90        ctx.getCcWorkAreas().getCcWorkAreaReference().getField().initialize();
91        ctx.getWsMiscStorage().getField().initialize();
92        DataUtils.initialize(ctx.getWsCommarea().getWsCommareaReference());
93
```

CocrdslcProcedureDivisionStateMachineService.java

```
     *
     * @param ctx
     * @param ctrl
     */
    void abendRoutine(CocrdslcContext ctx, ExecutionController ctrl) {
        if (DataUtils.isLowValue(ctx.getAbendData().getAbendMsgReference())) {
            ctx.getAbendData().setAbendMsg("UNEXPECTED ABEND OCCURRED.");
        }
        ctx.getAbendData().setAbendCulprit(ctx.getWsLiterals().getLitThispgm());
        SendTextBuilder.newInstance(ctx.getDfheiblk(), ctx)
        .withData(ctx.getAbendData())
        .withLength(134)
        .execute();
        HandleAbendBuilder.newInstance(ctx.getDfheiblk(), ctx).cancel().execute().handleException();
        AbendBuilder.newInstance(ctx.getDfheiblk(), ctx).withAbendCode("9999").execute().handleException();

        /*
        Ver: CardDemo_v1.0-15-g27d6c6f-68 Date: 2022-07-19 23:12:33 CDT */
        instanceStateMachineController.sendEvent(Events.TO_FINAL);

    }
}
```

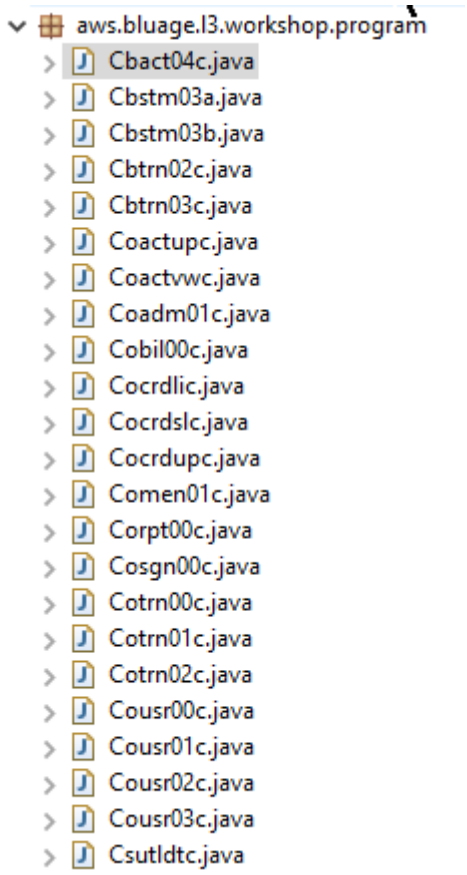
Layanan statemachine juga melakukan panggilan ke implementasi layanan proses yang dijelaskan sebelumnya:

CocrdslcProcedureDivisionStateMachineService.java

```
        /*
        ************************************************************
        COMING FROM CREDIT CARD LIST SCREEN
        SELECTION CRITERIA ALREADY VALIDATED
        ************************************************************ */
    } else if (ctx.getCarddemoCommarea().isCdemoPgmEnter() && DataUtils.compare(ctx.getCarddemoCommarea().getCdemoFromProgramReference(), ctx.getWsLiterals().getLitCclistpgmReference()) == 0) {
        ctx.getWsMiscStorage().setInputOk(true);
        ctx.getCcWorkAreas().setCcAcctIdN(ctx.getCarddemoCommarea().getCdemoAcctId());
        ctx.getCcWorkAreas().setCcCardNumN(ctx.getCarddemoCommarea().getCdemoCardNum());
        instanceCocrdslcProcess.getcardByacctcard(ctx, ctrl);
        instanceCocrdslcProcess.sendMap(ctx, ctrl);
        instanceCocrdslcProcess.commonReturn(ctx, ctrl);
    } else if (ctx.getCarddemoCommarea().isCdemoPgmEnter()) {

        /*
        ************************************************************
        COMING FROM SOME OTHER CONTEXT
        SELECTION CRITERIA TO BE GATHERED
        ************************************************************ */
        instanceCocrdslcProcess.sendMap(ctx, ctrl);
        instanceCocrdslcProcess.commonReturn(ctx, ctrl);
    } else if (ctx.getCarddemoCommarea().isCdemoPgmReenter()) {
        instanceCocrdslcProcess.processInputs(ctx, ctrl);
        if (ctx.getWsMiscStorage().isInputError()) {
            instanceCocrdslcProcess.sendMap(ctx, ctrl);
            instanceCocrdslcProcess.commonReturn(ctx, ctrl);
        } else {
            instanceCocrdslcProcess.getcardByacctcard(ctx, ctrl);
            instanceCocrdslcProcess.sendMap(ctx, ctrl);
            instanceCocrdslcProcess.commonReturn(ctx, ctrl);
        }
    } else {
        ctx.getAbendData().setAbendCulprit(ctx.getWsLiterals().getLitThispgm());
        ctx.getAbendData().setAbendCode("0001");
        DataUtils.setToBlank(ctx.getAbendData().getAbendReasonReference());
        ctx.getWsMiscStorage().setWsReturnMsg("UNEXPECTED DATA SCENARIO");
        instanceCocrdslcProcess.sendPlainText(ctx, ctrl);
    }

    /*
    If we had an error setup error message that slipped through
    Display and return */
    if (ctx.getWsMiscStorage().isInputError()) {
        ctx.getCcWorkAreas().setCcardErrorMsg(ctx.getWsMiscStorage().getWsReturnMsg());
        instanceCocrdslcProcess.sendMap(ctx, ctrl);
        instanceCocrdslcProcess.commonReturn(ctx, ctrl);
    }
    instanceCocrdslcProcess.commonReturn(ctx, ctrl);
```

Selain itu, paket bernama *`base package`*`.program` memainkan peran penting, karena mengumpulkan satu kelas per program, yang akan berfungsi sebagai titik masuk program (rincian

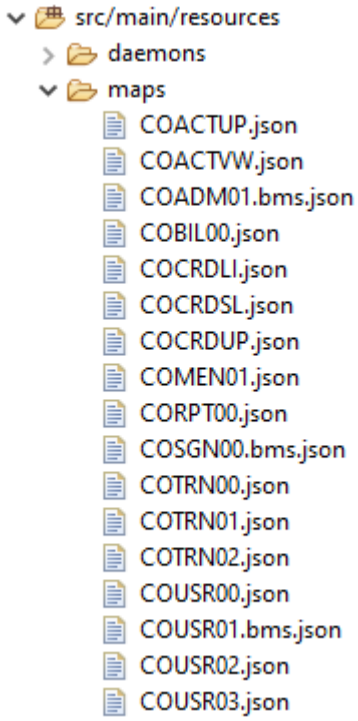
lebih lanjut tentang ini nanti). Setiap kelas mengimplementasikan `Program` antarmuka, penanda untuk titik masuk program.



Artefak lainnya

- Pendamping BMS MAP

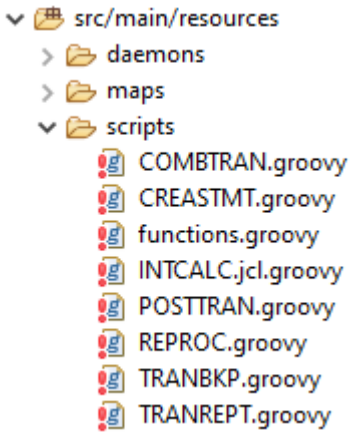
  Selain artefak terkait program, proyek layanan dapat berisi artefak lain untuk berbagai keperluan. Dalam kasus modernisasi aplikasi online CICS, proses modernisasi menghasilkan file json dan dimasukkan ke dalam folder peta folder /src/main/resources:



Runtime Blu Age menggunakan file json tersebut untuk mengikat catatan yang digunakan oleh pernyataan SEND MAP dengan bidang layar.

- Skrip Groovy

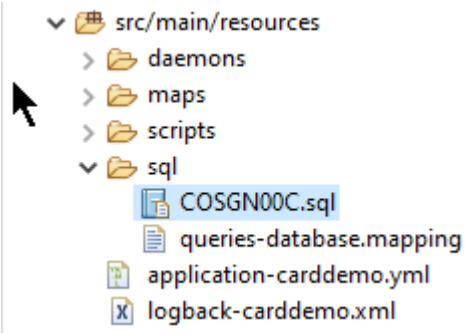
  Jika aplikasi lama memiliki skrip JCL, itu telah dimodernisasi sebagai skrip groovy, disimpan di folder /src/main/resources/scripts (lebih lanjut tentang lokasi spesifik itu nanti):



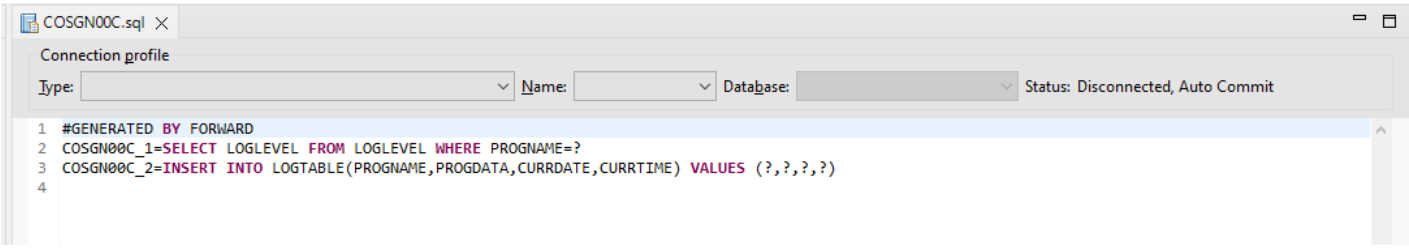
  Skrip tersebut digunakan untuk meluncurkan pekerjaan batch (beban kerja pemrosesan data khusus, non-interaktif, dan intensif CPU).

- File SQL

Jika aplikasi lama menggunakan kueri SQL, kueri SQL modern yang sesuai telah dikumpulkan dalam file properti khusus, dengan program pola penamaan .sql, di mana program adalah nama program yang menggunakan kueri tersebut.



Isi dari file sql tersebut adalah kumpulan entri (key=query), di mana setiap kueri dikaitkan dengan kunci unik, yang digunakan program modern untuk menjalankan kueri yang diberikan:

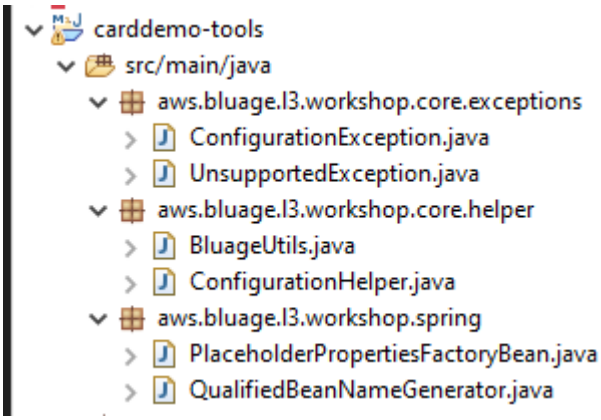


Misalnya, program COSSN00C mengeksekusi query dengan kunci “COSGN00C_1" (entri pertama dalam file sql):

```
    /**
     * Process operation getProgramLogLevel.
     *
     * ****************************************************************
     *                       GET PROGRAM LOG LEVEL
     * ****************************************************************
     *
     * @param ctx
     * @param ctrl
     */
    @Override
    public void getProgramLogLevel(final Cosgn00cContext ctx, final ExecutionController ctrl) {
        SQLExecutorBuilder.newInstance(ctrl, ctx, ctx.getSqlca())
            .mapInParameter(SQLParameterBuilder.newInstance(ctx.getLogData().getLogProgramName()).type(JDBCType.CHAR))
            .mapOutParameter(SQLParameterBuilder.newInstance(ctx.getLogData().getLogProgramLevelReference()))
            .execute("COSGN00C_1");
        if (ctx.getSqlca().getSqlcode() == 100) {
            ctx.getLogData().setLogProgramLevel("N");
        }
    }
```

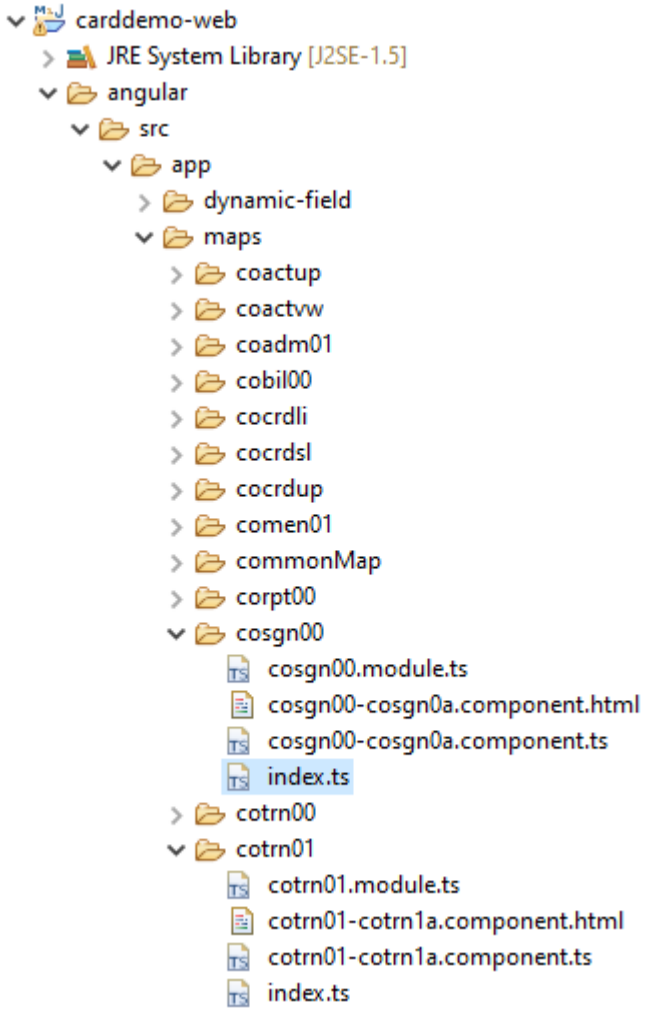
Isi proyek utilitas

Proyek utilitas, yang namanya diakhiri dengan “-tools”, berisi seperangkat utilitas teknis, yang mungkin digunakan oleh semua proyek lainnya.



Konten proyek web

Proyek web hanya hadir saat memodernisasi elemen UI lama. Elemen UI modern yang digunakan untuk membangun front-end aplikasi modern didasarkan pada Angular. Contoh aplikasi yang digunakan untuk menunjukkan artefak modernisasi adalah aplikasi COBOL/CICS, berjalan pada mainframe. Sistem CICS menggunakan MAP untuk mewakili layar UI. Elemen modern yang sesuai adalah, untuk setiap peta, file html disertai dengan file TypeScript:



Proyek web hanya menangani aspek frontend aplikasi Proyek layanan, yang bergantung pada proyek utilitas dan entitas, menyediakan layanan backend. Hubungan antara frontend dan backend dibuat melalui aplikasi web bernama Gapwalk-Application, yang merupakan bagian dari distribusi runtime Blu Age standar. AWS

Menjalankan dan memanggil program

Pada sistem lama, program dikompilasi sebagai executable yang berdiri sendiri yang dapat menyebut diri mereka melalui mekanisme CALL, seperti pernyataan COBOL CALL, melewati argumen bila diperlukan. Aplikasi modern menawarkan kemampuan yang sama tetapi menggunakan pendekatan yang berbeda, karena sifat artefak yang terlibat berbeda dari yang lama.

Di sisi modern, titik masuk program adalah kelas khusus yang mengimplementasikan `Program` antarmuka, adalah komponen Spring (@Component) dan terletak di proyek layanan, dalam paket bernama. *`base package`*`.program`

Registrasi program

Setiap kali server Tomcat yang menghosting aplikasi modern dimulai, aplikasi layanan Springboot juga dimulai, yang memicu pendaftaran program. Registri khusus bernama `ProgramRegistry` diisi dengan entri program, setiap program terdaftar menggunakan pengenalnya, satu entri per pengidentifikasi program yang dikenal, yang berarti bahwa jika suatu program dikenal oleh beberapa pengidentifikasi yang berbeda, registri berisi entri sebanyak yang ada pengidentifikasi.

Pendaftaran untuk program tertentu bergantung pada koleksi pengidentifikasi yang dikembalikan oleh metode getProgramIdentifiers ():

Cbact04c.java

```java
package aws.bluage.l3.workshop.program;

import aws.bluage.l3.workshop.SpringBootLauncher;

/**
 * Reference the spring application of program CBACT04C.
 *  Provides an access to the contained program for the run unit.
 */
@Component
@Import({
    aws.bluage.l3.workshop.cbact04c.business.context.Cbact04cConfiguration.class,
    aws.bluage.l3.workshop.cbact04c.business.context.Cbact04cContext.class,
    aws.bluage.l3.workshop.cbact04c.service.impl.Cbact04cProcessImpl.class
})
public class Cbact04c implements Program {
    /**
     * Unique identifiers for the contained program.
     */
    private static final Set<String> programIdentifiers = Collections.unmodifiableSet(Stream.of("CBACT04C").collect(Collectors.toSet()));

    /**
     * Main program identifier for the contained program.
     */
    private static final String programIdentifier = "CBACT04C";
    @Autowired
    PlatformTransactionManager transactionManager;

    @Autowired
    Map<String, DataSource> datasources;
    @Autowired
    BeanFactory beanFactory;
    /**
     * {@inheritDoc}
     */
    @Override
    public ConfigurableApplicationContext getSpringApplication() {
        return SpringBootLauncher.getCac();
    }

    /**
     * {@inheritDoc}
     */
    @Override
    public void updateExecutionContext(ExecutionContext executionContext) {
        executionContext.setDatasources(datasources);
        executionContext.setDatabaseSupport(ExecutionContext.DatabaseSupport.POSTGRE);
        executionContext.setSqlcaVersion(ExecutionContext.SqlcaVersion.getEnum("ansi-comp5"));
        executionContext.setTransactionManager(transactionManager);
        executionContext.setUseSQLDateNewParadigm(true);
        executionContext.setUseSQLTrimStringType(false);
    }

    /**
     * {@inheritDoc}
     */
    @Override
    public Set<String> getProgramIdentifiers() {
        return programIdentifiers;
    }
```

Dalam contoh ini, program terdaftar sekali, dengan nama 'CBACT04C' (lihat isi koleksi ProgramIdentifiers). Log tomcat menunjukkan setiap pendaftaran program. Pendaftaran program hanya bergantung pada pengidentifikasi program yang dideklarasikan dan bukan nama kelas program itu sendiri (meskipun biasanya pengidentifikasi program dan nama kelas program diselaraskan.

Mekanisme pendaftaran yang sama berlaku untuk program utilitas yang dibawa oleh berbagai aplikasi web utilitas AWS Blu Age, yang merupakan bagian dari distribusi runtime AWS Blu Age. Misalnya, webapp Gapwalk-Utility-Pgm menyediakan ekuivalen fungsional dari utilitas sistem z/OS (IDCAMS, ICEGENER, SORT, dan sebagainya) dan dapat dipanggil dengan program atau skrip modern. Semua program utilitas yang tersedia yang terdaftar di startup Tomcat dicatat di log Tomcat.

Pendaftaran skrip dan daemon

Proses pendaftaran serupa, pada waktu startup Tomcat, terjadi untuk skrip groovy yang terletak di hierarki folder /src/main/resources/scripts. Hirarki folder skrip dilalui, dan semua skrip asyik yang ditemukan (kecuali skrip khusus functions.groovy reserved) terdaftar di`ScriptRegistry`, menggunakan nama pendeknya (bagian dari nama file skrip yang terletak sebelum karakter titik pertama) sebagai kunci untuk pengambilan.

> **Note**
>
> - Jika beberapa skrip memiliki nama file yang akan menghasilkan kunci pendaftaran yang sama, hanya yang terbaru yang terdaftar, menimpa pendaftaran yang sebelumnya ditemui untuk kunci yang diberikan.
> - Mempertimbangkan catatan di atas, perhatikan saat menggunakan sub-folder karena mekanisme pendaftaran meratakan hierarki dan dapat menyebabkan penimpaan yang tidak terduga. Hirarki tidak dihitung dalam proses pendaftaran: biasanya /scripts/a/myscript.groovy dan /scripts/b/myscript.groovy akan menyebabkan /scripts/b/myscript.groovy menimpa /scripts/a/myscripts.groovy.

Skrip groovy di folder /src/main/resources/daemons ditangani sedikit berbeda. Mereka masih terdaftar sebagai skrip biasa, tetapi di samping itu, mereka diluncurkan sekali, langsung pada waktu startup Tomcat, secara asinkron.

Setelah skrip terdaftar di, panggilan REST dapat meluncurkannya`ScriptRegistry`, menggunakan titik akhir khusus yang diekspos oleh Aplikasi Gapwalk. Untuk informasi lebih lanjut, lihat dokumentasi yang sesuai.

Program memanggil program

Setiap program dapat memanggil program lain sebagai subprogram, meneruskan parameter ke sana. Program menggunakan implementasi `ExecutionController` antarmuka untuk melakukannya (sebagian besar waktu, ini akan menjadi `ExecutionControllerImpl` contoh), bersama dengan mekanisme API yang lancar bernama `CallBuilder` untuk membangun argumen panggilan program.

Semua metode program mengambil argumen a `RuntimeContext` dan a `ExecutionController` as method, sehingga selalu `ExecutionController` tersedia untuk memanggil program lain.

Lihat, misalnya, diagram berikut, yang menunjukkan bagaimana program CBST03A memanggil program CBST03B sebagai sub-program, meneruskan parameter ke sana:

Cbstm03aProcessImpl.java

```
67      /**
68       * Process operation xreffileGetNext.
69       *
70       * ----------------------------------------------------------------*
71       *
72       * @param ctx
73       * @param ctrl
74       */
75      @Override
76      public void xreffileGetNext(final Cbstm03aContext ctx, final ExecutionController ctrl) {
77          ctx.getWsM03bArea().setWsM03bDd("XREFFILE");
78          ctx.getWsM03bArea().setM03bRead(true);
79          DataUtils.setToZeroes(ctx.getWsM03bArea().getWsM03bRcReference());
80          DataUtils.setToBlank(ctx.getWsM03bArea().getWsM03bFldtReference());
81          ctrl.callSubProgram("CBSTM03B", CallBuilder.newInstance()
82              .byReference(ctx.getWsM03bArea())
83              .getArguments(), ctx);
84          if (DataUtils.compare(ctx.getWsM03bArea().getWsM03bRcReference(), "00") == 0) {
85
86              /*
87              Do nothing */
88          } else if (DataUtils.compare(ctx.getWsM03bArea().getWsM03bRcReference(), "10") == 0) {
89              ctx.getMiscVariables().setEndOfFile("Y");
90          } else {
91              if (LOGGER.isInfoEnabled()) LOGGER.info("ERROR READING XREFFILE");
92              if (LOGGER.isInfoEnabled()) LOGGER.info("{}{}" , "RETURN CODE: " , ctx.getWsM03bArea().getWsM03bRc());
93              abendProgram(ctx, ctrl);
94          }
95          ctx.getCardXrefRecord().setBytes(ctx.getWsM03bArea().getWsM03bFldtReference().getBytes());
96      }
97
```

- Argumen pertama `ExecutionController.callSubProgram` adalah pengidentifikasi program untuk memanggil (yaitu, salah satu pengidentifikasi yang digunakan untuk pendaftaran program - lihat paragraf di atas).

- Argumen kedua, yang merupakan hasil dari build on the`CallBuilder`, adalah array dari`Record`, sesuai dengan data yang diteruskan dari pemanggil ke callee.
- Argumen ketiga dan terakhir adalah `RuntimeContext` contoh penelepon.

Ketiga argumen adalah wajib dan tidak bisa null, tetapi argumen kedua dapat berupa array kosong.

Callee akan dapat menangani parameter yang dilewatkan hanya jika awalnya dirancang untuk melakukannya. Untuk program COBOL warisan, itu berarti memiliki bagian LINKAGE dan klausa USE untuk pembagian prosedur untuk memanfaatkan elemen LINKAGE.

Misalnya, lihat file sumber COBOL CBSTM03B.CBL yang sesuai:

github.com/aws-samples/aws-mainframe-modernization-carddemo/blob/main/app/cbl/CBSTM03B.CBL

```
 98
 99          LINKAGE SECTION.
100          01  LK-M03B-AREA.
101              05  LK-M03B-DD          PIC X(08).
102              05  LK-M03B-OPER        PIC X(01).
103                88  M03B-OPEN       VALUE 'O'.
104                88  M03B-CLOSE      VALUE 'C'.
105                88  M03B-READ       VALUE 'R'.
106                88  M03B-READ-K     VALUE 'K'.
107                88  M03B-WRITE      VALUE 'W'.
108                88  M03B-REWRITE    VALUE 'Z'.
109              05  LK-M03B-RC          PIC X(02).
110              05  LK-M03B-KEY         PIC X(25).
111              05  LK-M03B-KEY-LN      PIC S9(4).
112              05  LK-M03B-FLDT        PIC X(1000).
113
114          PROCEDURE DIVISION USING LK-M03B-AREA.
115
```

Jadi program CBSTM03B mengambil satu `Record` sebagai parameter (array ukuran 1). Inilah yang `CallBuilder` sedang dibangun, menggunakan rantai metode byReference () dan getArguments ().

Class API `CallBuilder` fasih memiliki beberapa metode yang tersedia untuk mengisi array argumen untuk diteruskan ke callee:

- asPointer (RecordAdaptable): tambahkan argumen jenis pointer, dengan referensi. Pointer mewakili alamat struktur data target.

- byReference (RecordAdaptable): tambahkan argumen dengan referensi. Penelepon akan melihat modifikasi yang dilakukan callee.
- ByReference (RecordAdaptable...): varian varargs dari metode sebelumnya.
- byValue (Object): tambahkan argumen, diubah menjadi`Record`, berdasarkan nilai. Penelepon tidak akan melihat modifikasi yang dilakukan callee.
- byValue (RecordAdaptable): sama seperti metode sebelumnya, tetapi argumen langsung tersedia sebagai`RecordAdaptable`.
- byValueWithBounds (Object, int, int): tambahkan argumen, diubah menjadi a`Record`, mengekstrak bagian array byte yang ditentukan oleh batas yang diberikan, berdasarkan nilai.

Akhirnya, metode getArguments akan mengumpulkan semua argumen yang ditambahkan dan mengembalikannya sebagai array. `Record`

> **Note**
>
> Ini adalah tanggung jawab pemanggil untuk memastikan array argumen memiliki ukuran yang diperlukan, bahwa item dipesan dengan benar dan kompatibel, dalam hal tata letak memori dengan tata letak yang diharapkan untuk elemen tautan.

Skrip memanggil program

Memanggil program terdaftar dari skrip groovy memerlukan penggunaan instance kelas yang mengimplementasikan antarmuka. `MainProgramRunner` Biasanya, mendapatkan instance seperti itu dicapai melalui ApplicationContext penggunaan Spring:

REPROC.groovy

```groovy
// Import
import com.netfective.bluage.gapwalk.rt.provider.ScriptRegistry
import com.netfective.bluage.gapwalk.rt.call.MainProgramRunner
import com.netfective.bluage.gapwalk.io.support.FileConfigurationUtils
import com.netfective.bluage.gapwalk.rt.job.support.DefaultJobContext
import com.netfective.bluage.gapwalk.rt.utils.GroovyUtils
import com.netfective.bluage.gapwalk.rt.io.support.FileConfiguration
import com.netfective.bluage.gapwalk.rt.shared.AbendException
import com.netfective.bluage.gapwalk.rt.call.exception.GroovyExecutionException
// Variables
mpr = applicationContext.getBean("com.netfective.bluage.gapwalk.rt.call.ExecutionController", MainProgramRunner.class)
TreeMap mapTransfo = [:]
```

Setelah `MainProgramRunner` antarmuka tersedia, gunakan metode RunProgram untuk memanggil program dan meneruskan pengenal program target sebagai parameter:

REPROC.groovy

```
//*******************************************************************
//*                              STEPS                              *
//*******************************************************************
// STEP PRC001 - PGM - IDCAMS*****************************************
def stepPRC001(Object shell, Map params, Map programResults){
    shell.with {
        if (checkValidProgramResults(programResults)) {
            return execStep("PRC001", "IDCAMS", programResults, {
                mpr
                    .withFileConfigurations(new FileConfigurationUtils()
                        .systemOut("SYSPRINT")
                        .output("*")
                        .build()
                        .bluesam("FILEIN")
                        .dataset("NULLFILE")
                        .disposition("SHR")
                        .build()
                        .bluesam("FILEOUT")
                        .dataset("NULLFILE")
                        .disposition("SHR")
                        .build()
                        .fileSystem("SYSIN")
                        .path("&CNTLLIB(REPROCT)")
                        .disposition("SHR")
                        .build()
                        .getFileConfigurations(fcmap))
                    .withParameters(params)
                    .runProgram("IDCAMS")
                })
        }
    }
}
```

Dalam contoh sebelumnya, langkah pekerjaan memanggil IDCAMS (program utilitas penanganan file), menyediakan pemetaan antara definisi kumpulan data aktual dan pengidentifikasi logisnya.

Saat berhadapan dengan kumpulan data, program lama sebagian besar menggunakan nama logis untuk mengidentifikasi kumpulan data. Ketika program dipanggil dari skrip, skrip harus memetakan nama logis dengan kumpulan data fisik yang sebenarnya. Kumpulan data ini dapat berada di sistem file, dalam penyimpanan Blusam atau bahkan ditentukan oleh aliran inline, penggabungan beberapa set data, atau pembuatan GDG.

Gunakan withFileConfiguration metode ini untuk membangun peta logis ke fisik kumpulan data dan membuatnya tersedia untuk program yang disebut.

Tulis program Anda sendiri

Menulis program Anda sendiri untuk skrip atau program modern lainnya untuk dipanggil adalah tugas umum. Biasanya, pada proyek modernisasi, Anda menulis program Anda sendiri ketika program warisan yang dapat dieksekusi ditulis dalam bahasa yang tidak didukung oleh proses modernisasi, atau sumber telah hilang (ya, itu bisa terjadi), atau program adalah utilitas yang sumbernya tidak tersedia.

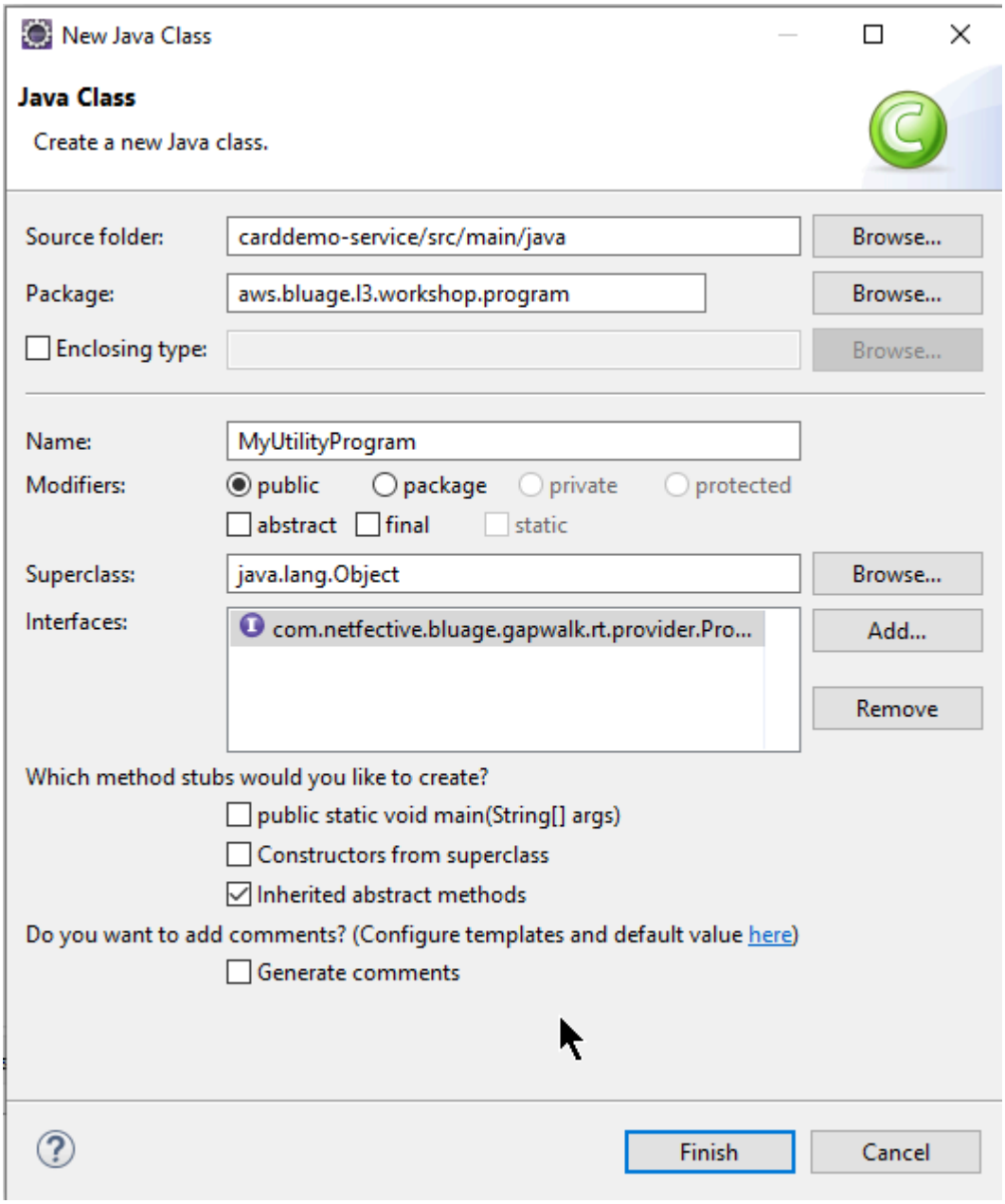
Dalam hal ini, Anda mungkin harus menulis program yang hilang, di java, sendiri (dengan asumsi Anda memiliki pengetahuan yang cukup tentang perilaku yang diharapkan program, tata letak memori argumen program jika ada, dan sebagainya.) Program java Anda harus mematuhi mekanisme program yang dijelaskan dalam dokumen ini, sehingga program dan skrip lain dapat menjalankannya.

Untuk memastikan program ini dapat digunakan, Anda harus menyelesaikan dua langkah wajib:

- Tulis kelas yang mengimplementasikan `Program` antarmuka dengan benar, sehingga dapat didaftarkan dan dipanggil.
- Pastikan program Anda terdaftar dengan benar, sehingga terlihat dari program/skrip lain.

Menulis implementasi program

Gunakan IDE Anda untuk membuat kelas java baru yang mengimplementasikan `Program` antarmuka:



Gambar berikut menunjukkan Eclipse IDE, yang menangani pembuatan semua metode wajib untuk diimplementasikan:

MyUtilityProgram.java

```
1  package aws.bluage.l3.workshop.program;
2
3  import java.util.Set;
10
11 public class MyUtilityProgram implements Program {
12
13     @Override
14     public ConfigurableApplicationContext getSpringApplication() {
15         // TODO Auto-generated method stub
16         return null;
17     }
18
19     @Override
20     public Set<String> getProgramIdentifiers() {
21         // TODO Auto-generated method stub
22         return null;
23     }
24
25     @Override
26     public Context getContext() {
27         // TODO Auto-generated method stub
28         return null;
29     }
30
31     @Override
32     public void run(ExecutionController ctrl) {
33         // TODO Auto-generated method stub
34
35     }
36
37 }
38
```

Integrasi musim semi

Pertama, kelas harus dinyatakan sebagai komponen Spring. Anotasi kelas dengan anotasi`@Component`:

```
import org.springframework.context.ConfigurableApplicationContext;
import org.springframework.stereotype.Component;

import com.netfective.bluage.gapwalk.rt.call.ExecutionController;
import com.netfective.bluage.gapwalk.rt.context.Context;
import com.netfective.bluage.gapwalk.rt.provider.Program;

import aws.bluage.l3.workshop.SpringBootLauncher;

@Component
public class MyUtilityProgram implements Program {
```

Selanjutnya, terapkan metode yang diperlukan dengan benar. Dalam konteks sampel ini, kami menambahkan `MyUtilityProgram` ke paket yang sudah berisi semua program modern.

Penempatan itu memungkinkan program untuk menggunakan aplikasi Springboot yang ada untuk menyediakan yang diperlukan `ConfigurableApplicationContext` untuk implementasi metode: getSpringApplication

```
public class MyUtilityProgram implements Program {

    @Override
    public ConfigurableApplicationContext getSpringApplication() {
        return SpringBootLauncher.getCac();
    }
```

Anda dapat memilih lokasi yang berbeda untuk program Anda sendiri. Misalnya, Anda dapat menemukan program yang diberikan dalam proyek layanan khusus lainnya. Pastikan proyek layanan yang diberikan memiliki aplikasi Springboot sendiri, yang memungkinkan untuk mengambil ApplicationContext (yang seharusnya a). `ConfigurableApplicationContext`

Memberikan identitas pada program

Agar dapat dipanggil oleh program dan skrip lain, program harus diberikan setidaknya satu pengenal, yang tidak boleh bertabrakan dengan program terdaftar lain yang ada di dalam sistem. Pilihan pengenal mungkin didorong oleh kebutuhan untuk mencakup penggantian program lama yang ada; dalam hal ini, Anda harus menggunakan pengenal yang diharapkan, seperti yang terpenuhi dalam kejadian CALL yang ditemukan di seluruh program lama. Sebagian besar pengidentifikasi program memiliki panjang 8 karakter dalam sistem warisan.

Membuat seperangkat pengidentifikasi yang tidak dapat dimodifikasi dalam program adalah salah satu cara untuk melakukan ini. Contoh berikut menunjukkan memilih “MYUTILPG” sebagai pengidentifikasi tunggal:

```
@Component
public class MyUtilityProgram implements Program {

    /**
     * Unique identifiers for the contained program.
     */
    private static final Set<String> programIdentifiers = Collections.unmodifiableSet(Stream.of("MYUTILPG").collect(Collectors.toSet()));


    public ConfigurableApplicationContext getSpringApplication() {

    @Override
    public Set<String> getProgramIdentifiers() {
        return programIdentifiers;
    }
```

Kaitkan program dengan konteks

Program ini membutuhkan `RuntimeContext` contoh pendamping. Untuk program modern, AWS Blu Age secara otomatis menghasilkan konteks pendamping, menggunakan struktur data yang merupakan bagian dari program warisan.

Jika Anda menulis program Anda sendiri, Anda harus menulis konteks pendamping juga.

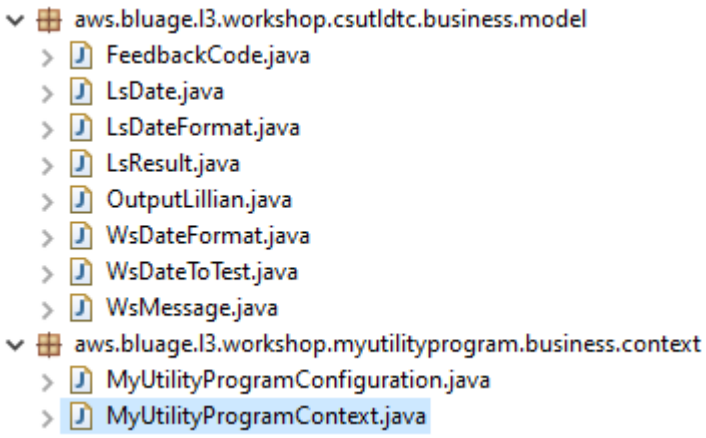
Mengacu padaKelas terkait program, Anda dapat melihat bahwa suatu program membutuhkan setidaknya dua kelas pendamping:

- kelas konfigurasi.
- kelas konteks yang menggunakan konfigurasi.

Jika program utilitas menggunakan struktur data tambahan, itu harus ditulis juga dan digunakan oleh konteksnya.

Kelas-kelas tersebut harus berada dalam paket yang merupakan bagian dari hierarki paket yang akan dipindai saat startup aplikasi, untuk memastikan komponen konteks dan konfigurasi akan ditangani oleh framework Spring.

Mari kita tulis konfigurasi dan konteks minimal, dalam *`base package`*`.myutilityprogram.business.context` paket, yang baru dibuat di proyek entitas:



Berikut adalah konten konfigurasi. Ini menggunakan build konfigurasi yang mirip dengan program -- modern -- lainnya di dekatnya. Anda mungkin harus menyesuaikan ini untuk kebutuhan spesifik Anda.

MyUtilityProgramConfiguration.java

```java
package aws.bluage.l3.workshop.myutilityprogram.business.context;

import java.nio.charset.Charset;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Lazy;

import com.netfective.bluage.gapwalk.datasimplifier.configuration.Configuration;
import com.netfective.bluage.gapwalk.datasimplifier.configuration.ConfigurationBuilder;

/**
 * Creates Datasimplifier configuration for the MyUtilityProgram context.
 */
@org.springframework.context.annotation.Configuration
@Lazy
public class MyUtilityProgramConfiguration {

    @Bean(name = "MyUtilityProgramContextConfiguration")
    public Configuration configuration() {
        return new ConfigurationBuilder()
                .encoding(Charset.forName("CP1047"))
                .humanReadableEncoding(Charset.forName("ISO-8859-15"))
                .initDefaultByte(0)
                .build();
    }
}

```

Catatan:

- Konvensi penamaan umum adalah ProgramNameKonfigurasi.
- Itu harus menggunakan anotasi @org .springFramework.Context.Annotation.Configuration dan @Lazy.
- Nama kacang biasanya mengikuti ProgramNameContextConfiguration konvensi, tetapi ini tidak wajib. Pastikan untuk menghindari tabrakan nama kacang di seluruh proyek.
- Metode tunggal untuk mengimplementasikan harus mengembalikan `Configuration` objek. Gunakan API yang `ConfigurationBuilder` lancar untuk membantu Anda membangunnya.

Dan konteks terkait:

MyUtilityProgramContext.java

```java

import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.context.annotation.Lazy;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;

import com.netfective.bluage.gapwalk.datasimplifier.configuration.Configuration;
import com.netfective.bluage.gapwalk.rt.context.RuntimeContext;

@Component("aws.bluage.l3.workshop.mytuilityprogram.business.context.MyUtilityProgramContext")
@Lazy
@Scope("prototype")
public class MyUtilityProgramContext extends RuntimeContext{

    protected MyUtilityProgramContext(@Qualifier("MyUtilityProgramContextConfiguration") Configuration configuration) {
        super(configuration);
    }

    @Override
    public void cleanUp() {
        // TODO implement clean-up of associated data structures if any
    }

    @Override
    protected void doReset() {
        // TODO implement reset of associated data structures if any
    }

}

```

Catatan

- Kelas konteks harus memperluas implementasi `Context` antarmuka yang ada (salah satu `RuntimeContext` atau`JicsRuntimeContext`, yang ditingkatkan `RuntimeContext` dengan item spesifik JICS).
- Konvensi penamaan umum adalah ProgramNameKonteks.
- Anda harus mendeklarasikannya sebagai komponen Prototipe, dan menggunakan anotasi @Lazy.
- Konstruktor mengacu pada konfigurasi terkait, menggunakan anotasi @Qualifier untuk menargetkan kelas konfigurasi yang tepat.
- Jika program utilitas menggunakan beberapa struktur data tambahan, mereka harus:
  - ditulis dan ditambahkan ke *`base package`*`.business.model` paket.
  - direferensikan dalam konteksnya. Lihatlah kelas konteks lain yang ada untuk melihat bagaimana mereferensikan kelas strcutures data dan mengadaptasi metode konteks (konstruktor/bersih/reset) sesuai kebutuhan.

Sekarang konteks khusus tersedia, biarkan program baru menggunakannya:

MyUtilityProgram.java

```java
import aws.bluage.l3.workshop.SpringBootLauncher;

@Component
@Import({
    aws.bluage.l3.workshop.myutilityprogram.business.context.MyUtilityProgramConfiguration.class,
    aws.bluage.l3.workshop.myutilityprogram.business.context.MyUtilityProgramContext.class
})
public class MyUtilityProgram implements Program {

    @Autowired
    BeanFactory beanFactory;

    /**
     * Unique identifiers for the contained program.
     */
    private static final Set<String> programIdentifiers = Collections.unmodifiableSet(Stream.of("MYUTILPG").collect(Collectors.toSet()));

    private static final String programIdentifier = "MYUTILPG";

    @Override
    public ConfigurableApplicationContext getSpringApplication() {
        return SpringBootLauncher.getCac();
    }

    @Override
    public Set<String> getProgramIdentifiers() {
        return programIdentifiers;
    }

    /**
     * {@inheritDoc}
     */
    @Override
    public String getProgramMainIdentifier() {
        return programIdentifier;
    }


    @Override
    public Context getContext() {
        return ProgramContextStore.getOrCreate(
        getProgramMainIdentifier(),
        aws.bluage.l3.workshop.myutilityprogram.business.context.MyUtilityProgramContext.class,
        beanFactory);
    }

```

Catatan:

- Metode getContext harus diimplementasikan secara ketat seperti yang ditunjukkan, menggunakan delegasi ke getOrCreate metode `ProgramContextStore` kelas dan Spring autowired. `BeanFactory` Sebuah program identifier tunggal digunakan untuk menyimpan konteks program dalam`ProgramContextStore`; identifier ini direferensikan sebagai 'program utama identifier'.
- Konfigurasi pendamping dan kelas konteks harus direferensikan menggunakan anotasi `@Import` pegas.

Menerapkan logika bisnis

Ketika kerangka program selesai, terapkan logika bisnis untuk program utilitas baru.

Lakukan ini dalam `run` metode program. Metode ini akan dijalankan kapan saja program dipanggil, etiher oleh program lain atau oleh skrip.

Selamat mengkodekan!

Menangani pendaftaran program

Akhirnya, pastikan program baru terdaftar dengan benar di`ProgramRegistry`. Jika Anda menambahkan program baru ke paket yang sudah berisi program lain, tidak ada lagi yang harus dilakukan. Program baru diambil dan didaftarkan dengan semua program tetangganya saat startup aplikasi.

Jika Anda memilih lokasi lain untuk program ini, Anda harus memastikan program terdaftar dengan benar di startup Tomcat. Untuk beberapa inspirasi tentang cara melakukannya, lihat metode inisialisasi SpringbootLauncher kelas yang dihasilkan dalam proyek layanan (lihatIsi proyek layanan).

Periksa log startup Tomcat. Setiap pendaftaran program dicatat. Jika program Anda berhasil didaftarkan, Anda akan menemukan entri log yang cocok.

Ketika Anda yakin bahwa program Anda terdaftar dengan benar, Anda dapat mulai mengulangi pengkodean logika bisnis.

Pemetaan nama yang sepenuhnya memenuhi syarat

Bagian ini berisi daftar AWS Blu Age dan pemetaan nama pihak ketiga yang sepenuhnya memenuhi syarat untuk digunakan dalam aplikasi modern Anda.

AWS Pemetaan nama Blu Age sepenuhnya memenuhi syarat

Nama pendek	Nama yang sepenuhnya memenuhi syarat
`CallBuilder`	`com.netfective.bluage.gapwalk.runtime.statements.CallBuilder`
`Configuration`	`com.netfective.bluage.gapwalk.datasimplifier.configuration.Configuration`
`ConfigurationBuilder`	`com.netfective.bluage.gapwalk.datasimplifier.configuration.ConfigurationBuilder`

Nama pendek	Nama yang sepenuhnya memenuhi syarat
ExecutionController	com.netfective.bluage.gapwalk.rt.call.ExecutionController
ExecutionControllerImpl	com.netfective.bluage.gapwalk.rt.call.internal.ExecutionControllerImpl
File	com.netfective.bluage.gapwalk.rt.io.File
MainProgramRunner	com.netfective.bluage.gapwalk.rt.call.MainProgramRunner
Program	com.netfective.bluage.gapwalk.rt.provider.Program
ProgramContextStore	com.netfective.bluage.gapwalk.rt.context.ProgramContextStore
ProgramRegistry	com.netfective.bluage.gapwalk.rt.provider.ProgramRegistry
Record	com.netfective.bluage.gapwalk.datasimplifier.data.Record
RecordEntity	com.netfective.bluage.gapwalk.datasimplifier.entity.RecordEntity
RuntimeContext	com.netfective.bluage.gapwalk.rt.context.RuntimeContext
SimpleStateMachineController	com.netfective.bluage.gapwalk.rt.statemachine.SimpleStateMachineController

Nama pendek	Nama yang sepenuhnya memenuhi syarat
StateMachineController	com.netfective.bluage.gapwalk.rt.statemachine.StateMachineController
StateMachineRunner	com.netfective.bluage.gapwalk.rt.statemachine.StateMachineRunner

Pemetaan nama pihak ketiga yang sepenuhnya memenuhi syarat

Nama pendek	Nama yang sepenuhnya memenuhi syarat
@Autowired	org.springframework.beans.factory.annotation.Autowired
@Bean	org.springframework.context.annotation.Bean
BeanFactory	org.springframework.beans.factory.BeanFactory
@Component	org.springframework.stereotype.Component
ConfigurableApplicationContext	org.springframework.context.ConfigurableApplicationContext
@Import	org.springframework.context.annotation.Import
@Lazy	org.springframework.context.annotation.Lazy

Penyederhanaan Data

Pada sistem mainframe dan midrange (disebut dalam topik berikut sebagai sistem “warisan”), bahasa pemrograman yang sering digunakan seperti COBOL, PL/I atau RPG menyediakan akses tingkat rendah ke memori. Akses ini berfokus pada tata letak memori yang diakses melalui tipe asli seperti dikategorikan, dikemas, atau alfanumerik, mungkin juga digabungkan melalui grup atau array.

Campuran akses ke sepotong memori tertentu, melalui kedua bidang yang diketik dan sebagai akses langsung ke byte (memori mentah), hidup berdampingan dalam program tertentu. Misalnya, program COBOL akan meneruskan argumen ke penelepon sebagai set byte yang berdekatan (LINKAGE), atau membaca/menulis data dari file dengan cara yang sama (catatan), sambil menafsirkan rentang memori tersebut dengan bidang yang diketik yang diatur dalam buku salinan.

Kombinasi seperti akses mentah dan terstruktur ke memori, ketergantungan pada tata letak memori tingkat byte yang tepat, dan jenis warisan, seperti dikategorikan atau dikemas, adalah fitur yang tidak secara asli atau mudah tersedia di lingkungan pemrograman Java.

Sebagai bagian dari solusi AWS Blu Age untuk memodernisasi program warisan ke Java, perpustakaan Data Simplifier menyediakan konstruksi seperti itu ke program Java yang dimodernisasi, dan mengeksposnya dengan cara yang seakrab mungkin bagi pengembang Java (getters/setter, array byte, berbasis kelas). Ini adalah ketergantungan inti dari kode Java modern yang dihasilkan dari program tersebut.

Untuk mempermudah, sebagian besar penjelasan berikut didasarkan pada konstruksi COBOL, tetapi Anda dapat menggunakan API yang sama untuk modernisasi tata letak data PL1 dan RPG, karena sebagian besar konsepnya serupa.

Topik

Kelas utama

Untuk memudahkan pembacaan, dokumen ini menggunakan nama pendek Java dari antarmuka dan AWS kelas Blu Age API. Untuk informasi selengkapnya, lihat FQN dari tipe Java yang dibahas.

Representasi memori tingkat rendah

Pada tingkat terendah, memori (rentang byte yang berdekatan dapat diakses dengan cara yang cepat dan acak) diwakili oleh antarmuka. `Record` Antarmuka ini pada dasarnya adalah abstraksi dari array byte dengan ukuran tetap. Dengan demikian, ia menyediakan setter dan getter yang dapat mengakses atau memodifikasi byte yang mendasarinya.

Representasi data terstruktur

Untuk mewakili data terstruktur, seperti “01 item data”, atau “01 copybook”, seperti yang ditemukan di COBOL DATA DIVISION, `RecordEntity` subclass kelas digunakan. Itu biasanya tidak ditulis dengan tangan, tetapi dihasilkan oleh alat modernisasi Zaman AWS Blu dari konstruksi warisan yang sesuai. Masih berguna untuk mengetahui tentang struktur utama dan API mereka, sehingga Anda dapat memahami bagaimana kode dalam program modern menggunakannya. Dalam kasus COBOL, kode itu adalah Java yang dihasilkan dari DIVISI PROSEDUR mereka.

Kode yang dihasilkan mewakili setiap “01 item data” dengan `RecordEntity` subkelas; setiap bidang dasar atau agregat yang menyusunnya direpresentasikan sebagai bidang Java pribadi, diatur sebagai pohon (setiap item memiliki induk, kecuali yang root).

Untuk tujuan ilustrasi, berikut adalah contoh item data COBOL, diikuti oleh kode yang dihasilkan AWS Blu Age yang sesuai yang memodernkannya:

```
01 TST2.
 02 FILLER PIC X(4).
 02 F1     PIC 9(2) VALUE 42.
 02 FILLER PIC X.
 02        PIC 9(3) VALUE 123.
 02 F2     PIC X VALUE 'A'.
```

```
public class Tst2 extends RecordEntity {

    private final Group root = new Group(getData()).named("TST2");
    private final Filler filler = new Filler(root,new AlphanumericType(4));
    private final Elementary f1 = new Elementary(root,new ZonedType(2, 0, false),new
 BigDecimal("42")).named("F1");
    private final Filler filler1 = new Filler(root,new AlphanumericType(1));
    private final Filler filler2 = new Filler(root,new ZonedType(3, 0, false),new
 BigDecimal("123"));
    private final Elementary f2 = new Elementary(root,new
 AlphanumericType(1),"A").named("F2");
```

```
    /**
     * Instantiate a new Tst2 with a default record.
     * @param configuration the configuration
     */
    public Tst2(Configuration configuration) {
        super(configuration);
        setupRoot(root);
    }
    /**
     * Instantiate a new Tst2 bound to the provided record.
     * @param configuration the configuration
     * @param record the existing record to bind
     */
    public Tst2(Configuration configuration, RecordAdaptable record) {
        super(configuration);
        setupRoot(root, record);
    }

    /**
     * Gets the reference for attribute f1.
     * @return the f1 attribute reference
     */
    public ElementaryRangeReference getF1Reference() {
        return f1.getReference();
    }

    /* *
     * Getter for f1 attribute.
     * @return f1 attribute
     */
    public int getF1() {
        return f1.getValue();
    }

    /**
     * Setter for f1 attribute.
     * @param f1 the new value of f1
     */
    public void setF1(int f1) {
        this.f1.setValue(f1);
    }
    /**
     * Gets the reference for attribute f2.
```

```
     * @return the f2 attribute reference
     */
    public ElementaryRangeReference getF2Reference() {
        return f2.getReference();
    }

    /**
     * Getter for f2 attribute.
     * @return f2 attribute
     */
    public String getF2() {
        return f2.getValue();
    }

    /**
     * Setter for f2 attribute.
     * @param f2 the new value of f2
     */
    public void setF2(String f2) {
        this.f2.setValue(f2);
    }
}
```

Bidang dasar

Bidang kelas `Elementary` (atau`Filler`, ketika tidak disebutkan namanya) mewakili “daun” dari struktur data lama. Mereka terkait dengan rentang byte yang mendasari (“rentang”) yang berdekatan dan umumnya memiliki tipe (mungkin berparameter) yang mengekspresikan cara menafsirkan dan memodifikasi byte tersebut (dengan masing-masing “decoding” dan “encoding” nilai dari/ke array byte).

Semua tipe dasar adalah subkelas dari. `RangeType` Jenis umum adalah:

Jenis COBOL	Jenis Penyederhanaan Data
`PIC X(n)`	`AlphanumericType`
`PIC 9(n)`	`ZonedType`
`PIC 9(n) COMP-3`	`PackedType`
`PIC 9(n) COMP-5`	`BinaryType`

Bidang agregat

Bidang agregat mengatur tata letak memori isinya (agregat lain atau bidang dasar). Mereka tidak memiliki tipe dasar sendiri.

`Group`bidang mewakili bidang yang berdekatan dalam memori. Masing-masing bidang yang terkandung ditata dalam urutan yang sama dalam memori, bidang pertama berada di offset sehubungan `0` dengan posisi bidang grup dalam memori, bidang kedua berada di offset`0 + (size in bytes of first field)`, dll. Mereka digunakan untuk mewakili urutan bidang COBOL di bawah bidang berisi yang sama.

`Union`bidang mewakili kelipatan bidang mengakses memori yang sama. Masing-masing bidang yang terkandung ditata secara offset sehubungan `0` dengan posisi bidang serikat dalam memori. Mereka misalnya digunakan untuk mewakili konstruksi COBOL “REDEFINES” (anak Union pertama adalah item data yang didefinisikan ulang, anak kedua menjadi redefinisi pertamanya, dll.).

Bidang array (subclass dari`Repetition`) mewakili pengulangan, dalam memori, tata letak bidang anak mereka (baik itu agregat itu sendiri atau item dasar). Mereka meletakkan sejumlah tata letak anak seperti itu dalam memori, masing-masing berada di offset`index * (size in bytes of child)`. Mereka digunakan untuk mewakili konstruksi COBOL “TERJADI”.

Primitif

Dalam beberapa kasus modernisasi, “Primitif” juga dapat digunakan untuk menyajikan item data “root” yang independen. Itu sangat mirip digunakan `RecordEntity` tetapi tidak berasal darinya, juga tidak didasarkan pada kode yang dihasilkan. Sebaliknya mereka langsung disediakan oleh runtime AWS Blu Age sebagai subclass dari antarmuka. `Primitive` Contoh kelas yang disediakan tersebut adalah `Alphanumeric` atau`ZonedDecimal`.

Pengikatan dan akses data

Hubungan antara data terstruktur dan data yang mendasari dapat dilakukan dengan berbagai cara.

Antarmuka penting untuk tujuan ini adalah`RecordAdaptable`, yang digunakan untuk mendapatkan `Record` penyediaan “tampilan yang dapat ditulis” pada data yang `RecordAdaptable` mendasarinya. Seperti yang akan kita lihat di bawah, beberapa kelas menerapkan`RecordAdaptable`. Secara timbal balik, AWS Blu Age API dan kode yang memanipulasi memori tingkat rendah (seperti argumen program, file I/O record, CICS commarea, memori yang dialokasikan...) akan sering mengharapkan sebagai pegangan untuk memori itu. `RecordAdaptable`

Dalam kasus modernisasi COBOL, sebagian besar item data dikaitkan dengan memori yang akan diperbaiki selama masa pakai eksekusi program yang sesuai. Untuk tujuan ini, `RecordEntity` subclass dipakai sekali dalam objek induk yang dihasilkan (program Context), dan akan menangani instance yang mendasarinya`Record`, berdasarkan ukuran byte. `RecordEntity`

Dalam kasus COBOL lainnya, seperti mengaitkan elemen LINKAGE dengan argumen program, atau memodernisasi SET ADDRESS OF konstruksi, sebuah `RecordEntity` instance harus dikaitkan dengan yang disediakan. `RecordAdaptable` Untuk tujuan ini, ada dua mekanisme:

- jika `RecordEntity` instance sudah ada, `RecordEntity.bind(RecordAdaptable)` metode (diwarisi dari`Bindable`) dapat digunakan untuk membuat instance ini “menunjuk” ke ini`RecordAdaptable`. Setiap pengambil atau penyetel yang dipanggil `RecordEntity` akan didukung (byte membaca atau menulis) oleh byte yang mendasarinya. `RecordAdaptable`
- jika ingin `RecordEntity` dipakai, konstruktor yang dihasilkan yang menerima a tersedia. `RecordAdaptable`

Sebaliknya, data terstruktur yang `Record` saat ini terikat dapat diakses. Untuk ini, `RecordEntity` mengimplementasikan`RecordAdaptable`, sehingga `getRecord()` dapat dipanggil pada setiap contoh tersebut.

Akhirnya, banyak kata kerja COBOL atau CICS memerlukan akses ke satu bidang, untuk tujuan membaca atau menulis. `RangeReference`Kelas digunakan untuk mewakili akses tersebut. Instance-nya dapat diperoleh dari `getXXXReference()` metode `RecordEntity` yang dihasilkan (`XXX`menjadi bidang yang diakses), dan diteruskan ke metode runtime. `RangeReference`biasanya digunakan untuk mengakses keseluruhan `RecordEntity` atau`Group`, sementara subkelasnya `ElementaryRangeReference` mewakili akses ke `Elementary` bidang.

Perhatikan bahwa sebagian besar pengamatan di atas berlaku untuk `Primitive` subkelas, karena mereka berusaha menerapkan perilaku serupa seperti `RecordEntity` saat disediakan oleh runtime AWS Blu Age (bukan kode yang dihasilkan). Untuk tujuan ini, semua subclass `Primitive` implementasi`RecordAdaptable`, `ElementaryRangeReference` dan `Bindable` antarmuka sehingga dapat digunakan di tempat kedua `RecordEntity` subclass dan bidang dasar.

FQN dari tipe Java yang dibahas

Tabel berikut menunjukkan nama-nama yang sepenuhnya memenuhi syarat dari jenis Java dibahas dalam bagian ini.

Nama pendek	Nama Sepenuhnya Memenuhi Syarat
Alphanumeric	com.netfective.bluage.gapwalk.datasimplifier.elementary.Alphanumeric
AlphanumericType	com.netfective.bluage.gapwalk.datasimplifier.metadata.type.AlphanumericType
BinaryType	com.netfective.bluage.gapwalk.datasimplifier.metadata.type.BinaryType
Bindable	com.netfective.bluage.gapwalk.datasimplifier.data.Bindable
Elementary	com.netfective.bluage.gapwalk.datasimplifier.data.structure.Elementary
ElementaryRangeReference	com.netfective.bluage.gapwalk.datasimplifier.entity.ElementaryRangeReference
Filler	com.netfective.bluage.gapwalk.datasimplifier.data.structure.Filler
Group	com.netfective.bluage.gapwalk.datasimplifier.data.structure.Group
PackedType	com.netfective.bluage.gapwalk.datasimplifier.metadata.type.PackedType

Nama pendek	Nama Sepenuhnya Memenuhi Syarat
Primitive	com.netfective.bluage.gapwalk.datasimplifier.elementary.Primitive
RangeReference	com.netfective.bluage.gapwalk.datasimplifier.entity.RangeReference
RangeType	com.netfective.bluage.gapwalk.datasimplifier.metadata.type.RangeType
Record	com.netfective.bluage.gapwalk.datasimplifier.data.Record
RecordAdaptable	com.netfective.bluage.gapwalk.datasimplifier.data.RecordAdaptable
RecordEntity	com.netfective.bluage.gapwalk.datasimplifier.entity.RecordEntity
Repetition	com.netfective.bluage.gapwalk.datasimplifier.data.structure.Repetition
Union	com.netfective.bluage.gapwalk.datasimplifier.data.structure.Union
ZonedDecimal	com.netfective.bluage.gapwalk.datasimplifier.elementary.ZonedDecimal

Nama pendek	Nama Sepenuhnya Memenuhi Syarat
`ZonedType`	`com.netfective.bluage.gapwalk.datasimplifier.metadata.type.ZonedType`

AWS Konfigurasi Blu Age Runtime dan file konfigurasi

Framework Velocity dan kode klien adalah aplikasi web yang menggunakan framework Spring Boot. Ini memanfaatkan kemampuan Spring untuk memasok konfigurasi, dengan beberapa kemungkinan lokasi dan aturan prioritas. Ada juga aturan prioritas serupa untuk memasok banyak file lain, seperti skrip groovy, sql, dll.

Framework Velocity juga berisi aplikasi web opsional tambahan, yang dapat dipilih jika diperlukan.

Topik

- Dasar-dasar konfigurasi aplikasi
- Prioritas aplikasi
- JNDI untuk database
- Menggunakan AWS rahasia
- File lain (groovy, sql, dll.)
- Aplikasi web tambahan
- Mengaktifkan properti
- Konfigurasikan otentikasi untuk aplikasi Gapwalk

Dasar-dasar konfigurasi aplikasi

Cara default untuk menangani konfigurasi aplikasi adalah melalui penggunaan file YAMM khusus yang akan disediakan di `config` folder server aplikasi. Ada dua file konfigurasi YAMB utama:

- `application-main.yaml`
- `application-`*`profile`*`.yaml`(di mana *`profile`* nilai diatur selama pembuatan aplikasi).

File pertama mengkonfigurasi kerangka kerja, yaitu`Gapwalk-application.war`, sedangkan yang kedua adalah untuk opsi tambahan khusus untuk aplikasi klien. Ini berfungsi dengan penggunaan profil pegas: aplikasi Gapwalk menggunakan `main` profil, sedangkan aplikasi klien menggunakan profil. *`profile`*

Contoh berikut menunjukkan file YAMM utama yang khas.

```
########################################
##### JICS datasource configuration #####
########################################
datasource:
  jicsDs:
    driver-class-name : org.postgresql.Driver
    url: jdbc:postgresql://localhost/jics
    username: jics
    password: jics
    type : org.postgresql.ds.PGSimpleDataSource

#########################################################
##### Embedded Bluesam datasource configuration #####
#########################################################
  bluesamDs :
    driver-class-name : org.postgresql.Driver
    url : jdbc:postgresql://localhost/bluesam
    username : bluesam
    password : bluesam
    type : org.postgresql.ds.PGSimpleDataSource

########################################
##### Embedded Bluesam configuration #####
########################################
bluesam :
  remote : false
  cache : ehcache
  persistence : pgsql #pgsql, mssql, xodus...
  ehcache:
    resource-pool:
      size: 4GB
  write-behind:
```

Contoh berikut menunjukkan file YAMM klien tipikal.

```
# Logback context logger integration.
logging.config : classpath:logback-[redacted].xml
# Limits Spring logger output.
logging.level.org.springframework.beans.factory.support.DefaultListableBeanFactory : WARN
logging.level.org.springframework.statemachine : WARN
# If the datasource support mode is not static-xa, spring JTA transactions autoconfiguration must me disabled
spring.jta.enabled : false

spring:
  aws:
    client:
      datasources:
        names: primary
        primary:
          secret: arn:aws:secretsmanager:[redacted]

spring.jta.atomikos.datasource.primary.unique-resource-name: primary
spring.jta.atomikos.datasource.primary.xa-data-source-class-name: org.postgresql.xa.PGXADataSource
spring.jta.atomikos.datasource.primary.maxPoolSize: 20
spring.jta.atomikos.datasource.primary.autoCommit: false
```

Untuk informasi tentang konten file YAMB, lihatMengaktifkan properti.

Prioritas aplikasi

Untuk file konfigurasi ini, aturan prioritas musim semi berlaku. Khususnya:

- File `application-main` YAMB muncul di file perang utama Gapwalk dengan nilai default, dan yang ada di `config` folder menggantikannya.
- Hal yang sama harus dilakukan untuk konfigurasi aplikasi klien
- Parameter tambahan dapat diteruskan pada baris perintah pada waktu peluncuran server. Mereka akan mengesampingkan yang YAMM.

Untuk informasi selengkapnya, lihat dokumentasi Official Spring Boot.

JNDI untuk database

Konfigurasi database mungkin disertakan dengan JNDI di file context.xml di Tomcat. Konfigurasi seperti itu akan mengesampingkan konfigurasi YAMAL. Tetapi perhatikan bahwa menggunakan ini tidak akan memungkinkan untuk membungkus kredensil Anda di manajer rahasia (lihat di bawah).

Contoh berikut menunjukkan konfigurasi sampel untuk JICS dan BluSam database.

```
<Resource auth="Container" driverClassName="org.postgresql.Driver" initialSize="0"
 maxIdle="5"
   maxOpenPreparedStatements="-1" maxTotal="10" maxWaitMillis="-1" name="jdbc/jics"
```

```
    poolPreparedStatements="true" testOnBorrow="false" type="javax.sql.DataSource"
    url="jdbc:postgresql://XXXX.rds.amazonaws.com:5432/XXXX" username="XXXX"
  password="XXXX" />
```

jdbc/jics

Akan `jdbc/jics` untuk database JICS dan `jdbc/bluesam` (perhatikan 'e') untuk database blusam.

url="jdbc:postgresql: //xxxx.rds.amazonaws.com:5432/xxxx” Username="xxxx” kata sandi="xxxx”

Url database, nama pengguna dan kata sandi.

Menggunakan AWS rahasia

Beberapa konfigurasi sumber daya yang berisi kredensyal dapat diamankan lebih lanjut dengan menggunakan rahasia. AWS Idenya adalah untuk menyimpan data penting AWS secara rahasia dan memiliki referensi ke rahasia dalam konfigurasi YAMB sehingga konten rahasia dipilih dengan cepat di startup tomcat.

Rahasia untuk Aurora

Konfigurasi database Aurora (untuk jics, blusam, db pelanggan, dll) akan menggunakan rahasia database bawaan, yang akan mengisi semua bidang yang relevan secara otomatis dari database yang sesuai.

> **Note**
>
> `dbname`Kuncinya adalah opsional, tergantung pada konfigurasi database Anda, itu akan masuk ke rahasia atau tidak. Anda dapat menambahkannya di sana secara manual, atau dengan memasok nama ke file YAMB.

Rahasia lainnya

Rahasia lainnya adalah untuk sumber daya yang memiliki satu kata sandi (terutama cache redis yang dilindungi kata sandi). Dalam hal ini jenis rahasia lainnya harus digunakan, dengan satu `password` kunci.

Referensi YAMB ke rahasia

`application-main.yaml`Dapat mereferensikan rahasia ARN untuk berbagai sumber daya. Yang paling penting adalah:

- Kredensi database JICS dengan `spring.aws.jics.db.secret`
- JICS TS Mengantri kredensi Redis dengan `spring.aws.client.jics.queues.ts.redis.secret`
- Kredensi basis data Blusam dengan `spring.aws.client.bluesam.db.secret`
- Kata sandi cache blusam dengan `spring.aws.client.bluesam.redis.secret`
- Blusam mengunci kata sandi cache dengan `spring.aws.client.bluesam.locks.redis.secret`

Contoh berikut menunjukkan bagaimana untuk mendeklarasikan rahasia ini dalam file YAMB.

```
spring:
  aws:
    client:
      bluesam:
        locks:
          redis:
            secret: arn:aws:secretsmanager:XXXX
        db:
          dbname: bluesam
          secret: arn:aws:secretsmanager:XXXX
        redis:
          secret: arn:aws:secretsmanager:XXXX
      jics:
         queues:
          ts:
           redis:
            secret: arn:aws:secretsmanager:XXXX
    jics:
      db:
        secret: arn:aws:secretsmanager:XXXX
```

dbname: bluesam

Dalam contoh ini, nama database tidak dalam rahasia dan disediakan di sini sebagai gantinya.

Klien `application-profile.yaml` dapat mereferensikan ARN rahasia untuk database klien. Ini memerlukan properti tambahan untuk membuat daftar sumber data, yang ditampilkan dalam contoh di bawah ini:

```
spring:
  aws:
    client:
      datasources:
        names: primary,host
        primary:
          secret: arn:aws:secretsmanager:XXXX
        host:

          secret: arn:aws:secretsmanager:XXXX
```

nama: primer, host

Contoh dengan dua sumber data klien bernama primer dan host, masing-masing dengan database dan kredensialnya.

dbname: mydb

Dalam contoh ini, nama database “host” tidak dalam rahasia dan disediakan di sini sebagai gantinya, sedangkan untuk database “primer” itu dalam rahasia.

Tidak ada kunci rahasia yang didukung XA

- mesin (postgres/oracle/db2/mssql)
- port
- dbname
- Skema saat ini
- nama pengguna
- password
- url

`postgres`Hanya untuk kunci `sslMode` rahasia bernilai (`disable/allow/prefer/require/verify-ca/verify-full`) selain properti `spring.aws.rds.ssl.cert-path` yl memungkinkan untuk terhubung dengan SSL.

Kunci rahasia yang didukung XA

Jika database klien menggunakan XA, sub xa-properti didukung melalui nilai-nilai rahasia.

- host
- port
- dbname
- Skema saat ini
- nama pengguna
- password
- url
- SSLConnection (benar/salah)

Namun untuk xa-properti lainnya (misalnya `maxPoolSize` atau`driverType`), kunci YAMAL reguler `spring.jta.atomikos.datasource.XXXX.unique-resource-name` harus tetap disediakan.

Nilai rahasia mengesampingkan properti YAMB.

File lain (groovy, sql, dll.)

File lain yang digunakan oleh proyek pelanggan menggunakan aturan prioritas yang sama seperti yang untuk konfigurasi pegas. Contoh:

- Skrip Groovy adalah `.groovy` file di folder atau `scripts` subfolder.
- Skrip SQL adalah `.sql` file dalam `sql` folder atau subfolder.
- Skrip daemon adalah `.groovy` file di folder atau `daemons` subfolder.
- Query File pemetaan database adalah file bernama `queries-database.mapping` file dalam `sql` subfolder folder.
- Template Jasper adalah `.jrxml` file di `templates` folder atau subfolder.
- Katalog dataset adalah `.json` file dalam folder. `catalog`
- File LNK adalah `.json` file di folder. `lnk`

Semua lokasi ini dapat diganti melalui properti sistem atau properti ymlklien.

- Untuk skrip Groovy: `configuration.scripts`

- Untuk skrip SQL: `configuration.sql`
- Untuk skrip Daemon: `configuration.daemons`
- Untuk file pemetaan Database Query: `configuration.databaseMapping`
- Untuk template Jasper: `configuration.templates`
- Untuk katalog Dataset: `configuration.catalog`
- Untuk file Lnk: `configuration.lnk`

Jika properti tidak ditemukan, file akan diambil dari lokasi default yang disebutkan di atas. Pencarian pertama akan dilakukan dengan direktori kerja tomcat sebagai root, dan terakhir dalam file perang aplikasi.

Aplikasi web tambahan

Framework Velocity berisi aplikasi web tambahan di `webapps-extra` foldernya. Aplikasi ini tidak dilayani secara default oleh server tomcat.

Memilih masuk ke aplikasi web ini bergantung pada proyek modernisasi dan dilakukan dengan memindahkan file perang yang diinginkan dari `webapps-extra` folder ke folder. `webapps` Setelah itu, perang akan dilayani oleh server tomcat pada startup berikutnya.

Beberapa konfigurasi tambahan khusus proyek juga dapat ditambahkan dalam file konfigurasi YAMB untuk setiap perang tambahan, seperti yang dilakukan dalam `application-main.yml` file dan dijelaskan di atas. Perang tambahan adalah:

- `gapwalk-utility-pgm.war`: berisi dukungan untuk program utilitas ZOS dan digunakan `application-utility-pgm.yaml` sebagai konfigurasinya.
- `gapwalk-cl-command.war`: berisi dukungan untuk program utilitas AS/400 dan digunakan `application-cl-command.yaml` sebagai konfigurasinya.
- `gapwalk-hierarchical-support.war`: berisi dukungan transaksi IMS/MFS dan digunakan `application-jhdb.yaml` sebagai konfigurasinya

Mengaktifkan properti

Dalam Spring Boot aplikasi `application-main.yml` adalah file konfigurasi di mana kita mendefinisikan berbagai jenis properti seperti port mendengarkan, konektivitas database, dan banyak lagi.

Topik

Notasi YML

Dalam dokumentasi berikut, properti seperti `parent.child1.child2=true` ditulis sebagai berikut dalam format YAMAL.

```
parent:
 child1:
   child2: true
```

Mulai cepat/Gunakan kasus

Kasus penggunaan berikut menunjukkan contoh kunci dan nilai yang berlaku.

- Aplikasi-main.yml-default

```
----
#### DEFAULT APPLICATION-MAIN.YML FILE      #####
#### SHOWING USEFUL CONFIGURATION ELEMENTS #####
#### SHOULD BE OVERRIDDEN AND EXTERNALIZED   #####

###################################
##### Logging configuration #####
###################################
logging:
  config: classpath:logback-main.xml
  level.org.springframework.beans.factory.support.DefaultListableBeanFactory : WARN

###################################
##### Spring configuration #####
###################################
spring:
  quartz:
    auto-startup: false
    scheduler-name: Default
```

```
    properties:
      org.quartz.threadPool.threadCount: 1
  jta:
    enabled: false
    atomikos.properties.maxTimeout : 600000
    atomikos.properties.default-jta-timeout : 100000
  jpa:
# DISABLE OpenEntityManagerInViewInterceptor
    open-in-view: false
    # Fix Postgres JPA Error:
    # Method org.postgresql.jdbc.PgConnection.createClob() is not yet implemented.
    properties.hibernate.temp.use_jdbc_metadata_defaults : false
#####################################
##### Jics tables configuration #####
#####################################

    # The dialect should match the jics datasource choice
    database-platform : org.hibernate.dialect.PostgreSQLDialect #
 org.hibernate.dialect.PostgreSQLDialect, org.hibernate.dialect.SQLServerDialect

    # those properties can be used to create and initialize jics tables
 automatically.
#    properties:
#      hibernate:
#      globally_quoted_identifiers: true
#        hbm2ddl:
#           import_files_sql_extractor :
 org.hibernate.tool.hbm2ddl.MultipleLinesSqlCommandExtractor
#           import_files : file:./setup/initJics.sql
#           auto : create

#########################
###### Level 2 cache #####
#########################
#        cache:
#          use_second_level_cache: true
#          use_query_cache: true
#          region:
#            factory_class: org.hibernate.cache.ehcache.EhCacheRegionFactory
#      javax:
#        persistence:
#          sharedCache:
#            mode: ENABLE_SELECTIVE
#########################
```

```
###### Redis settings #####
#########################
  session:
    store-type: none #redis

########################################
##### JICS datasource configuration #####
########################################
datasource:
  jicsDs:
    driver-class-name : org.postgresql.Driver # org.postgresql.Driver,
 com.microsoft.sqlserver.jdbc.SQLServerDriver
    url: jdbc:postgresql://localhost/jics # jdbc:postgresql://localhost:5433/jics,
 jdbc:sqlserver://localhost\SQLEXPRESS:1434;databasename=jics;
    username: jics
    password: jics
    type : org.postgresql.ds.PGSimpleDataSource #
 org.postgresql.ds.PGSimpleDataSource,
 com.microsoft.sqlserver.jdbc.SQLServerDataSource

#####################################################
##### Embedded Bluesam datasource configuration #####
#####################################################
  bluesamDs :
    driver-class-name : org.postgresql.Driver # org.postgresql.Driver,
 com.microsoft.sqlserver.jdbc.SQLServerDriver
    url : jdbc:postgresql://localhost/bluesam # jdbc:postgresql://localhost:5433/
jics, jdbc:sqlserver://localhost\SQLEXPRESS:1434;databasename=jics;
    username : bluesam
    password : bluesam
    type : org.postgresql.ds.PGSimpleDataSource #
 org.postgresql.ds.PGSimpleDataSource,
 com.microsoft.sqlserver.jdbc.SQLServerDataSource

########################################
##### Embedded Bluesam configuration #####
########################################
bluesam :
  remote : false
  cache : ehcache
  persistence : pgsql #pgsql, mssql, xodus...
  ehcache:
    resource-pool:
      size: 4GB
```

```
  write-behind:
    enabled: true
  pgsql :
    dataSource : bluesamDs

#########################
##### Jics settings #####
#########################
rabbitmq.host: localhost
jics:
  cache: false #redis
  resource-definitions.store-type: jpa # default value: jpa, other possible value:
 redis
  redis.hostname: 127.0.0.1 # Redis server host.
  redis.password: redis # Login password of the redis server.
  redis.port: 6379 # Redis server port.
  redis.username: # Redis username
  redis.mode: standalone # Redis mode. Possible values: standalone, cluster
jics.disableSyncpoint : false
#jics.initList:
#jics.parameters.datform: DDMMYY
#jics.parameters.applid: VELOCITY
#jics.parameters.sysid: CICS
#jics.parameters.eibtrmid: TERM
#jics.parameters.userid: MYUSERID
#jics.parameters.username: MYUSERNAME
#jics.parameters.opid: XXX
#jics.parameters.cwa.length: 0
#jics.parameters.netname: MYNETNAME
#jics.parameters.jobname: MJOBNAME
#jics.parameters.sysname: SYSNAME

###############################################
##### Jics RunUnitLauncher pool settings #####
###############################################
#jics.runUnitLauncherPool.enable: false
#jics.runUnitLauncherPool.size: 20
#jics.runUnitLauncherPool.validationInterval: 1000

#########################
##### Jhdb settings #####
#########################
#jhdb.lterm: LTERMVAL
#jhdb.identificationCardData: SomeIDData
```

```
###################################
##### DateHelper configuration ####
###################################
#forcedDate: "2013-08-26T12:59:58+01:57"

#############################
##### Sort configuration ####
#############################
#externalSort.threshold: 256MB

###################################
##### Server timeout (10 min)  ####
###################################
spring.mvc.async.request-timeout: 600000

#############################
##### DATABASE STATISTICS  ####
#############################
databaseStatistics : false

######################
##### CALLS GRAPH ####
######################
callGraph : false

#############################
##### SQL SHIFT CODE POINT ####
#############################
# Code point 384 match unicode character \u0180
sqlCodePointShift : 384

#############################
##### LOCK TIMEOUT RECORD  ####
#############################
# Blu4IV record lock timeout
lockTimeout : 100

#############################
##### REPORTS OUTPUT PATH ####
#############################
reportOutputPath: reports

#############################
```

```
##### TASK EXECUTOR       ####
##############################
taskExecutor:
  corePoolSize: 5
  maxPoolSize: 10
  queueCapacity: 50
  allowCoreThreadTimeOut: false

##############################
##### PROGRAM NOT FOUND   ####
##############################
stopExecutionWhenProgNotFound: false

#########################################################
#####  DISP DEFAULT VALUE (to be removed one day) ####
#########################################################
defaultKeepExistingFiles: true

#####################################
#####  JOBQUEUE CONFIGURATION    ####
#####################################
jobqueue:
  api.enabled: false
  impl: none # possible values: quartz, none
  schedulers:  # list of schedulers
    -
      name: queue1
      threadCount: 5
    -
      name: queue2
      threadCount: 5

#############################################################################
##### QUERY BUILDING                                                     ##
# useConcatCondition : false by default
# if true, in the query, the where condition is build with key concatenation  ##
#############################################################################
# query.useConcatCondition: true
----
```

- Gunakan file panjang variabel dengan perintah LISTCAT

```
[**/*.*]
encoding=IBM930
reencoding=false

[global]
listcat.variablelengthpreprocessor.enabled=true
listcat.variablelengthpreprocessor.type=rdw
# use "rdw" if your .listcat file contains a set of records (RDW)
# use "bdw" if your .listcat file contains a set of blocks (bdw)
```

- Berikan Nilai Indikator Byte Null dalam utilitas LOAD/UNLOAD

```
# Unload properties
# For date/time: if use database configuration is enabled, formats are ignored
# For nbi; use hexadecimal syntax to specify the byte value
# - When the value is null in database : the value dumped to the file is filled by
 low value characters and the NBI is
# equal to the byte 6F (the ? character)
# - When the value is not null in database and the column is nullable: the NBI is
 equal to the byte 00 (low value) and NOT
# equal to the byte 40 (space)
unload:
  sqlCodePointShift: 0
  nbi:
    whenNull: "6F"
    whenNotNull: "00"
  useDatabaseConfiguration: false
  format:
    date: MM/dd/yyyy
    time: HH.mm.ss
    timestamp: yyyy-MM-dd-HH.mm.ss.SSSSSS
```

Properti yang tersedia untuk aplikasi utama

Tabel ini memberikan tampilan lengkap parameter kunci/nilai.

Kunci	Tipe	Nilai default	Deskripsi
`logging.config`	Jalur	classpath:logback-main.xml	Kunci standar untuk referensi ke file konfigurasi logback. Kunci logging standar lainnya juga tersedia.
`spring.jta.enabled`	boolean	false	Kunci standar. Jika mode dukungan sumber data tidak statis-xa, konfigurasi otomatis transaksi JTA pegas harus dinonaktifkan.
`datasource.jicsDs` + `-driver-class-name` + `-url` + `-username` + `-password` + `-type`	Sumber data pegas standar dengan subkunci		Berisi informasi koneksi untuk database Jics. Sebagai alternatif, penggunaan rahasia AWS sangat dianjurkan, seperti yang dijelaskan dalam xref:.. /configuration/configuration.adoc [Konfigurasi].
`datasource.bluesamDs` + `-driver-class-name` + `-url` + `-username` + `-password` + `-type`	Sumber data pegas standar dengan subkunci		Berisi informasi koneksi untuk database Blusam. Sebagai alternatif, penggunaan rahasia AWS sangat dianjurkan, seperti yang dijelaskan dalam xref:.. /configuration/

Kunci	Tipe	Nilai default	Deskripsi
			configuration.adoc [Konfigurasi].
`bluesam.disabled`	boolean	false	Apakah akan menonaktifkan blusam sepenuhnya.
`bluesam.cache`	string		Jika tidak diatur, cache blusam tidak akan digunakan. Nilai yang mungkin (implementasi cache) adalah ehcache dan redis.
`forcedDate`	string		Memaksa tanggal ke tanggal yang diberikan jika ada.
`frozenDate`	boolean	true	Menentukan apakah untuk membekukan tanggal. Berlaku hanya `forcedDate` jika juga diatur.
`externalSort.threshold`	datasize (mis. 12MB)		Ambang batas pengurutan: kapan harus beralih ke pengurutan eksternal (gabungan).
`jics.parameters.datform`	string	MMDDYY	Formulir tanggal.

Kunci	Tipe	Nilai default	Deskripsi
`jics.initList``	string		Daftar inisialisasi jics, dipisahkan dengan koma. Jika ada, ini mendefinisikan nama daftar yang dipisahkan koma untuk diaktifkan pada startup tomcat di antara daftar CICS. Nilai contoh:`$UUU,DFH$IVPL,PEZ1` . Ini akan mengalir ke grup yang terdapat dalam daftar tersebut dan definisi sumber daya yang mendasarinya, yang kemudian akan terlihat oleh runtime. Kosong secara default.
`jics.parameters.applid`	string	KECEPATAN	Aplikasi untuk mengidentifikasi aplikasi di JICS (setidaknya 4 karakter, tidak ada panjang maksimal).
`jics.parameters.sysid`	string	CICS	Identifikasi sistem (SYSID).
`jics.parameters.eibtrmid`	string	KETENTUAN	Pengidentifikasi terminal (maksimum 4 karakter, minimal 1).

Kunci	Tipe	Nilai default	Deskripsi
`jics.parameters.userid`	string		Id pengguna (maksimum 8 karakter, tidak ada minimum). Ketika tidak ada nilai yang diberikan (kosong secara default) id sesi HTTP digunakan sebagai id pengguna.
`jics.parameters.username`	string	NAMA PENGGUNA SAYA	Nama pengguna (maksimum 10 karakter, minimal 1).
`jics.parameters.netname`	string	MYNETNAME	Nama jaringan (maksimal 8 karakter, 1 mimmum).
`jics.parameters.opid`	string	XXX	Identifikasi operator 3 karakter.
`jics.parameters.jobname`	string	MJOBNAME	Nama kerja.
`jics.parameters.sysname`	string	SYSNAME	Nama sistem AS400 (sysname).
`jics.parameters.cwa.length`	nomor	0	Panjang area kerja umum (cwa).
`jics.parameters.charset`	string	CP037	JICS secara global menggunakan set karakter.

Kunci	Tipe	Nilai default	Deskripsi
`jics.parameters.tsqimpl`	string	bluesam	Implementasi JICS Temporary Storage Queue (TSQ) (nilai yang diizinkan adalah`bluesam`//) `memory` `redis`
`jics.queues.ts.redis.hostname`	string	127.0.0.1	Nama host dari jics cache redis server.
`jics.queues.ts.redis.port`	nomor	6379	Port dari jics cache redis server.
`jics.queues.ts.redis.password`	string	redis	Kata sandi untuk jics cache redis server.
`jics.queues.ts.redis.username`	string		Nama pengguna untuk server jics cache redis. Default kosong (tidak ada nama pengguna).
`jics.queues.ts.redis.mode`	string	mandiri	Mode cache jics. Nilai yang mungkin adalah `standalone` atau`cluster`. Default adalah`standalone` .
`lockTimeout`	nomor	500	Batas waktu kunci, dalam milidetik.

Kunci	Tipe	Nilai default	Deskripsi
`sqlCodePointShift`	nomor		Opsional. Pergeseran titik kode sql. Menggeser titik kode untuk karakter kontrol yang mungkin kita temui saat memigrasikan data rdbms lama ke rdbms modern. Misalnya, Anda dapat menentukan `384` untuk mencocokkan karakter `\u0180` unicode.
`sqlIntegerOverflowAllowed`	boolean	false	Menentukan apakah untuk memungkinkan SQL integer overflow, yang berarti apakah menempatkan nilai yang lebih besar dalam variabel host diperbolehkan.

Kunci	Tipe	Nilai default	Deskripsi
`database.cursor.overflow.allowed`	boolean	true	Menentukan apakah untuk memungkinkan cursor overflow. Setel `true` untuk melakukan panggilan berikutnya pada kursor apa pun posisinya. Atur `false` untuk memeriksa apakah kursor berada di posisi terakhir sebelum melakukan panggilan berikutnya pada kursor. Hanya aktifkan jika kursor dapat digulir (SENSITIF atau TIDAK SENSITIF).
`reportOutputPath`	string	`/reports`	Jalur keluaran laporan.
`spring.session.store-type`	string	none	Cache sesi untuk lingkungan ketersediaan tinggi. Nilai yang mungkin adalah none atau`redis`. Default adalah`none`.

Kunci	Tipe	Nilai default	Deskripsi
`stopExecutionWhenProgNotFound`	boolean	true	Menentukan apakah akan berhenti berjalan jika program tidak ditemukan. Jika diatur ke`true`, interupsi run jika program tidak ditemukan.
`forceHR`	boolean	false	Menentukan whehter untuk menggunakan Human Readable SYSPRINT, baik pada konsol atau file output.
`rollbackOnRTE`	boolean	false	Menentukan apakah untuk mengembalikan transaksi unit run implisit pada pengecualian runtime.
`sctThreadLimit`	long	5	Batas thread untuk memicu skrip.
`dataSimplifier.onInvalidNumericData`	string	menolak	Bagaimana bereaksi saat mendekode data numerik yang tidak valid. Nilai yang diizinkan adalah`reject`/`tolerates paces` /`tolerates paceslowvalues` /`toleratemost` . Default adalah`reject`.

Kunci	Tipe	Nilai default	Deskripsi
`filesDirectory`	string		Direktori untuk batch input/output file.
`ims.messages.extendedSize`	boolean	false	Menentukan apakah akan mengatur ExtendedSize pada pesan ims.
`defaultKeepExistingFiles`	boolean	false	Menentukan apakah akan mengatur nilai default dataset sebelumnya.
`jics.db.ddlScriptLocation`	string		Lokasi skrip ddl Jics. Memungkinkan Anda untuk memulai skema database jics menggunakan skrip.sql. Kosong secara default. Misalnya, `./jics/sql/jics.sql` .
`jics.db.schemaTestQueryLocation`	string		Lokasi file sql yang harus berisi kueri unik yang mengembalikan jumlah objek dalam skema jics (jika ada).
`jics.db.dataScriptLocation`	string		Lokasi skrip initJics.sql, disiapkan oleh Analyzer dari penguraian ekspor CSD dari mainframe.

Kunci	Tipe	Nilai default	Deskripsi
`jics.db.dataTestQueryLocation`	string		Lokasi skrip sql yang berisi kueri sql tunggal yang diharapkan mengembalikan hitungan objek (misalnya: menghitung jumlah catatan dalam tabel program jics). Jika hitungan sama dengan 0, database akan dimuat menggunakan `jics.db.dataScriptLocation` skrip, jika tidak beban database akan dilewati.
`jics.data.dataJsonInitLocation`	string		
`jics.xa.agent.timeout`	nomor		
`query.useConcatCondition`	boolean	false	Menentukan apakah kondisi kunci dibangun oleh penggabungan kunci atau tidak.
`system.qdecfmt`	string		

Kunci	Tipe	Nilai default	Deskripsi
`dispositi on.checke xistence`	boolean	false	Menentukan apakah akan merilis cek pada keberadaan file untuk Dataset dengan DISP SHR atau OLD.
`useContro lMVariable`	boolean	false	Menentukan apakah akan menggunakan spesifikasi Control-m untuk penggantian variabel.
`card.encoding`	string	CP1145	Pengkodean kartu: untuk digunakan dengan `useContro lMVariable`.
`mapTransf o.prefixes`	string	&,@,%%	Daftar awalan yang akan digunakan saat mengubah variabel ControlM. Masing-masing dipisahkan dengan koma.
`checkinpu tfilesize`	boolean	false	Menentukan apakah akan merilis cek jika ukuran file adalah kelipatan dari ukuran rekaman.
`stepFailW henAbend`	boolean	true	Menentukan apakah akan menaikkan abend jika langkah gagal atau menyelesa ikan eksekusi.

Kunci	Tipe	Nilai default	Deskripsi
`bluesam.fileLoading.commitInterval`	nomor	100000	Interval komit bluesam.
`uppercaseUserInput`	boolean	true	Menentukan apakah input pengguna harus dalam huruf besar.
`jhdb.lterm`	string		Memungkinkan Anda untuk membuat ID terminal logis umum dalam kasus emulasi IMS. Jika tidak disetel maka sessionId digunakan.
`jhdb.identificationCardData`	string		Digunakan untuk hardcode beberapa “data kartu identifikasi operator” ke bidang MID yang ditunjuk oleh parameter CARD. Kosong secara default, tidak ada batasan input.
`encoding`	string	ASCII	Pengkodean yang digunakan dalam proyek (bukan dalam file groovy). Mengharapkan pengkodean yang validCP1047,,IBM930,ASCII..UTF-8

Kunci	Tipe	Nilai default	Deskripsi
`cl.configuration.context.encoding`	string	CP297	Pengkodean file CL. Mengharapkan pengkodean yang validCP1047,,IBM930,ASCII.. UTF-8 Nilai default adalah CP297
`cl.zonedMode`	string	EBCDIC_STRICT	Mode untuk perintah encoding atau decoding control language (CL). Nilai yang diizinkan adalahEBCDIC_STRICT /EBCDIC_MODIFIED /AS400.
`ims.programs`	string		Daftar program IMS yang akan digunakan . Pisahkan setiap parameter dengan titik koma (;) dan setiap transaksi dengan koma (). , Sebagai contoh: PCP008,PCT008;PCP054,PCT054;PCP066,PCT066;PCP068,PCT068;

Kunci	Tipe	Nilai default	Deskripsi
`jhdb.configuration.context.encoding`	string	CP297	Pengkodean JHDB (Java Hierarchical Database). Mengharapkan string pengkodean yang validCP1047,,IBM930,ASCII..UTF-8
`jhdb.metadata.extrapath`	string	berkas:. /setup/	Parameter konfigurasi yang menentukan folder root khusus runtime tambahan untuk folder psbs dan dbds.

Kunci	Tipe	Nilai default	Deskripsi
`jhdb.checkpointPersistence`	string	none	Mode persistensi pos pemeriksaan. Nilai yang diizinkan adalah`none/add/end`. Gunakan `add` untuk mempertahankan pos pemeriksaan saat yang baru dibuat dan ditambahkan ke registri. Gunakan pos pemeriksaan yang `end` terlalu bertahan saat server shutdown. Nilai lain menonaktifkan persistensi. Perhatikan bahwa setiap kali pos pemeriksaan baru ditambahkan ke registri, semua pos pemeriksaan yang ada akan diserialisasi dan file akan dihapus. Ini bukan tambahan ke data yang ada dalam file. Jadi tergantung pada jumlah pos pemeriksaan, itu dapat memiliki beberapa efek pada pertunjukan.

Kunci	Tipe	Nilai default	Deskripsi
`jhdb.checkpointPath`	string	berkas:. /setup/	Jika `jhdb.checkpointPersistence` tidak none maka parameter ini memungkinkan Anda untuk mengatur jalur persistensi pos pemeriksaan (lokasi penyimpanan file checkpoint.dat), semua data pos pemeriksaan yang terkandung dalam registri diserialkan dan dicadangkan dalam file (checkpoint.dat) yang terletak di folder yang disediakan. Perhatikan bahwa hanya data pos pemeriksaan (scriptID, StepID, posisi database dan area pos pemeriksaan) yang terkait dengan cadangan ini.
`jhdb.navigation.cachenexts`	nomor	5000	Durasi cache (dalam milidetik) yang digunakan dalam navigasi hierarkis untuk RDBMS.

Kunci	Tipe	Nilai default	Deskripsi
`jhdb.use-db-prefix`	boolean	true	Menentukan apakah untuk mengaktifkan awalan database dalam navigasi hierarkis untuk RDBMS.
`jhdb.query.limitJoinUsage`	boolean	true	Menentukan apakah akan menggunakan batas bergabung parameter penggunaan pada grafik RDBMS.
`taskExecutor.corePoolSize`	nomor	5	Ketika transaksi di terminal dimulai melalui skrip groovy, thead baru dibuat. Gunakan parameter ini untuk mengatur ukuran kolam inti.
`taskExecutor.maxPoolSize`	nomor	10	Ketika transaksi di terminal dimulai melalui skrip groovy, thead baru dibuat. Gunakan parameter ini untuk mengatur ukuran kolam maks (jumlah maksimum thread paralel).

Kunci	Tipe	Nilai default	Deskripsi
`taskExecutor.queueCapacity`	nomor	50	Ketika transaksi di terminal dimulai melalui skrip groovy, thead baru dibuat. Gunakan parameter ini untuk mengatur ukuran antrian. (= jumlah maksimum transaksi yang tertunda saat `taskExecutor.maxPoolSize` tercapai)
`taskExecutor.allowCoreThreadTimeOut`	boolean	false	Menentukan apakah untuk memungkinkan thread inti untuk time out di JCIS. Hal ini memungkinkan pertumbuhan dan penyusutan dinamis bahkan dalam kombinasi dengan antrian bukan nol (karena ukuran kolam maksimal hanya akan bertambah setelah antrian penuh).
`jics.runUnitLauncherPool.enable`	boolean	false	Menentukan apakah akan mengaktifkan run unit launcher pool di JICS.

Kunci	Tipe	Nilai default	Deskripsi
`jics.runUnitLauncherPool.size`	nomor	20	Ukuran kolam peluncur unit run di JICS.
`jics.runUnitLauncherPool.validationInterval`	nomor	1000	Interval validasi kumpulan peluncur unit run di JICS, dinyatakan dalam milidetik.
`spring.aws.application.credentials`	string	kosong	Muat kredensyal AWS dari file profil kredensyal di JICS.
`jics.queues.sqs.region`	string	eu-west-1	Wilayah AWS untuk AWS Simple Queue Service, digunakan di JICS.
`mq.queues.sqs.region`	string	eu-west-3	Wilayah AWS untuk layanan AWS SQS MQ.
`quartz.scheduler.stand-by-if-error`	boolean	false	Menentukan apakah akan memicu eksekusi pekerjaan jika penjadwal pekerjaan dalam modus siaga. Jika benar, Ketika diaktifkan eksekusi pekerjaan tidak dipicu.

Kunci	Tipe	Nilai default	Deskripsi
`databaseStatistics`	boolean	false	Menentukan apakah akan memungkinkan pembangun SQL untuk mengumpulkan dan menampilkan informasi statistik.
`dbDateFormat`	string	YYYY-MM-DD	Format tanggal target db.
`dbTimeFormat`	string	HH: mm: SS	Format waktu target db.
`dbTimestampFormat`	string	YYYY-MM-DD HH: mm: SS.SSSSSS	Format stempel waktu target db.
`dateTimeFormat`	string	ISO	Ini dateTimeFormat menjelaskan cara menumpahkan tipe stempel waktu tanggal waktu database ke entitas penyederhana data. Nilai yang diizinkan adalah`ISO/EUR/EUR/USA/LOCA`
`localDateFormat`	string		Daftar format tanggal lokal. Pisahkan setiap format dengan. \
`localTimeFormat`	string		Daftar format waktu lokal. Pisahkan setiap format dengan \

Kunci	Tipe	Nilai default	Deskripsi
`localTimestampFormat`	string		Daftar format stempel waktu lokal. Pisahkan setiap format dengan\.
`pgmDateFormat`	string	YYYY-MM-DD	Format waktu tanggal.
`pgmTimeFormat`	string	HH.mm.ss	Format waktu yang digunakan untuk eksekusi pgm (program).
`pgmTimestampFormat`	string	yyyy-mm-dd-hh.mm.ssssss	Format stempel waktu.
`cacheMetadata`	boolean	true	Menentukan apakah untuk cache metadata database.
`forceDisableSQLTrimStringType`	boolean	false	Menentukan apakah untuk menonaktifkan trim dari semua parameter string sql.
`fetchSize`	nomor		Nilai fetchSize untuk kursor. Gunakan saat mengambil data menggunakan potongan dengan memuat/membongkar utilitas.
`check-groovy-file`	boolean	true	Menentukan apakah untuk memeriksa konten file asyik sebelum mendaftar.

Kunci	Tipe	Nilai default	Deskripsi
`qtemp.uuid.length`	nomor	9	Panjang id unik QTEMP.
`qtemp.dblog`	boolean	false	Apakah akan mengaktifkan pencatatan Database QTEMP.
`qtemp.cleanup.threshold.hours`	nomor	0	Untuk menentukan kapan `qtemp.dblog` diaktifkan. Masa pakai partisi db (dalam jam).
`sort.function`	string		Nama fungsi sort untuk database blu4iv.

Properti yang tersedia untuk aplikasi web opsional

Bergantung pada aplikasi modern Anda, Anda mungkin perlu mengonfigurasi satu atau lebih aplikasi web opsional yang mewakili dukungan untuk dependensi seperti z/OS, AS/400, atau IMS/MFS. Tabel berikut berisi daftar parameter kunci/nilai yang tersedia untuk mengkonfigurasi setiap aplikasi web opsional.

gapwalk-utility-pgm.perang

Aplikasi web opsional ini berisi dukungan untuk program utilitas Z/OS.

Tabel ini memberikan tampilan lengkap parameter kunci/nilai untuk aplikasi ini.

Kunci	Tipe	Nilai default	Deskripsi
`logging.config`	Jalur	classpath:logback-utility.xml	Kunci standar untuk referensi ke file konfigurasi logback.

Kunci	Tipe	Nilai default	Deskripsi
			Kunci logging standar lainnya juga tersedia.
`spring.jta.enabled`	boolean	false	Kunci standar. Jika mode dukungan sumber data tidak statis-xa, konfigurasi otomatis transaksi JTA pegas harus dinonaktifkan.
`spring.datasource.primary.jndi-name`	string	jdbc/primer	Nama jndi (Java Naming And Directory Interface) untuk sumber data utama, jika menggunakan JNDI.
`primary.datasource + -driver-class-name + -url + -username + -password`	Sumber data pegas standar dengan subkunci		Berisi informasi koneksi untuk database aplikasi, jika tidak menggunakan JNDI. Harus memiliki konfigurasi yang sama seperti pada file ymlaplikasi modern. Sebagai alternatif, penggunaan rahasia AWS sangat dianjurkan, seperti yang dijelaskan dalam xref:.. /configuration/configuration.adoc [Konfigurasi].

Kunci	Tipe	Nilai default	Deskripsi
`encoding`	string	ASCII	Pengkodean yang digunakan dalam program utilitas. Mengharapkan pengkodean yang validCP1047,,IBM930,ASCII.. UTF-8
`sysPunchEncoding`	string	ASCII	Set karakter pengkodean syspunch. Mengharapkan pengkodean yang validCP1047,,IBM930,ASCII.. UTF-8
`zonedMode`	string	EBCDIC_STRICT	Mode untuk encoding atau decoding tipe data yang dikategorikan. Nilai yang diizinkan adalahEBCDIC_STRICT /EBCDIC_MODIFIED /AS400.
`unload.chunkSize`	nomor	0	Ukuran potongan digunakan untuk utilitas bongkar.

Kunci	Tipe	Nilai default	Deskripsi
`unload.sqlCodePointShift`	nomor	0	Kode SQL pointshift untuk utilitas bongkar. Menjalankan proses pergeseran karakter. Diperlukan ketika database target Anda dari DB2 adalah Postgresql.
`unload.columnFiller`	string	yang lebih besar	Pengisi kolom utilitas bongkar muat.
`unload.varCharIsNull`	boolean	false	Gunakan parameter ini dalam program INFTILB, jika diatur untuk `true` maka semua bidang tidak nullable dengan nilai kosong (spasi) mengembalikan string kosong.
`unload.useDatabaseConfiguration`	boolean	false	Menentukan apakah akan menggunakan konfigurasi tanggal atau waktu dari application-main.yml dalam utilitas bongkar.

Kunci	Tipe	Nilai default	Deskripsi
`unload.format.date`	string	mm/dd/YYYY	Jika `unload.useDatabaseConfiguration` diaktifkan, format tanggal yang akan digunakan dalam utilitas bongkar.
`unload.format.time`	string	HH.mm.ss	Jika `unload.useDatabaseConfiguration` diaktifkan, format waktu untuk digunakan dalam utilitas bongkar.
`unload.format.timestamp`	string	yyyy-mm-dd-hh.mm.ssssss	Jika `unload.useDatabaseConfiguration` diaktifkan, format stempel waktu untuk digunakan dalam utilitas bongkar.
`unload.nbi.whenNull`	heksadesimal	6F	Nilai Null Byte Indicator (nbi) untuk ditambahkan ketika nilai dari database adalah null.
`unload.nbi.whenNotNull`	heksadesimal	00	Nilai Null Byte Indicator (nbi) untuk ditambahkan ketika nilai dari database tidak null.

Kunci	Tipe	Nilai default	Deskripsi
`unload.nbi.writeNullIndicator`	boolean	false	Menentukan apakah akan menulis indikator null dalam file output bongkar.
`unload.fetchSize`	nomor	0	Memungkinkan Anda menyetel ukuran pengambilan saat menangani kursor di utilitas bongkar muat.
`treatLargeNumberAsInteger`	boolean	false	Menentukan apakah untuk memperlakukan jumlah besar sebagai`Integer`. Mereka diperlakukan sebagai `BigDecimal` default.
`load.batchSize`	nomor	0	Ukuran batch utilitas beban.
`load.format.localDate`	string	DD.mm.yyyy\ dd/mm/yyyy\ yyyy-mm-dd	Format tanggal lokal utilitas muat untuk digunakan.
`load.format.localTime`	string	HH: mm: ss\ hh.mm.ss	Format waktu lokal utilitas beban untuk digunakan.
`load.format.dbDate`	string	YYYY-MM-DD	Format database utilitas beban untuk digunakan.

Kunci	Tipe	Nilai default	Deskripsi
`load.format.dbTime`	string	HH: mm: SS	Waktu database utilitas beban untuk digunakan.
`load.sqlCodePointShift`	nomor	0s	Kode SQL pointshift untuk utilitas beban. Menjalankan proses pergeseran karakter. Diperlukan ketika database target Anda dari DB2 adalah Postgresql.
`forcedDate`	string		Memaksa tanggal ke tanggal yang diberikan jika ada.
`frozenDate`	boolean	true	Menentukan apakah untuk membekukan tanggal. Berlaku hanya `forcedDate` jika juga diatur.
`jcl.type`	string	mvs	Jenis file.jcl. Nilai yang diizinkan adalah`jcl`/`vse`. Perintah IDCAMS utilitas PRINT/REPRO mengembalikan 4 jika file kosong untuk non-vse jcl.
`hasGraphic`	boolean	false	Apakah utilitas INFUTILB perlu menangani kolom GRAPHIC DB2.

gapwalk-cl-command.perang

Aplikasi web opsional ini berisi dukungan untuk program utilitas AS/400.

Tabel ini memberikan tampilan lengkap parameter kunci/nilai untuk aplikasi ini.

Kunci	Tipe	Nilai default	Deskripsi
`logging.config`	Jalur	classpath:logback-utility.xml	Kunci standar untuk referensi ke file konfigurasi logback. Kunci logging standar lainnya juga tersedia.
`spring.jta.enabled`	boolean	false	Kunci standar. Jika mode dukungan sumber data tidak statis-xa, konfigurasi otomatis transaksi JTA pegas harus dinonaktifkan.
`spring.datasource.primary.jndi-name`	string	jdbc/primer	Nama jndi (Java Naming And Directory Interface) untuk sumber data utama, jika menggunakan JNDI.
`primary.datasource + -driver-class-name + -url + -username + -password`	Sumber data pegas standar dengan subkunci		Berisi informasi koneksi untuk database aplikasi, jika tidak menggunakan JNDI. Harus memiliki konfigurasi yang sama seperti pada file ymlaplikasi modern.

Kunci	Tipe	Nilai default	Deskripsi
			Sebagai alternatif, penggunaan rahasia AWS sangat dianjurkan, seperti yang dijelaskan dalam xref:.. /configuration/configuration.adoc [Konfigurasi].
`encoding`	string	ASCII	Pengkodean yang digunakan dalam program utilitas. Mengharapkan pengkodean yang validCP1047,,IBM930,ASCII..UTF-8
`zonedMode`	string	EBCDIC_STRICT	Mode untuk encoding atau decoding tipe data yang dikategorikan. Nilai yang diizinkan adalah`EBCDIC_STRICT` /`EBCDIC_MODIFIED` /`AS400`.

Kunci	Tipe	Nilai default	Deskripsi
`commands-off`	string		Daftar komand untuk dimatikan, dipisahkan dengan koma. Nilai yang diizinkan adalah `PGM_BASIC RCVMSG,SNDRCVF,CHGVAR,QC` Berguna saat Anda ingin menonaktifkan atau menimpa program yang ada. `PGM_BASIC` adalah program kecepatan khusus yang dirancang untuk tujuan debug.

gapwalk-hierarchical-support.perang

Aplikasi web opsional ini berisi dukungan transaksi IMS/MFS.

Tabel ini memberikan tampilan lengkap parameter kunci/nilai untuk aplikasi ini.

Kunci	Tipe	Nilai default	Deskripsi
`logging.config`	Jalur	classpath:logback-utility.xml	Kunci standar untuk referensi ke file konfigurasi logback. Kunci logging standar lainnya juga tersedia.
`spring.jta.enabled`	boolean	false	Kunci standar. Jika mode dukungan sumber data tidak statis-xa, konfigurasi otomatis transaksi

Kunci	Tipe	Nilai default	Deskripsi
			JTA pegas harus dinonaktifkan.
`jhdb.configuration.context.encoding`	string		Pengkodean JHDB (Java Hierarchical Database). Mengharapkan string pengkodean yang validCP1047,,IBM930,ASCII..UTF-8

Kunci	Tipe	Nilai default	Deskripsi
`jhdb.checkpointPersistence`	string	none	Mode persistensi pos pemeriksaan. Nilai yang diizinkan adalah`none/add/end`. Gunakan `add` untuk mempertahankan pos pemeriksaan saat yang baru dibuat dan ditambahkan ke registri. Gunakan pos pemeriksaan yang `end` terlalu bertahan saat server shutdown. Nilai lain menonaktifkan persistensi. Perhatikan bahwa setiap kali pos pemeriksaan baru ditambahkan ke registri, semua pos pemeriksaan yang ada akan diseriali sasi dan file akan dihapus. Ini bukan tambahan ke data yang ada dalam file. Jadi tergantung pada jumlah pos pemeriksaan, itu dapat memiliki beberapa efek pada pertunjukan.

Konfigurasikan otentikasi untuk aplikasi Gapwalk

Bagian ini menjelaskan cara mengonfigurasi otentikasi OAuth2 untuk aplikasi Gapwalk menggunakan Penyedia Identitas (IDP) seperti Cognito, Azure AD, Keycloak, dll.

Topik

Prasyarat

Dalam tutorial ini kita akan menggunakan Amazon Cognito sebagai IDP dan PlanetDemo sebagai proyek modern.

Anda dapat menggunakan penyedia identitas eksternal lainnya. ClientRegistration Informasi harus diperoleh dari IDP Anda dan diperlukan untuk otentikasi gapwalk. Untuk informasi selengkapnya, lihat dokumentasi resmi IDP Anda.

ClientRegistration Informasi:

id klien

: id dari ClientRegistration, dalam contoh kita itu akan menjadi PlanetDemo.

rahasia klien

: rahasia klien Anda.

titik akhir otorisasi

: URI titik akhir otorisasi untuk server otorisasi.

titik akhir token

: URI Token Endpoint untuk Server Otorisasi.

titik akhir jwks

: URI digunakan untuk mendapatkan JSON Web Key (JWK) yang berisi kunci untuk memvalidasi JSON Web Signature yang dikeluarkan oleh server otorisasi.

pengalihan URI

URI tempat server otorisasi mengalihkan pengguna akhir jika akses diberikan.

Pengaturan Amazon Cognito

Pertama kita akan membuat dan mengonfigurasi kumpulan pengguna dan pengguna Amazon Cognito yang akan kita gunakan dengan aplikasi gapwalk yang digunakan untuk tujuan pengujian.

> **Note**
>
> Jika Anda menggunakan IDP lain, Anda dapat melewati langkah ini.

Buat kumpulan pengguna

1. Buka Amazon Cognito di AWS Management Console dan autentikasi menggunakan kredensil Anda. AWS
2. Pilih Kolam Pengguna.
3. Pilih Buat kolam pengguna.
4. Di Konfigurasikan pengalaman masuk, pertahankan jenis penyedia default kumpulan pengguna Cognito. Anda dapat memilih satu atau beberapa opsi masuk kumpulan pengguna Cognito; untuk saat ini, pilih Nama pengguna, lalu pilih Berikutnya.
5. Di Konfigurasikan persyaratan keamanan, pertahankan default dan nonaktifkan otentikasi Multi-faktor dengan memilih Tidak ada MFA lalu pilih Berikutnya.
6. Nonaktifkan Aktifkan pendaftaran mandiri sebagai tindakan keamanan dan pilih Berikutnya.
7. Pilih Kirim email dengan Cognito. Pilih Berikutnya.
8. Di Integrasikan aplikasi Anda, pilih nama untuk kumpulan pengguna Anda. Di halaman otentikasi yang di-host, pilih Gunakan UI yang Dihosting Cognito.
9. Demi kesederhanaan, di Domain, pilih Gunakan domain Cognito dan masukkan awalan domain; misalnya,. `https://planetsdemo` Aplikasi demo harus ditambahkan sebagai klien.

    1. Di klien aplikasi awal, pilih Klien rahasia. Masukkan nama klien aplikasi, seperti `` `planetsdemo` `` dan pilih Hasilkan rahasia klien.

2. Di URL callback yang diizinkan, masukkan url untuk mengarahkan pengguna setelah otentikasi. URL http://localhost:8080/planetsdemo sementara juga dapat digunakan, lalu diedit nanti.
3. Simpan nilai default di pengaturan klien aplikasi lanjutan dan bagian izin baca dan tulis Atribut.
4. Pilih Berikutnya.

10. Di Tinjau dan buat, verifikasi pilihan Anda lalu pilih Buat kumpulan pengguna.

Untuk informasi selengkapnya, lihat Membuat kumpulan pengguna.

Pembuatan Pengguna

Karena pendaftaran mandiri dinonaktifkan, buat pengguna Amazon Cognito. Arahkan ke Amazon Cognito di. AWS Management Console Pilih kumpulan pengguna yang Anda buat, lalu di Pengguna pilih Buat pengguna.

Di Informasi pengguna, pilih Kirim undangan email, masukkan nama pengguna dan alamat email, lalu pilih Buat kata sandi. Pilih Create user (Buat pengguna).

Integrasi Amazon Cognito dalam aplikasi Gapwalk

Sekarang setelah kumpulan pengguna dan pengguna Amazon Cognito Anda siap, buka `main-application.yml` file aplikasi modern Anda dan tambahkan kode berikut:

```
spring:
  security:
    oauth2:
      client:
        registration:
          cognito:
            client-id: client-id
            client-name: client-name
            client-secret: client-secret
            provider: cognito
            authorization-grant-type: authorization_code
            scope: openid
            redirect-uri: "{baseUrl}/login/oauth2/code/{registrationId}"
        provider:
          cognito:
            issuerUri: ${gapwalk-application.security.issuerUri}
            authorization-uri: ${gapwalk-application.security.domainName}/oauth2/
authorize
```

```
            jwk-set-uri: ${gapwalk-application.security.issuerUri}/.well-known/
jwks.json
            token-uri: ${gapwalk-application.security.domainName}/oauth2/token
            user-name-attribute: cognito:username
      resourceserver:
        jwt:
          jwk-set-uri: ${gapwalk-application.security.issuerUri}/.well-known/jwks.json

gapwalk-application.security: enabled
gapwalk-application.security.identity: oauth

gapwalk-application.security.issuerUri: https://cognito-idp.region.amazonaws.com/pool-
id
gapwalk-application.security.domainName: your-cognito-domain
```

Ganti placeholder berikut seperti yang dijelaskan:

1. Buka Amazon Cognito di AWS Management Console dan autentikasi menggunakan kredensil Anda. AWS
2. Pilih Kumpulan Pengguna dan pilih kumpulan pengguna yang Anda buat. Anda dapat menemukan *`pool-id Anda di User Pool ID`*.
3. Pilih Integrasi aplikasi di mana Anda dapat menemukan *`your-cognito-domain`*dan pergi ke Klien aplikasi dan analitik dan memilih aplikasi Anda.
4. Di App client: YourApp Anda dapat menemukan ***`client-name, client-id dan client-secret`*** (Tampilkan rahasia klien).
5. *`region-id`* sesuai dengan id wilayah tempat Anda membuat pengguna dan kumpulan pengguna Amazon Cognito`eu-west-3`,; misalnya.
6. Untuk pengalihan, masukkan URI untuk mengarahkan pengguna setelah otentikasi.

Anda dapat menerapkan aplikasi Gapwalk Anda sekarang dan menggunakan pengguna yang dibuat sebelumnya untuk masuk ke aplikasi Anda.

> **Note**
>
> Jika selama login Anda mendapatkan kesalahan dengan pengecualian “Atribut hilang 'cognito:username' in attributes”, hapus baris berikut:: cognito:username user-name-attribute

AWS Blu Age Runtime API

AWS Blu Age Runtime menggunakan beberapa aplikasi web untuk mengekspos titik akhir REST, menyediakan cara untuk berinteraksi dengan aplikasi modern menggunakan klien REST (misalnya memanggil pekerjaan menggunakan penjadwal).

Tujuan dari dokumen ini adalah untuk daftar endpoint REST yang tersedia, memberikan rincian tentang:

- Peran mereka
- Cara menggunakannya dengan benar

Daftar titik akhir disusun ke dalam kategori, tergantung pada sifat layanan yang disediakan dan aplikasi web yang mengekspos titik akhir.

Kami berasumsi bahwa Anda sudah memiliki pengetahuan dasar tentang menggunakan titik akhir REST menggunakan alat khusus seperti POSTMAN, Thunder Client, CURL, browser web, dll...) atau menulis kode Anda sendiri untuk melakukan panggilan API.

Topik

Membangun URL

Setiap aplikasi web di bawah ini mendefinisikan jalur root, dibagikan oleh semua titik akhir. Setiap titik akhir kemudian menambahkan jalur dedikasinya sendiri. URL yang dihasilkan untuk digunakan adalah hasil dari rangkaian jalur. Misalnya, mengingat titik akhir pertama untuk Aplikasi Gapwalk, kami memiliki:

- `/gapwalk-application`untuk jalur aplikasi web root.
- `/scripts`untuk jalur titik akhir khusus.

URL yang dihasilkan untuk digunakan adalah `http://server:port/gapwalk-application/scripts`

server

menunjuk pada nama server (yang menghosting aplikasi web yang diberikan).

port

port yang diekspos oleh server.

Aplikasi Gapwalk-

Endpoint untuk aplikasi web Gapwalk menggunakan jalur root. `/gapwalk-application`

Topik

- Pekerjaan Batch (JCL modern dan sejenisnya) titik akhir terkait
- Titik Akhir Metrik
- Titik Akhir Lainnya
- Endpoint terkait Job Queues

Pekerjaan Batch (JCL modern dan sejenisnya) titik akhir terkait

Pekerjaan batch dapat dijalankan secara sinkron atau asinkron (lihat detail di bawah). Pekerjaan batch sedang dijalankan menggunakan skrip groovy yang merupakan hasil modernisasi skrip warisan (JCL).

Topik

- Daftar skrip yang digunakan
- Luncurkan skrip secara sinkron
- Luncurkan skrip secara asinkron
- Daftar skrip yang dipicu
- Mengambil detail eksekusi pekerjaan
- Daftar skrip yang diluncurkan secara asinkron yang dapat dimatikan
- Daftar skrip yang diluncurkan secara sinkron yang dapat dimatikan

- Membunuh eksekusi pekerjaan yang diberikan
- Daftar pos pemeriksaan yang ada untuk restartabilitas
- Memulai ulang pekerjaan (sinkron)
- Memulai ulang pekerjaan (secara asinkron)
- Menetapkan batas utas untuk eksekusi pekerjaan asinkron

Daftar skrip yang digunakan

- Metode yang didukung: GET
- Jalan: `/scripts`
- Argumen: tidak ada
- Endpoint ini mengembalikan daftar skrip groovy yang digunakan di server, sebagai String. Titik akhir ini terutama dimaksudkan untuk digunakan dari browser web, karena String yang dihasilkan adalah halaman HTML, dengan tautan aktif (tautan per skrip yang dapat diluncurkan - lihat contoh di bawah).

Contoh respons:

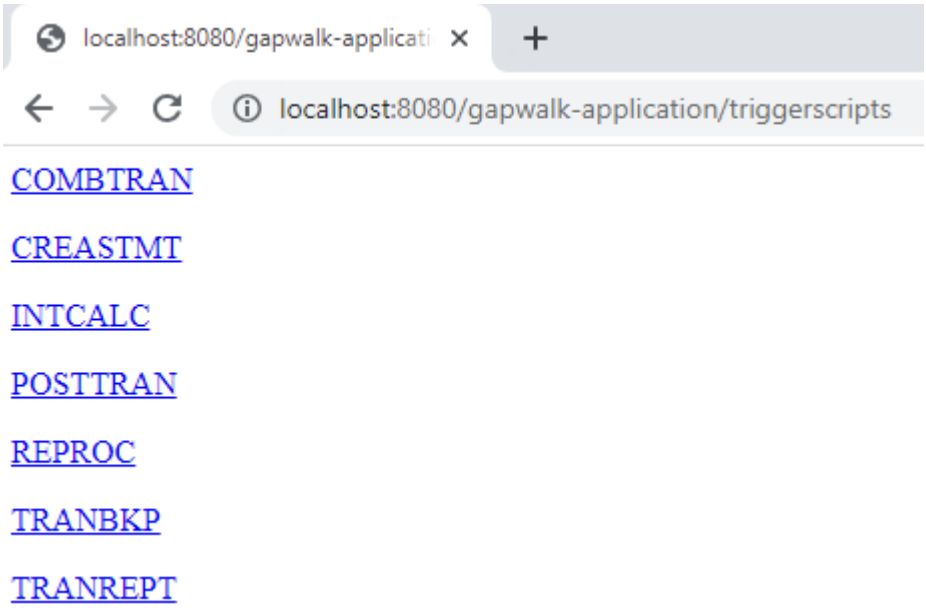
```
<p><a href=./script/COMBTRAN>COMBTRAN</a></p><p><a href=./script/CREASTMT>CREASTMT</
a></p><p><a href=./script/INTCALC>INTCALC</a></p><p><a href=./script/
POSTTRAN>POSTTRAN</a></p><p><a href=./script/REPROC>REPROC</a></p><p><a href=./script/
TRANBKP>TRANBKP</a></p><p><a href=./script/TRANREPT>TRANREPT</a></p><p><a href=./
script/functions>functions</a></p>
```

> **Note**
>
> Tautan mewakili url yang akan digunakan untuk meluncurkan setiap skrip yang terdaftar secara serempak.

- Metode yang didukung: GET
- Jalan: `/triggerscripts`
- Argumen: tidak ada
- Endpoint ini mengembalikan daftar skrip groovy yang digunakan di server, sebagai String. Titik akhir ini terutama dimaksudkan untuk digunakan dari browser web, karena String yang dihasilkan

adalah halaman HTML, dengan tautan aktif (tautan per skrip yang dapat diluncurkan - lihat contoh di bawah).

Berbeda dengan respons titik akhir sebelumnya, tautan mewakili url yang akan digunakan untuk meluncurkan setiap skrip yang terdaftar secara asinkron.



Luncurkan skrip secara sinkron

Titik akhir ini memiliki dua varian dengan jalur khusus untuk penggunaan GET dan POST (lihat di bawah).

- Metode yang didukung: GET
- Jalan: `/script/{scriptId:.+}`
- Metode yang didukung: POST
- Jalan: `/post/script/{scriptId:.+}`
- Pendapat:
  - pengenal skrip yang akan diluncurkan
  - opsional: parameter untuk diteruskan ke skrip, menggunakan parameter permintaan (dilihat sebagai a`Map<String,String>`). Parameter yang diberikan akan secara otomatis ditambahkan ke binding skrip groovy yang dipanggil.

- Panggilan akan meluncurkan skrip dengan pengenal yang diberikan, menggunakan parameter tambahan jika disediakan dan menunggu penyelesaian eksekusi skrip sebelum mengembalikan message (`String`) yang akan berupa:
  - “Selesai.” (jika eksekusi pekerjaan berjalan lancar).
  - Pesan kesalahan JSON dengan rincian tentang apa yang salah selama eksekusi pekerjaan. Rincian lebih lanjut dapat diambil dari log server, untuk memahami apa yang salah dengan eksekusi pekerjaan.

```
{
    "exitCode": -1,
    "stepName": "STEP15",
    "program": "CBACT04C",
    "status": "Error"
}
```

Melihat log server, kita dapat mengetahui bahwa ini adalah masalah penerapan (program yang diharapkan belum diterapkan dengan benar, sehingga tidak dapat ditemukan, membuat eksekusi pekerjaan gagal):

```
2023-06-09 10:27:28 default        INFO  - c.n.b.g.r.s.BatchWebController          - --> executing script INTCALC
2023-06-09 10:27:28 default        INFO  - c.n.b.g.r.s.BatchWebController          - Bound jobContext 419695287 - GDGEventsQueueHandler :907380469
2023-06-09 10:27:28 default        INFO  - c.n.b.g.r.s.ScriptControlTower          - Added jobExecutor [a65c2791-864f-43c9-972a-b5f2353389e6] to Sync Script Control Tower.
2023-06-09 10:27:28 default        INFO  - c.n.b.g.r.j.s.JobExecutor               - a65c2791-864f-43c9-972a-b5f2353389e6 - worker :Thread-26 [1547512424]
2023-06-09 10:27:28 default        INFO  - c.n.b.g.r.j.s.JobExecutor               - Triggered script: INTCALC - [a65c2791-864f-43c9-972a-b5f2353389e6] - jobContext [419695287]
2023-06-09_10-27-29-613 | [JOB] INTCALC - Started
2023-06-09_10-27-29-651 | [STEP] STEP15 - Started
2023-06-09 10:27:29 default        ERROR - c.n.b.g.r.c.i.ExecutionControllerImpl    - Could not find program "CBACT04C" in the program registry.
2023-06-09 10:27:29 default        ERROR - c.n.b.g.r.c.i.ExecutionControllerImpl    - Could not find program "CBACT04C" in the program registry.
2023-06-09_10-27-29-760 | Program not found => not executed !
2023-06-09_10-27-29-761 | [STEP] STEP15 - Ended
2023-06-09_10-27-29-772 | [JOB] INTCALC - Ended
2023-06-09 10:27:29 default        INFO  - c.n.b.g.r.j.s.DefaultJobContext         - Job [419695287] - starting final operation
2023-06-09 10:27:29 default        INFO  - c.n.b.g.r.j.s.DefaultJobContext         - End of job [419695287]
2023-06-09 10:27:29 default        INFO  - c.n.b.g.r.s.ScriptControlTower          - Removed jobExecutor [a65c2791-864f-43c9-972a-b5f2353389e6] from Script Control Tower.
2023-06-09 10:27:29 default        INFO  - c.n.b.g.r.s.ScriptControlTower          - Remaining jobExecutors:0
```

> **Note**
>
> Panggilan sinkron harus dicadangkan untuk pekerjaan yang berjalan dalam waktu singkat. Pekerjaan yang berjalan lama sebaiknya diluncurkan secara asinkron (lihat titik akhir khusus di bawah).

Luncurkan skrip secara asinkron

- Metode yang didukung: GET /POST
- Jalan: `/triggerscript/{scriptId:.+}`
- argumen:

  - pengenal skrip yang akan diluncurkan
  - opsional: parameter untuk diteruskan ke skrip, menggunakan parameter permintaan (dilihat sebagai a`Map<String,String>`). Parameter yang diberikan akan ditambahkan secara otomatis ke https://docs.groovy-lang.org/latest/html/api/groovy/lang/Binding.html[bindings] dari skrip groovy yang dipanggil.
- Berbeda dengan mode sinkron di atas, titik akhir tidak menunggu eksekusi pekerjaan selesai untuk mengirim respons. Eksekusi pekerjaan diluncurkan sekaligus, jika utas yang tersedia dapat ditemukan untuk melakukannya, dan respons dikirim segera ke pemanggil, dengan id eksekusi pekerjaan, pengenal unik yang mewakili eksekusi pekerjaan, yang dapat digunakan untuk menanyakan status eksekusi pekerjaan atau memaksa mematikan eksekusi pekerjaan yang seharusnya tidak berfungsi. Format responsnya adalah:

```
Triggered script <script identifier> [unique job execution id] @ <date and time>
```

- Karena eksekusi asinkron pekerjaan bergantung pada jumlah utas terbatas yang tetap, eksekusi pekerjaan mungkin tidak diluncurkan jika tidak ada utas yang tersedia yang dapat ditemukan. Dalam hal ini, pesan yang dikembalikan akan terlihat seperti:

```
Script [<script identifier>] NOT triggered - Thread limit reached (<actual thread
 limit>) - Please retry later or increase thread limit.
```

  Lihat `settriggerthreadlimit` titik akhir di bawah ini untuk mempelajari cara meningkatkan batas utas.

Contoh respons:

```
Triggered script INTCALC [d43cbf46-4255-4ce2-aac2-79137573a8b4] @ 06-12-2023 16:26:15
```

Pengidentifikasi eksekusi pekerjaan yang unik memungkinkan untuk dengan cepat mengambil entri log terkait di log server jika diperlukan. Ini juga digunakan oleh beberapa titik akhir lainnya yang dirinci di bawah ini.

Daftar skrip yang dipicu

- Metode yang didukung: GET
- Jalan:`/triggeredscripts/{status:.+}`, `/triggeredscripts/{status:.+}/{namefilter}`

- Pendapat:
  - Status (wajib): status skrip yang dipicu untuk diambil. Nilai yang mungkin adalah:
    - `all`: tampilkan semua detail pelaksanaan pekerjaan, apakah pekerjaan masih berjalan atau tidak.
    - `running`: hanya tampilkan detail pekerjaan untuk pekerjaan yang sedang berjalan.
    - `done`: hanya menampilkan detail pekerjaan untuk pekerjaan yang pelaksanaannya selesai.
    - `killed`: hanya menampilkan detail pekerjaan untuk pekerjaan yang pelaksanaannya telah dibunuh secara paksa menggunakan titik akhir khusus (lihat di bawah).
    - `triggered`: hanya menampilkan detail pekerjaan untuk pekerjaan yang telah dipicu tetapi belum diluncurkan.
    - `failed`: hanya menampilkan rincian pekerjaan untuk pekerjaan yang pelaksanaannya telah ditandai sebagai gagal.
    - _namefilter (opsional) _: ambil hanya eksekusi untuk pengidentifikasi skrip yang diberikan.
- Mengembalikan koleksi rincian eksekusi pekerjaan sebagai JSON. Untuk informasi selengkapnya, lihat Struktur pesan Detail Eksekusi Job.

Contoh respons:

```
[
    {
      "scriptId": "INTCALC",
      "caller": "127.0.0.1",
      "identifier": "d43cbf46-4255-4ce2-aac2-79137573a8b4",
      "startTime": "06-12-2023 16:26:15",
      "endTime": "06-12-2023 16:26:15",
      "status": "DONE",
      "executionResult": "{ \"exitCode\": -1, \"stepName\": \"STEP15\", \"program\":
 \"CBACT04C\", \"status\": \"Error\" }",
      "executionMode": "ASYNCHRONOUS"
    }
  ]
```

Mengambil detail eksekusi pekerjaan

- Metode yang didukung: GET
- Jalan: `/getjobexecutioninfo/{jobexecutionid:.+}`

- Pendapat:
  - jobexecutionid (wajib): pengidentifikasi eksekusi pekerjaan unik untuk mengambil detail pelaksanaan pekerjaan yang sesuai.
- Pengembalian: string JSON yang mewakili rincian eksekusi pekerjaan tunggal (lihatStruktur pesan Detail Eksekusi Job) atau respon kosong jika tidak ada rincian eksekusi pekerjaan dapat ditemukan untuk identifier yang diberikan.

Daftar skrip yang diluncurkan secara asinkron yang dapat dimatikan

- Metode yang didukung: GET
- Jalan: `/killablescripts`
- Mengembalikan kumpulan pengidentifikasi eksekusi pekerjaan yang telah diluncurkan secara asinkron yang masih berjalan dan dapat dimatikan secara paksa (lihat titik akhir di bawah). `/kill`

Daftar skrip yang diluncurkan secara sinkron yang dapat dimatikan

- Metode yang didukung: GET
- Jalan: `/killablesyncscripts`
- Mengembalikan koleksi pengidentifikasi eksekusi pekerjaan dari pekerjaan yang telah diluncurkan secara serempak, saat ini masih berjalan dan dapat dimatikan secara paksa (lihat titik `/kill` akhir di bawah).

Membunuh eksekusi pekerjaan yang diberikan

- Metode yang didukung: GET
- Jalan: `/kill/{identifier:.+}`
- argumen: pengenal eksekusi pekerjaan (wajib): pengidentifikasi eksekusi pekerjaan unik untuk menunjuk pada eksekusi pekerjaan yang akan dibunuh secara paksa.
- Pengembalian: pesan tekstual yang merinci eksekusi pekerjaan membunuh hasil percobaan; pesan akan berisi pengidentifikasi skrip, pengidentifikasi unik eksekusi pekerjaan dan tanggal dan waktu di mana eksekusi mati terjadi. Jika tidak ada eksekusi pekerjaan yang berjalan dapat ditemukan untuk pengenal yang diberikan, pesan kesalahan akan dikembalikan sebagai gantinya.

⚠ Warning

- Runtime melakukan upaya terbaiknya untuk membunuh eksekusi pekerjaan target dengan baik. Dengan demikian, respons dari titik akhir /kill mungkin membutuhkan sedikit waktu untuk mencapai penelepon, karena runtime AWS Blu Age akan mencoba meminimalkan dampak bisnis dari pembunuhan pekerjaan.
- Pembunuhan paksa eksekusi pekerjaan tidak boleh dilakukan dengan mudah, karena mungkin memiliki konsekuensi bisnis langsung, termasuk kemungkinan kehilangan data atau korupsi. Ini harus dicadangkan untuk kasus-kasus di mana eksekusi pekerjaan tertentu telah berjalan menyamping dan sarana remediasi data diidentifikasi dengan jelas.
- Membunuh pekerjaan harus mengarah pada penyelidikan lebih lanjut (analisis post-mortem) untuk mencari tahu apa yang salah dan mengambil tindakan remediasi yang tepat.
- Bagaimanapun, upaya untuk mematikan pekerjaan yang sedang berjalan akan dicatat di log server dengan pesan tingkat peringatan.

Daftar pos pemeriksaan yang ada untuk restartabilitas

Job Restartability bergantung pada kemampuan skrip untuk mendaftarkan pos pemeriksaan `CheckpointRegistry` untuk melacak kemajuan pelaksanaan pekerjaan. Jika eksekusi pekerjaan gagal berakhir dengan benar, dan memulai kembali pos pemeriksaan telah terdaftar, seseorang dapat dengan mudah memulai ulang eksekusi pekerjaan dari pos pemeriksaan terdaftar terakhir yang diketahui (tanpa harus menjalankan langkah-langkah di atas pos pemeriksaan).

- Metode yang didukung: GET
- Jalan: `/restarts`
- Mengembalikan daftar titik restart yang ada, yang dapat digunakan untuk memulai kembali pekerjaan yang pelaksanaannya tidak datang dan berakhir dengan benar, sebagai halaman html. Jika tidak ada pos pemeriksaan yang terdaftar oleh skrip apa pun, konten halaman akan menjadi “Tidak ada pos pemeriksaan terdaftar.”.

Memulai ulang pekerjaan (sinkron)

- Metode yang didukung: GET
- Jalan: `/restart/{hashcode}`

- Argumen: kode hash (integer - wajib): restart eksekusi pekerjaan yang sebelumnya dibatalkan, menggunakan kode hash yang disediakan sebagai nilai pos pemeriksaan (lihat `/restarts` titik akhir di atas untuk mempelajari cara mengambil nilai pos pemeriksaan yang valid).
- Pengembalian: lihat deskripsi `script` kembali di atas.

Memulai ulang pekerjaan (secara asinkron)

- Metode yang didukung: GET
- Jalan: `/triggerrestart/{hashcode}`
- Argumen: kode hash (integer - wajib): restart eksekusi pekerjaan yang sebelumnya dibatalkan, menggunakan kode hash yang disediakan sebagai nilai pos pemeriksaan (lihat `/restarts` titik akhir di atas untuk mempelajari cara mengambil nilai pos pemeriksaan yang valid).
- Pengembalian: lihat deskripsi `triggerscript` kembali di atas.

Menetapkan batas utas untuk eksekusi pekerjaan asinkron

Eksekusi asinkron pekerjaan bergantung pada kumpulan utas khusus di JVM. Kumpulan itu memiliki batas tetap mengenai jumlah utas yang tersedia. Yang digunakan memiliki kemampuan untuk menyesuaikan batas sesuai dengan kemampuan host (jumlah CPU, memori yang tersedia, dll...). Secara default, batas utas diatur ke 5 utas.

- Metode yang didukung: GET
- Jalan: `/settriggerthreadlimit/{threadlimit:.+}`
- Argumen (integer): batas thread baru untuk diterapkan. Harus berupa bilangan bulat yang sangat positif.
- Mengembalikan pesan (`String`) memberikan batas thread baru dan yang sebelumnya, atau en pesan kesalahan jika nilai batas thread yang disediakan tidak valid (bukan integer benar-benar positif).

Contoh respons:

```
Set thread limit for Script Tower Control to 10 (previous value was 5)
```

Menghitung saat ini berjalan dipicu eksekusi pekerjaan

- Metode yang didukung: GET

- Jalan: `/countrunningtriggeredscripts`
- Mengembalikan pesan yang menunjukkan jumlah pekerjaan yang berjalan yang diluncurkan secara asinkron dan batas utas (yaitu jumlah maksimum pekerjaan yang dipicu yang dapat dijalankan secara bersamaan).

Contoh respons:

```
0 triggered script(s) running (limit =10)
```

> **Note**
>
> Ini dapat digunakan untuk memeriksa, sebelum meluncurkan pekerjaan, jika batas utas belum tercapai (yang akan mencegah pekerjaan diluncurkan).

Membersihkan informasi eksekusi pekerjaan

Informasi eksekusi pekerjaan tetap berada di memori server selama server aktif. Mungkin lebih mudah untuk membersihkan informasi tertua dari memori, karena tidak relevan lagi; inilah tujuan dari titik akhir ini.

- Metode yang didukung: GET
- Jalan: `/purgejobinformation/{age:.+}`
- Argumen: nilai integer yang sangat positif yang mewakili usia dalam jam informasi yang akan dibersihkan.
- Mengembalikan pesan dengan informasi berikut:
  - Nama file pembersihan tempat informasi pelaksanaan pekerjaan yang dibersihkan disimpan untuk tujuan pengarsipan.
  - Jumlah informasi pelaksanaan pekerjaan yang dibersihkan.
  - Jumlah informasi pelaksanaan pekerjaan yang tersisa dalam memo

Titik Akhir Metrik

JVM

Titik akhir ini mengembalikan metrik yang tersedia terkait dengan JVM.

- Metode yang didukung: GET
- Jalan: `/metrics/jvm`
- Argumen: tidak ada
- Mengembalikan pesan dengan informasi berikut:
  - threadActiveCount: Jumlah utas aktif.
  - jvmMemoryUsedMemori aktif digunakan oleh Java Virtual Machine.
  - jvmMemoryMax: Memori maksimum yang diizinkan untuk Java Virtual Machine.
  - jvmMemoryFree: Memori yang tersedia saat ini tidak digunakan oleh Java Virtual Machine.

Sesi

Titik akhir ini mengembalikan metrik yang terkait dengan sesi HTTP yang sedang dibuka.

- Metode yang didukung: GET
- Jalan: `/metrics/session`
- Argumen: tidak ada
- Mengembalikan pesan dengan informasi berikut:
  - SessionCount: Jumlah sesi pengguna aktif yang saat ini dikelola oleh server.

Batch

- Metode yang didukung: GET
- Jalan: `/metrics/batch`
- Pendapat:
  - startTimeStamp (opsional, angka): Memulai stempel waktu untuk pemfilteran data.
  - EndTimeStamp (opsional, angka): Mengakhiri stempel waktu untuk pemfilteran data.
  - halaman (opsional, nomor): Nomor halaman untuk pagination.
  - PageSize (opsional, nomor): Jumlah item per halaman dalam pagination.
- Mengembalikan pesan dengan informasi berikut:
  - content: Daftar metrik eksekusi batch.
  - PageNumber: Nomor halaman saat ini dalam pagination.
  - PageSize: Jumlah item yang ditampilkan per halaman.
  - TotalPages: Jumlah total halaman yang tersedia.

- numberOfElements: Hitungan item pada halaman saat ini.
- terakhir: Bendera Boolean untuk halaman terakhir.
- pertama: Bendera Boolean untuk halaman pertama.

Transaksi

- Metode yang didukung: GET
- Jalan: `/metrics/transaction`
- Pendapat:
  - startTimeStamp (opsional, angka): Memulai stempel waktu untuk pemfilteran data.
  - EndTimeStamp (opsional, angka): Mengakhiri stempel waktu untuk pemfilteran data.
  - halaman (opsional, nomor): Nomor halaman untuk pagination.
  - PageSize (opsional, nomor): Jumlah item per halaman dalam pagination.
- Mengembalikan pesan dengan informasi berikut:
  - content: Daftar metrik eksekusi transaksi.
  - PageNumber: Nomor halaman saat ini dalam pagination.
  - PageSize: Jumlah item yang ditampilkan per halaman.
  - TotalPages: Jumlah total halaman yang tersedia.
  - numberOfElements: Hitungan item pada halaman saat ini.
  - terakhir: Bendera Boolean untuk halaman terakhir.
  - pertama: Bendera Boolean untuk halaman pertama.

Titik Akhir Lainnya

Gunakan titik akhir ini untuk mencantumkan daftar program atau layanan terdaftar, menemukan status kesehatan, dan mengelola transaksi JICS.

Topik

- Luncurkan transaksi JICS
- Luncurkan transaksi JICS (alternatif)

Daftar program terdaftar

- Metode yang didukung: GET
- Jalan: `/programs`
- Mengembalikan daftar program terdaftar, sebagai halaman html. Setiap program ditunjuk oleh pengidentifikasi program utamanya. Baik program warisan modern dan program utilitas (IDCAMS, IEBGENER, dll...) dikembalikan dalam daftar. Harap dicatat bahwa program utilitas yang tersedia akan tergantung pada aplikasi web utilitas yang telah digunakan di server tomcat Anda. Misalnya, program dukungan utilitas z/OS mungkin tidak tersedia untuk aset iSeries yang dimodernisasi, karena tidak relevan.

Daftar layanan terdaftar

- Metode yang didukung: GET
- Jalan: `/services`
- Mengembalikan daftar layanan runtime terdaftar, sebagai halaman html. Layanan yang diberikan dibawa oleh runtime AWS Blu Age sebagai utilitas, yang dapat digunakan misalnya dalam skrip groovy. Layanan pemuatan Blusam (untuk membuat kumpulan data Blusam dari kumpulan data lama) termasuk dalam kategori itu.

Contoh respons:

```
<p>BluesamESDSFileLoader</p><p>BluesamKSDSFileLoader</p><p>BluesamRRDSFileLoader</p>
```

Status kondisi

- Metode yang didukung: GET
- Jalan: `/`
- Mengembalikan pesan sederhana, menunjukkan bahwa gapwalk-aplikasi aktif dan berjalan () `Jics application is running.`

Daftar transaksi JICS yang tersedia

- Metode yang didukung: GET
- Jalan: `/transactions`
- Mengembalikan halaman html yang mencantumkan semua transaksi JICS yang tersedia. Ini hanya masuk akal untuk lingkungan dengan elemen JICS (modernisasi elemen CICS lama).

Contoh respons:

```
<p>INQ1</p><p>MENU</p><p>MNT2</p><p>ORD1</p><p>PRNT</p>
```

Luncurkan transaksi JICS

- Metode yang didukung: GET, POST
- Jalan: `/jicstransrunner/{jtrans:.+}`
- argumen:
  - Pengidentifikasi transaksi JICS (string, wajib): pengenal transaksi JICS yang akan diluncurkan (panjang 8 karakter maksimal)
  - diperlukan: data input tambahan untuk diteruskan ke transaksi, sebagai Peta<String, Object>. Isi peta ini akan digunakan untuk memberi makan COMMAREA yang akan dikonsumsi oleh transaksi JICS. Peta bisa kosong jika tidak ada data yang diperlukan untuk menjalankan transaksi.
  - opsional: entri header Http, untuk menyesuaikan lingkungan run untuk transaksi yang diberikan. Tombol header berikut sedang didukung:
    - `jics-channel`: Nama JICS CHANNEL yang akan digunakan oleh program yang akan diluncurkan oleh peluncuran transaksi ini.
    - `jics-container`: Nama JICS CONTAINER yang akan digunakan untuk peluncuran transaksi JICS ini.
    - `jics-startcode`: STARTCODE (String, hingga 2 karakter) untuk digunakan pada awal transaksi JICS. Lihat STARTCODE untuk nilai yang mungkin (telusuri halaman).
    - `jicxa-xid`: XID (X/Open transaction identifier XID structure) dari “transaksi global” (XA), yang diprakarsai oleh penelepon, dimana peluncuran transaksi JICS saat ini akan berpartisipasi.

- Pengembalian: serialisasi `com.netfective.bluage.gapwalk.rt.shared.web.TransactionResultBean` JSON, yang mewakili hasil peluncuran transaksi JICS.

Untuk informasi lebih lanjut tentang detail struktur, lihatStruktur hasil peluncuran transaksi.

Luncurkan transaksi JICS (alternatif)

- metode yang didukung: GET, POST
- jalan: `/jicstransaction/{jtrans:.+}`
- argumen:

  Pengidentifikasi transaksi JICS (string, wajib)

  pengenal transaksi JICS yang akan diluncurkan (panjang maksimal 8 karakter)

  diperlukan: data input tambahan untuk diteruskan ke transaksi, sebagai Peta<String, Object>

  Isi peta ini akan digunakan untuk memberi makan COMMAREA yang akan dikonsumsi oleh transaksi JICS. Peta bisa kosong jika tidak ada data yang diperlukan untuk menjalankan transaksi.

  opsional: entri header Http, untuk menyesuaikan lingkungan run untuk transaksi yang diberikan.

  Tombol header berikut sedang didukung:

  - `jics-channel`: Nama JICS CHANNEL yang akan digunakan oleh program yang akan diluncurkan oleh peluncuran transaksi ini.
  - `jics-container`: Nama JICS CONTAINER yang akan digunakan untuk peluncuran transaksi JICS ini.
  - `jics-startcode`: STARTCODE (String, hingga 2 karakter) untuk digunakan pada awal transaksi JICS. Untuk nilai yang mungkin, lihat STARTCODE (telusuri halaman).
  - `jicxa-xid`: XID (X/Open transaction identifier XID structure) dari “transaksi global” (XA), yang diprakarsai oleh penelepon, dimana peluncuran transaksi JICS saat ini akan berpartisipasi.
- return: serialisasi `com.netfective.bluage.gapwalk.rt.shared.web.RecordHolderBean` JSON, yang mewakili hasil peluncuran transaksi JICS. Rincian struktur dapat ditemukan diStruktur hasil catatan peluncuran transaksi.

Endpoint terkait Job Queues

Job Queues adalah dukungan AWS Blu Age untuk mekanisme pengajuan pekerjaan AS400. Job Queues digunakan di AS400 untuk menjalankan pekerjaan pada kumpulan thread tertentu. Antrian pekerjaan ditentukan oleh nama dan jumlah maksimum utas yang sesuai dengan jumlah maksimum program yang dapat dijalankan secara bersamaan pada antrian itu. Jika lebih banyak pekerjaan dikirimkan pada antrian daripada jumlah maksimum utas, pekerjaan akan menunggu utas tersedia.

Untuk daftar lengkap status pekerjaan dalam antrian, lihat. Kemungkinan status pekerjaan dalam antrian

Operasi pada antrian pekerjaan ditangani melalui titik akhir khusus berikut. Anda dapat menjalankan operasi ini dari URL Aplikasi Gapwalk dengan URL root berikut:. `http://server:port/gapwalk-application/jobqueue`

Topik

- Daftar antrian yang tersedia
- Memulai atau memulai ulang antrian pekerjaan
- Kirim pekerjaan untuk peluncuran
- Daftar semua pekerjaan terjadwal
- Buat daftar semua pekerjaan yang “ditahan”
- Daftar semua pekerjaan aktif
- Daftar semua pekerjaan yang menunggu untuk diluncurkan
- Lepaskan semua pekerjaan yang “ditahan”
- Lepaskan semua pekerjaan yang “ditahan” untuk nama pekerjaan tertentu
- Lepaskan pekerjaan yang diberikan untuk nomor pekerjaan

Daftar antrian yang tersedia

- Metode yang didukung: GET
- Jalan: `list-queues`
- Mengembalikan daftar antrian yang tersedia bersama dengan statusnya, sebagai daftar JSON nilai kunci.

Contoh respons:

```
{"Default":"STAND_BY","queue1":"STARTED","queue2":"STARTED"}
```

Status yang mungkin untuk antrian pekerjaan adalah:

SIAGA

antrian pekerjaan sedang menunggu untuk dimulai.

DIMULAI

antrian pekerjaan aktif dan berjalan.

TIDAK DIKETAHUI

status antrian pekerjaan tidak dapat ditentukan.

Memulai atau memulai ulang antrian pekerjaan

- Metode yang didukung: POST
- Jalan: `/restart/{name}`
- Argumen: nama antrian yang akan dimulai/dimulai ulang, sebagai String - wajib.
- Titik akhir tidak mengembalikan apa pun melainkan bergantung pada status http untuk menunjukkan hasil operasi start/restart:

  HTTP 200

  operasi start/restart berjalan dengan baik: antrian pekerjaan yang diberikan sekarang DIMULAI.

  HTTP 404

  antrian pekerjaan tidak ada.

  HTTP 503

  pengecualian terjadi selama upaya start/restart (log server harus diperiksa untuk mencari tahu apa yang salah).

Kirim pekerjaan untuk peluncuran

- Metode yang didukung: POST

- Argumen: wajib sebagai badan permintaan, serialisasi JSON dari suatu `com.netfective.bluage.gapwalk.rt.jobqueue.SubmitJobMessage` objek. Untuk informasi selengkapnya, lihat Kirim masukan pekerjaan.
- Pengembalian: JSON yang berisi asli `SubmitJobMessage` dan log yang menunjukkan apakah pekerjaan telah dikirimkan atau tidak.

Daftar semua pekerjaan terjadwal

- Metode yang didukung: GET
- Jalan: `list-jobs`
- Pengembalian: daftar semua pekerjaan terjadwal, sebagai string JSON. Untuk respons sampel, lihatDaftar respon pekerjaan terjadwal.

Buat daftar semua pekerjaan yang “ditahan”

- Metode yang didukung: GET
- Jalan: `list-jobs-hold`
- Pengembalian: daftar semua pekerjaan terjadwal, sebagai string JSON. Untuk respons sampel, lihatDaftar respons pekerjaan “ditahan”.

Daftar semua pekerjaan aktif

- Metode yang didukung: GET
- Jalan: `list-jobs-active`
- Pengembalian: daftar semua pekerjaan dengan status ACTIVE, sebagai string JSON. Responsnya mirip dalam struktur dengan yang dariDaftar respon pekerjaan terjadwal.

Daftar semua pekerjaan yang menunggu untuk diluncurkan

- Metode yang didukung: GET
- Jalan: `list-jobs-waiting`
- Pengembalian: daftar semua pekerjaan dengan status EXECUTION_WAIT (pekerjaan yang menunggu thread tersedia untuk diluncurkan), sebagai string JSON. Responsnya mirip dalam struktur dengan yang darithe section called “Daftar respon pekerjaan terjadwal”.

Lepaskan semua pekerjaan yang “ditahan”

- Metode yang didukung: POST
- Jalan: `release-all`
- Returns: pesan yang menunjukkan hasil untuk operasi percobaan rilis. Dua kemungkinan kasus di sini:
  - HTTP 200 dan pesan “Semua pekerjaan dirilis dengan sukses!” jika semua pekerjaan berhasil dilepaskan.
  - HTTP 503 dan pesan “Pekerjaan tidak dirilis. Terjadi kesalahan yang tidak diketahui. Lihat log untuk detail selengkapnya” jika ada yang tidak beres dengan upaya rilis.

Lepaskan semua pekerjaan yang “ditahan” untuk nama pekerjaan tertentu

<job name, job number>Untuk nama pekerjaan tertentu, beberapa pekerjaan dapat diajukan, dengan nomor pekerjaan yang berbeda (keunikan dari pekerjaan yang dijalankan dijamin oleh pasangan). Titik akhir akan mencoba untuk merilis semua kiriman pekerjaan dengan nama pekerjaan yang diberikan, yang “ditahan”.

- Metode yang didukung: POST
- Jalan: `/release/{name}`
- Argumen: nama pekerjaan yang harus dicari, sebagai string. Wajib.
- Returns: pesan yang menunjukkan hasil untuk operasi percobaan rilis. Dua kemungkinan kasus di sini:
  - HTTP 200 dan pesan “Pekerjaan dalam grup <name>(<number of released jobs>) dirilis dengan sukses!” Pekerjaan berhasil dilepaskan.
  - HTTP 503 dan pesan “Pekerjaan dalam grup <name>tidak dirilis. Terjadi kesalahan yang tidak diketahui. Lihat log untuk detail selengkapnya” jika ada yang tidak beres dengan upaya rilis.

Lepaskan pekerjaan yang diberikan untuk nomor pekerjaan

<job name, job number>Titik akhir akan mencoba untuk merilis pengajuan pekerjaan unik yang “ditahan”, untuk pasangan yang diberikan.

- Metode yang didukung: POST
- Jalan: `/release/{name}/{number}`

- Pendapat:

  name

  nama pekerjaan yang harus dicari, sebagai string. Wajib.

  nomor

  nomor pekerjaan yang harus dicari, sebagai bilangan bulat. Wajib.

  pulang

  pesan yang menunjukkan hasil untuk operasi percobaan rilis. Dua kemungkinan kasus di sini:

  - HTTP 200 dan pesan “" Job <name/number> dirilis dengan sukses!” jika pekerjaan itu berhasil dilepaskan.
  - HTTP 503 dan pesan “Job <name/number> not released. Terjadi kesalahan yang tidak diketahui. Lihat log untuk detail selengkapnya” jika ada yang tidak beres dengan upaya rilis.

Titik akhir REST Konsol Aplikasi Blusam

Konsol Aplikasi Blusam adalah API yang dirancang untuk menyederhanakan pengelolaan kumpulan data VSAM yang dimodernisasi. Titik akhir untuk aplikasi web Blusam menggunakan jalur root. `/bac`

Topik

Set data titik akhir terkait

Gunakan titik akhir berikut untuk membuat atau mengelola kumpulan data tertentu.

Topik

- Memuat satu set data
- Memuat kumpulan data dari bucket Amazon S3
- Mengekspor kumpulan data ke bucket Amazon S3
- Hapus kumpulan data
- Hapus kumpulan data
- Hitung catatan kumpulan data

Buat kumpulan data

Membuat titik akhir kumpulan data memungkinkan untuk membuat definisi kumpulan data, dan memerlukan otentikasi.

- Metode yang didukung: POST
- Jalan: `/api/services/rest/bluesamservice/createDataSet`
- Pendapat:

  name

  (required, string): nama kumpulan data.

  tipe

  (wajib, string): tipe kumpulan data. Nilai yang mungkin adalah:ESDS,KSDS,RRDS.

  RecordSize

  (opsional, string): Ukuran maksimum setiap catatan kumpulan data.

  FixedLength

  (opsional, boolean): Menunjukkan jika panjang catatan diperbaiki.

  Kompresi

  (opsional, boolean): Menunjukkan apakah kumpulan data dikompresi.

  CacheEnable

  (opsional, boolean): Menunjukkan apakah caching diaktifkan untuk kumpulan data.

  AlternativeKeys

  (opsional, daftar kunci):

    - offset (wajib, nomor)
    - panjang (diperlukan, nomor)
    - nama (wajib, nomor)
- Mengembalikan file json yang mewakili set data yang baru dibuat.

Permintaan sampel:

```
POST /api/services/rest/bluesamservice/createDataSet
{
  "name": "DATASET",
  "checked": false,
  "records": [],
  "primaryKey": {
    "name": "PK"
  },
  "alternativeKeys": [
    {
      "offset": 10,
      "length": 10,
      "name": "ALTK_0"
    }
  ],
  "type": "ESDS",
  "recordSize": 10,
  "compression": true,
  "cacheEnable": true
}
```

Contoh respons:

```
{
    "dataSet": {
      "name": "DATASET",
      "checked": false,
      "nbRecords": 0,
      "keyLength": -1,
      "recordSize": 10,
      "compression": false,
      "fixLength": true,
      "type": "ESDS",
      "cacheEnable": false,
```

```
      "cacheWarmup": false,
      "cacheEviction": "100ms",
      "creationDate": 1686744961234,
      "modificationDate": 1686744961234,
      "records": [],
      "primaryKey": {
        "name": "PK",
        "offset": null,
        "length": null,
        "columns": null,
        "unique": true
      },
      "alternativeKeys": [
        {
          "offset": 10,
          "length": 10,
          "name": "ALTK_0"
        }
      ],
      "readLimit": 0,
      "readEncoding": null,
      "initCharacter": null,
      "defaultCharacter": null,
      "blankCharacter": null,
      "strictZoned": null,
      "decimalSeparator": null,
      "currencySign": null,
      "pictureCurrencySign": null
    },
    "message": null,
    "result": true
  }
```

Unggah sebuah file

Endpoint ini memungkinkan untuk meng-upload file ke server. File disimpan dalam folder sementara yang sesuai dengan setiap pengguna tertentu. Titik akhir ini harus digunakan setiap kali file diperlukan untuk diunggah. Hal ini membutuhkan otentikasi.

- Metode yang didukung: POST
- Jalan: `/api/services/rest/bluesamservice/upload`
- Pendapat:

file

(wajib, multipart/form-data): File yang akan diunggah.

- Mengembalikan boolean mencerminkan status upload

Memuat satu set data

Setelah definisi kumpulan data dibuat menggunakan titik akhir kumpulan data buat yang dijelaskan sebelumnya, Anda dapat memuat catatan yang terkait dengan file yang diunggah, ke kumpulan data tertentu. Hal ini membutuhkan otentikasi.

- Metode yang didukung: POST
- Jalan: `/api/services/rest/bluesamservice/loadDataSet`
- Pendapat:

  name

  (required, string): nama kumpulan data.
- Mengembalikan status permintaan dan kumpulan data yang dimuat.

Memuat kumpulan data dari bucket Amazon S3

Memuat kumpulan data menggunakan file listcat dari bucket Amazon S3.

- Metode yang didukung: GET
- Jalan: `` `/api/services/rest/bluesamservice/loadDataSetFromS3` ``
- Pendapat:

  ListCatFiles3Lokasi

  (wajib, string): lokasi Amazon S3 dari file listcat.

  DatasetFiles3Lokasi

  (wajib, string): lokasi Amazon S3 dari file kumpulan data.

  region

  (wajib, string): Amazon S3 Wilayah AWS tempat file disimpan.
- Mengembalikan set data yang baru dibuat

Permintaan sampel:

```
/BAC/api/services/rest/bluesamservice/loadDataSetFromS3?region=us-
east-1&listcatFileS3Location=s3://bucket-name/listcat.json&datasetFileS3Location=s3://
bucket-name/dataset.DAT
```

Mengekspor kumpulan data ke bucket Amazon S3

Mengekspor kumpulan data ke bucket Amazon S3 yang ditentukan.

- Metode yang didukung: GET
- Jalan: `/api/services/rest/bluesamservice/exportDataSetToS3`
- Pendapat:

  S3Lokasi

  (wajib, string): lokasi Amazon S3 untuk mengekspor kumpulan data ke.

  DatasetName

  (required, string): nama kumpulan data yang akan diekspor.

  region

  (wajib, string): Wilayah AWS bucket Amazon S3.
- Mengembalikan kumpulan data yang diekspor

Permintaan sampel:

```
/BAC/api/services/rest/bluesamservice/exportDataSetToS3?region=eu-
west-1&s3Location=s3://bucket-name/dump&datasetName=dataset
```

Hapus kumpulan data

Menghapus semua catatan dari kumpulan data. Hal ini membutuhkan otentikasi.

- Metode yang didukung: POST
- Jalan: `/api/services/rest/bluesamservice/clearDataSet`
- Pendapat:

name

(required, string): nama kumpulan data untuk dihapus.

- Mengembalikan status permintaan.

Hapus kumpulan data

Menghapus definiton dan catatan kumpulan data. Hal ini membutuhkan otentikasi.

- Metode yang didukung: POST
- Jalan: `/api/services/rest/bluesamservice/deleteDataSet`
- Pendapat:

  name

  (required, string): nama kumpulan data yang akan dihapus.
- Mengembalikan status permintaan dan kumpulan data yang dihapus.

Hitung catatan kumpulan data

Titik akhir ini mengembalikan jumlah catatan yang terkait dengan kumpulan data. Hal ini membutuhkan otentikasi.

- Metode yang didukung: POST
- Jalan: `/api/services/rest/bluesamservice/countRecords`
- Pendapat:

  name

  (required, string): nama kumpulan data.
- Pengembalian: jumlah catatan

Data massal menetapkan titik akhir terkait

Gunakan titik akhir berikut untuk membuat atau mengelola beberapa kumpulan data sekaligus.

Topik

- Ekspor set data

- Buat beberapa set data
- Daftar semua set data
- Daftar Langsung semua set data
- Hapus semua set data
- Dapatkan definisi kumpulan data dari file listcat
- Dapatkan definisi kumpulan data dari file cat daftar yang diunggah
- Muat listcat dari file json

Ekspor set data

- Metode yang didukung: GET
- Jalan: `/api/services/rest/bluesamservice/exportDataSet`
- Pendapat:

  name

  (required, string): nama kumpulan data yang akan dihapus.

  datasetOutputFile

  (opsional, string): jalur tempat menyimpan kumpulan data yang diekspor di server

  rdw

  (opsional, boolean): haruskah bidang RDW diekspor.
- mengembalikan status permintaan dan file yang berisi kumpulan data yang diekspor.

Buat beberapa set data

- Metode yang didukung: POST
- Jalan: `/api/services/rest/bluesamservice/createAllDataSets`
- Pendapat:
  - Daftar kumpulan data

    name

    (required, string): nama kumpulan data.

tipe

(wajib, string): tipe kumpulan data. Nilai yang mungkin adalah:`ESDS`,`KSDS`,`RRDS`.

RecordSize

(opsional, string): Ukuran maksimum setiap catatan kumpulan data.

FixedLength

(opsional, boolean): Menunjukkan jika panjang catatan diperbaiki.

Kompresi

(opsional, boolean): Menunjukkan apakah kumpulan data dikompresi.

CacheEnable

(opsional, boolean): Menunjukkan apakah caching diaktifkan untuk kumpulan data.

- Pengembalian: status permintaan dan kumpulan data yang baru dibuat.

Daftar semua set data

- Metode yang didukung: GET
- Jalan: `/api/services/rest/bluesamservice/listDataSet`
- Argumen: Tidak ada
- Pengembalian: status permintaan dan daftar kumpulan data.

Daftar Langsung semua set data

- Metode yang didukung: GET
- Jalan: `/api/services/rest/bluesamservice/directListDataSet`
- Argumen: Tidak ada
- Pengembalian: status permintaan dan daftar kumpulan data.

Hapus semua set data

- Metode yang didukung: POST
- Jalan: `/api/services/rest/bluesamservice/removeAll`

- Argumen: Tidak ada
- Pengembalian: boolean yang mewakili status permintaan.

Dapatkan definisi kumpulan data dari file listcat

- Metode yang didukung: POST
- Jalan: `/api/services/rest/bluesamservice/getDataSetsDefinitionFromListcat`
- Pendapat:

  paramFilePath

  (required, string): Jalur ke file listcat.
- Pengembalian: daftar set data

Dapatkan definisi kumpulan data dari file cat daftar yang diunggah

- Metode yang didukung: POST
- Jalan: `` `/api/services/rest/bluesamservice/getDataSetsDefinitionFromUploadedListcat` ``
- Argumen: Tidak ada
- Pengembalian: daftar set data

Muat listcat dari file json

- Metode yang didukung: GET
- Jalan: `/api/services/rest/bluesamservice/loadListcatFromJsonFile`
- Pendapat:

  filePath

  (required, string): Jalur ke file listcat.
- Pengembalian: daftar set data

Catatan

Gunakan titik akhir berikut untuk membuat atau mengelola catatan dalam kumpulan data.

Topik

Buat catatan

Endpoint ini memungkinkan untuk membuat catatan baru. Hal ini membutuhkan otentikasi.

- Metode yang didukung: POST
- Jalan: `/api/services/rest/crud/createRecord`
- Pendapat:

  set data

  (wajib, DataSet): objek kumpulan data

  mask

  (wajib, topeng): objek topeng.
- Mengembalikan status permintaan dan catatan yang dibuat.

Membaca kumpulan data

Endpoint ini memungkinkan untuk membaca kumpulan data.

- Metode yang didukung: GET
- Jalan: `/api/services/rest/crud/readDataSet`
- Pendapat:

  set data

  (required, DataSet): objek kumpulan data.
- Mengembalikan status permintaan dan set data dengan catatan.

Hapus catatan

Titik akhir ini memungkinkan untuk menghapus catatan dari kumpulan data. Hal ini membutuhkan otentikasi.

- Metode yang didukung: HAPUS
- Jalan: `/api/services/rest/crud/deleteRecord`
- Pendapat:

  set data

  (wajib, DataSet): objek kumpulan data

  catatan

  (wajib, Rekam): catatan yang akan dihapus
- Mengembalikan status penghapusan.

Perbarui catatan

Titik akhir ini memungkinkan untuk memperbarui catatan yang terkait dengan kumpulan data. Hal ini membutuhkan otentikasi.

- Metode yang didukung: POST
- Jalan: `/api/services/rest/crud/updateRecord`
- Pendapat:

  set data

  (wajib, DataSet): objek kumpulan data

  catatan

  (wajib, Rekam): catatan untuk memperbarui
- Mengembalikan status permintaan dan set data dengan catatan.

Menyimpan catatan

Titik akhir ini memungkinkan untuk menyimpan catatan ke kumpulan data dan menggunakan topeng. Hal ini membutuhkan otentikasi.

- Metode yang didukung: POST

- Jalan: `/api/services/rest/crud/saveRecord`
- Pendapat:

  set data

  (wajib, DataSet): objek kumpulan data

  catatan

  (wajib, Rekam): catatan untuk disimpan
- Mengembalikan status permintaan dan set data dengan catatan.

Masker

Gunakan titik akhir berikut untuk memuat atau menerapkan masker ke kumpulan data.

Topik

- Muatkan masker
- Oleskan masker
- Terapkan filter masker

Muatkan masker

Titik akhir ini memungkinkan untuk mengambil semua topeng yang terkait dengan kumpulan data tertentu. Hal ini membutuhkan otentikasi.

- Metode yang didukung: POST
- Jalan: `/api/services/rest/crud/loadMasks`
- Pendapat:

  set data

  (wajib, DataSet): objek kumpulan data
- Mengembalikan status permintaan dan daftar topeng.

Oleskan masker

Endpoint ini memungkinkan untuk menerapkan mask ke kumpulan data tertentu. Hal ini membutuhkan otentikasi.

- Metode yang didukung: POST
- Jalan: `/api/services/rest/crud/applyMask`
- Pendapat:

  set data

  (wajib, DataSet): objek kumpulan data

  mask

  (wajib, Mask): objek kumpulan data
- Mengembalikan status permintaan dan set data dengan mask diterapkan.

Terapkan filter masker

Endpoint ini memungkinkan untuk menerapkan mask dan filter ke kumpulan data tertentu. Hal ini membutuhkan otentikasi.

- Metode yang didukung: POST
- Jalan: `/api/services/rest/crud/applyMaskFilter`
- Pendapat:

  set data

  (wajib, DataSet): objek kumpulan data

  mask

  (wajib, Mask): objek kumpulan data
- Mengembalikan status permintaan dan kumpulan data dengan masker dan filter yang diterapkan.

Lainnya

Gunakan titik akhir berikut untuk mengelola cache untuk kumpulan data atau memeriksa karakteristik kumpulan data

Topik

- Periksa cache pemanasan
- Periksa cache diaktifkan
- Aktifkan cache

- Periksa cache Ram yang dialokasikan
- Periksa kegigihan
- Periksa tipe kumpulan data yang didukung
- Periksa kesehatan server

Periksa cache pemanasan

Memeriksa apakah cache pemanasan diaktifkan untuk kumpulan data tertentu. Hal ini membutuhkan otentikasi.

- Metode yang didukung: POST
- Jalan: `/api/services/rest/bluesamservice/warmupCache`
- Pendapat:

  name

  (required, string): nama kumpulan data.
- Pengembalian: true jika cache pemanasan diaktifkan dan false sebaliknya.

Periksa cache diaktifkan

Memeriksa apakah cache diaktifkan untuk kumpulan data tertentu. Hal ini membutuhkan otentikasi.

- Metode yang didukung: GET
- Jalan: `/api/services/rest/bluesamservice/isEnableCache`
- Argumen: Tidak ada
- Mengembalikan nilai true jika caching diaktifkan.

Aktifkan cache

- Metode yang didukung: GET
- Jalan: `/api/services/rest/bluesamservice/enableDisableCache/{enable}`
- Pendapat:

  memungkinkan

  (required, boolean): jika disetel ke true, itu akan mengaktifkan caching.

- Mengembalikan Tidak Ada

Periksa cache Ram yang dialokasikan

Endpoint ini memungkinkan untuk mengambil memori cache RAM yang dialokasikan. Hal ini membutuhkan otentikasi.

- Metode yang didukung: GET
- Jalan: `/api/services/rest/bluesamservice/allocatedRamCache`
- Argumen: Tidak ada
- Mengembalikan: ukuran memori sebagai string

Periksa kegigihan

- Metode yang didukung: GET
- Jalan: `/api/services/rest/bluesamservice/persistence`
- Argumen: Tidak ada
- Pengembalian: persistensi yang digunakan sebagai string

Periksa tipe kumpulan data yang didukung

- Jalan: `/api/services/rest/bluesamservice/getDataSetTypes`
- Argumen: Tidak ada
- Pengembalian: daftar tipe kumpulan data yang didukung sebagai daftar string.

Periksa kesehatan server

- Metode yang didukung: GET
- Jalan: `/api/services/rest/bluesamservice/serverIsUp`
- Argumen: Tidak ada
- Pengembalian: Tidak ada

Pengguna

Gunakan titik akhir berikut untuk mengelola interaksi pengguna.

Topik

Login

- Metode yang didukung: POST
- Jalan: `/api/services/security/servicelogin/login`
- Pendapat:

  nama pengguna

  (diperlukan, string)

  password

  (diperlukan, string)
- Mengembalikan nama pengguna dan peran pengguna yang masuk

Respons sampel

```
{"login":"some-user","roles":[{"id":0,"roleName":"ROLE_ADMIN"}]}
```

Periksa akun pengguna

- Metode yang didukung: POST
- Jalan: `/api/services/security/servicelogin/hasAccount`
- Argumen: Tidak ada
- Pengembalian: true jika pengguna sudah login

Masuk

- Metode yang didukung: POST
- Jalan: `/api/services/security/servicelogin/recorduser`

- Argumen: Tidak ada
- Pengembalian: true jika pengguna sudah login

Daftar semua pengguna

- Metode yang didukung: GET
- Jalan: `/api/services/security/servicelogin/listusers`
- Argumen: Tidak ada
- Pengembalian: daftar semua pengguna

Keluar

- Metode yang didukung: POST
- Jalan: `/api/services/security/servicelogout/logout`
- Argumen: Tidak ada
- Pengembalian: true jika pengguna telah berhasil keluar.

Konsol Aplikasi JICS

Komponen JICS adalah dukungan AWS Blu Age untuk modernisasi sumber daya CICS warisan. Aplikasi web Konsol Aplikasi JICS didedikasikan untuk mengelola sumber daya JICS. Titik akhir berikut memungkinkan untuk melakukan tugas administrasi tanpa harus berinteraksi dengan antarmuka pengguna JAC. Setiap kali titik akhir memerlukan otentikasi, permintaan harus menyertakan detail otentikasi (nama pengguna/kata sandi biasanya, seperti yang dipersyaratkan oleh Otentikasi Dasar). Titik akhir untuk aplikasi web Konsol Aplikasi JICS menggunakan jalur root. `/jac/`

Topik

- Manajemen Sumber Daya JICS
- Titik akhir manajemen Pengguna JAC

Manajemen Sumber Daya JICS

Semua titik akhir berikut terkait dengan manajemen sumber daya JICS, memungkinkan administrator JICS untuk menangani sumber daya setiap hari.

Topik

- Daftar DAFTAR JICS dan GRUP
- Daftar JICS GROUPS
- Daftar JICS GROUPS untuk DAFTAR yang diberikan
- LIST sumber daya JICS untuk GROUP tertentu
- LIST JICS sumber daya untuk GROUP tertentu (alternatif menggunakan nama)
- Mengedit GRUP yang dimiliki dari beberapa DAFTAR
- Hapus LIST
- Menghapus GRUP
- Hapus TRANSAKSI
- Hapus program
- Hapus FILE
- Menghapus TDQUEUE
- Hapus TSMODEL
- Buat LIST
- Buat GRUP
- Pertimbangan pembuatan SUMBER DAYA umum
- Buat TRANSAKSI
- Buat PROGRAM
- Buat FILE
- Buat TDQUEUE
- Buat TSMODEL
- Memperbarui DAFTAR
- Memperbarui GRUP
- Pertimbangan pembaruan SUMBER DAYA umum
- Memperbarui TRANSAKSI
- Perbarui PROGRAM
- Memperbarui FILE
- Memperbarui TDQUEUE

- Perbarui TSMODEL

Daftar DAFTAR JICS dan GRUP

LIST dan GROUPS adalah sumber daya kontainer utama yang memiliki dalam komponen JICS. Semua sumber daya JICS harus milik GRUP. Grup dapat menjadi milik LISTS, tetapi ini tidak wajib. LISTS bahkan mungkin tidak ada pada lingkungan JICS tertentu, tetapi sebagian besar waktu, LISTS ada untuk memberikan lapisan organisasi tambahan untuk sumber daya. Untuk informasi selengkapnya tentang organisasi sumber daya CICS, lihat sumber daya CICS.

- Metode yang didukung: GET
- Membutuhkan otentikasi
- Jalan: `/api/services/rest/jicsservice/listJicsListsAndGroups`
- Pengembalian: daftar JicsContainer objek serial, baik LISTS dan GROUPS, sebagai JSON.

Contoh respons:

```
[
    {
      "name": "Resources",
      "children": [
        {
          "jacType": "JACList",
          "name": "MURACHS",
          "isActive": true,
          "children": [
            {
              "jacType": "JACGroup",
              "name": "MURACHS",
              "isActive": true,
              "children": []
            }
          ]
        },
        {
          "jacType": "JACGroup",
          "name": "TEST",
          "isActive": true,
          "children": []
        }
```

```
      ],
      "isExpanded": true
    }
  ]
```

Daftar JICS GROUPS

- Metode yang didukung: GET
- Membutuhkan otentikasi
- Jalan: `/api/services/rest/jicsservice/listJicsGroups`
- Pengembalian: daftar JicsContainer objek serial (GROUPS) sebagai JSON. GRUP dikembalikan tanpa informasi LIST milik mereka.

Contoh respons:

```
[
    {
      "jacType": "JACGroup",
      "name": "MURACHS",
      "isActive": true,
      "children": []
    },
    {
      "jacType": "JACGroup",
      "name": "TEST",
      "isActive": true,
      "children": []
    }
  ]
```

Daftar JICS GROUPS untuk DAFTAR yang diberikan

- Metode yang didukung: POST
- Membutuhkan otentikasi
- Jalan: `/api/services/rest/jicsservice/listGroupsForList`
- Argumen: payload JSON, mewakili DAFTAR JICS yang GROUPS yang kami cari. Ini adalah serialisasi JSON dari sebuah `com.netfective.bluage.jac.entities.JACList` objek.

  Permintaan sampel:

```
{
    "jacType":"JACList",
    "name":"MURACHS",
    "isActive":true
  }
```

- Pengembalian: daftar JicsContainer objek serial (GROUPS) sebagai JSON, yang dilampirkan ke LIST yang diberikan. GRUP dikembalikan tanpa informasi LIST milik mereka.

  Contoh respons:

```
[
    {
      "jacType": "JACGroup",
      "name": "MURACHS",
      "isActive": true,
      "children": []
    }
  ]
```

LIST sumber daya JICS untuk GROUP tertentu

- Metode yang didukung: POST
- Membutuhkan otentikasi
- Jalan: `/api/services/rest/jicsservice/listResourcesForGroup`
- Argumen: muatan JSON, mewakili JICS GROUP yang sumber dayanya kami cari. Ini adalah serialisasi JSON dari sebuah `com.netfective.bluage.jac.entities.JACGroup` objek. Anda tidak perlu menentukan semua bidang untuk GROUP, tetapi namanya wajib.

  Permintaan sampel:

```
{
    "jacType":"JACGroup",
    "name":"MURACHS",
    "isActive":true
  }
```

- Pengembalian: daftar JicsResource objek serial, yang dimiliki oleh GROUP yang diberikan. Objek dikembalikan tanpa urutan tertentu dan dari berbagai jenis (PROGRAM, TRANSAKSI, FILE, dll...).

LIST JICS sumber daya untuk GROUP tertentu (alternatif menggunakan nama)

- Metode yang didukung: POST
- Membutuhkan otentikasi
- Jalan: `/api/services/rest/jicsservice/listResourcesForGroupName`
- Argumen: nama GROUP yang memiliki sumber daya yang kami cari.
- Pengembalian: daftar JicsResource objek serial, yang dimiliki oleh GROUP yang diberikan. Objek dikembalikan tanpa urutan tertentu dan dari berbagai jenis (PROGRAM, TRANSAKSI, FILE, dll...)

Mengedit GRUP yang dimiliki dari beberapa DAFTAR

- Metode yang didukung: POST
- Membutuhkan otentikasi
- Jalan: `/api/services/rest/jicsservice/editGroupsList`
- Argumen: representasi JSON dari kumpulan LIST dengan anak-anak GROUPS;

  Permintaan sampel:

```
[
  {
        "jacType": "JACList",
        "name": "MURACHS",
        "isActive": true,
        "children": [
          {
            "jacType": "JACGroup",
            "name": "MURACHS",
            "isActive": true,
            "children": []
          },
          {
            "jacType": "JACGroup",
            "name": "TEST",
            "isActive": true,
            "children": []
          }
        ]
  }
]
```

Sebelum penyuntingan ini, hanya grup bernama “MURACHS” yang termasuk dalam DAFTAR bernama “MURACHS”. Dengan pengeditan ini, kami “menambahkan” grup bernama “TEST” ke DAFTAR bernama “MURACHS”.

- Mengembalikan nilai boolean. Jika nilainya 'true', modifikasi LISTS telah dipertahankan dengan benar ke penyimpanan JICS yang mendasarinya.

Hapus LIST

- Metode yang didukung: POST
- Membutuhkan otentikasi
- Jalan: `/api/services/rest/jicsservice/deleteList`
- Argumen: payload JSON, mewakili DAFTAR JICS untuk dihapus. Ini adalah serialisasi JSON dari sebuah `com.netfective.bluage.jac.entities.JACList` objek.
- Mengembalikan nilai boolean. Jika nilainya 'true', penghapusan LIST telah dioperasikan dengan benar pada penyimpanan JICS yang mendasarinya.

Menghapus GRUP

- Metode yang didukung: POST
- Membutuhkan otentikasi
- Jalan: `/api/services/rest/jicsservice/deleteGroup`
- Argumen: payload JSON, mewakili JICS GROUP untuk dihapus. Ini adalah serialisasi JSON dari sebuah `com.netfective.bluage.jac.entities.JACGroup` objek.
- Mengembalikan nilai boolean. Jika nilainya 'true', penghapusan GROUP telah dioperasikan dengan benar pada penyimpanan JICS yang mendasarinya.

Hapus TRANSAKSI

- Metode yang didukung: POST
- Membutuhkan otentikasi
- Jalan: `/api/services/rest/jicsservice/deleteTransaction`
- Argumen: payload JSON, mewakili Transaksi JICS untuk dihapus. Ini adalah serialisasi JSON dari sebuah `com.netfective.bluage.jac.entities.JACTransaction` objek.

- Mengembalikan nilai boolean. Jika nilainya 'benar', penghapusan TRANSAKSI telah dioperasikan dengan benar pada penyimpanan JICS yang mendasarinya.

Hapus program

- Metode yang didukung: POST
- Membutuhkan otentikasi
- Jalan: `/api/services/rest/jicsservice/deleteProgram`
- Argumen: payload JSON, mewakili Program JICS untuk dihapus. Ini adalah serialisasi JSON dari sebuah `com.netfective.bluage.jac.entities.JACProgram` objek.
- Mengembalikan nilai boolean. Jika nilainya 'benar', penghapusan PROGRAM telah dioperasikan dengan benar pada penyimpanan JICS yang mendasarinya.

Hapus FILE

- Metode yang didukung: POST
- Membutuhkan otentikasi
- Jalan: `/api/services/rest/jicsservice/deleteFile`
- Argumen: payload JSON, mewakili File JICS untuk dihapus. Ini adalah serialisasi JSON dari sebuah `com.netfective.bluage.jac.entities.JACFile` objek.
- Mengembalikan nilai boolean. Jika nilainya 'true', penghapusan FILE telah dioperasikan dengan benar pada penyimpanan JICS yang mendasarinya.

Menghapus TDQUEUE

- Metode yang didukung: POST
- Membutuhkan otentikasi
- Jalan: `/api/services/rest/jicsservice/deleteTDQueue`
- Argumen: payload JSON, mewakili JICS TDQUEUE untuk dihapus. Ini adalah serialisasi JSON dari objek `com.netfective.bluage.jac.entities.jactdQueue`.
- Mengembalikan nilai boolean. Jika nilainya 'true', penghapusan TDQUEUE telah dioperasikan dengan benar pada penyimpanan JICS yang mendasarinya.

Hapus TSMODEL

- Metode yang didukung: POST
- Membutuhkan otentikasi
- Jalan: `/api/services/rest/jicsservice/deleteTSModel`
- Argumen: payload JSON, mewakili JICS TSMODEL untuk dihapus. Ini adalah serialisasi JSON dari objek `com.netfective.bluage.jac.entities.jactsModel`.
- Mengembalikan nilai boolean. Jika nilainya 'benar', penghapusan TSMODEL telah dioperasikan dengan benar pada penyimpanan JICS yang mendasarinya.

Buat LIST

- Metode yang didukung: POST
- Membutuhkan otentikasi
- Jalan: `/api/services/rest/jicsservice/createList`
- Argumen: payload JSON, mewakili DAFTAR JICS untuk membuat. Ini adalah serialisasi JSON dari objek `com.netfective.bluage.jac.entities.jaclist`.
- Mengembalikan nilai boolean. Jika nilainya 'true', LIST telah dibuat dengan benar di penyimpanan JICS yang mendasarinya.

> **Note**
>
> Daftar akan selalu dibuat kosong. Melampirkan GRUP ke DAFTAR akan membutuhkan operasi lain.

Buat GRUP

- Metode yang didukung: POST
- Membutuhkan otentikasi
- Jalan: `/api/services/rest/jicsservice/createGroup`
- Argumen: payload JSON, mewakili JICS GROUP untuk membuat. Ini adalah serialisasi JSON dari sebuah `com.netfective.bluage.jac.entities.JACGroup` objek.
- Mengembalikan nilai boolean. Jika nilainya 'true', GROUP telah dibuat dengan benar di penyimpanan JICS yang mendasarinya.

> **Note**
>
> Grup akan selalu dibuat kosong. Melampirkan RESOURCES ke GROUP akan membutuhkan operasi tambahan (membuat sumber daya akan secara otomatis melampirkannya ke GROUP tertentu.

Pertimbangan pembuatan SUMBER DAYA umum

Semua titik akhir berikut terkait dengan pembuatan JICS RESOURCES dan berbagi beberapa kendala umum: dalam payload permintaan yang akan dikirim ke titik akhir, bidang harus dinilai. `groupName`

Kendala kepemilikan GROUP:

Tidak ada sumber daya yang dapat dibuat tanpa dilampirkan ke grup yang ada, dan titik akhir menggunakan GroupName untuk mengambil grup tempat sumber daya ini akan dilampirkan. `groupName`Harus menunjuk ke nama GRUP yang ada. Pesan kesalahan dengan HTTP STATUS 400 akan dikirim jika `groupName` tidak menunjuk ke grup yang ada di penyimpanan dasar JICS.

Kendala unisitas dalam GROUP:

Sumber daya yang diberikan dengan nama tertentu harus unik dalam grup tertentu. Pemeriksaan unisitas akan dilakukan oleh setiap titik akhir pembuatan sumber daya. Jika payload yang diberikan tidak menghormati batasan unicity, titik akhir akan mengirimkan respons dengan HTTP STATUS 400 (BAD REQUEST) -- lihat contoh respons di bawah ini.

Contoh payload: kami mencoba membuat transaksi 'ARIT' di grup 'TEST', tetapi transaksi dengan nama itu sudah ada di GROUP ini.

```
{
    "jacType":"JACTransaction",
    "name":"ARIT",
    "groupName":"TEST",
    "isActive":true
  }
```

Kami menerima respons kesalahan berikut:

```
{
```

```
    "timestamp": 1686759054510,
    "status": 400,
    "error": "Bad Request",
    "path": "/jac/api/services/rest/jicsservice/createTransaction"
  }
```

Insepecting server log akan mengkonfirmasi asal masalah:

```
2023-06-14 18:10:54 default         TRACE - o.s.w.m.HandlerMethod
        - Arguments: [java.lang.IllegalArgumentException: Transaction already
 present in the group, org.springframework.security.web.header.HeaderWriterFilter
$HeaderWriterResponse@e34f6b8]
2023-06-14 18:10:54 default         ERROR - c.n.b.j.a.WebConfig                    -
 400
java.lang.IllegalArgumentException: Transaction already present in the group
 at
 com.netfective.bluage.jac.server.services.rest.impl.JicsServiceImpl.createElement(JicsServiceI
```

Buat TRANSAKSI

- Metode yang didukung: POST
- Membutuhkan otentikasi
- Jalan: `/api/services/rest/jicsservice/createTransaction`
- Argumen: payload JSON, mewakili TRANSAKSI JICS untuk membuat. Ini adalah serialisasi JSON dari sebuah `com.netfective.bluage.jac.entities.JACTransaction` objek.
- Mengembalikan nilai boolean. Jika nilainya 'benar', TRANSAKSI telah dibuat dengan benar di penyimpanan JICS yang mendasarinya.

Buat PROGRAM

- Metode yang didukung: POST
- Membutuhkan otentikasi
- Jalan: `/api/services/rest/jicsservice/createProgram`
- Argumen: payload JSON, mewakili PROGRAM JICS untuk membuat. Ini adalah serialisasi JSON dari sebuah `com.netfective.bluage.jac.entities.JACProgram` objek.

- Mengembalikan nilai boolean. Jika nilainya 'true', PROGRAM telah dibuat dengan benar di penyimpanan JICS yang mendasarinya.

Buat FILE

- Metode yang didukung: POST
- Membutuhkan otentikasi
- Jalan: `/api/services/rest/jicsservice/createFile`
- Argumen: payload JSON, mewakili JICS FILE untuk membuat. Ini adalah serialisasi JSON dari objek `com.netfective.bluage.jac.entities.jacfile`.
- Mengembalikan nilai boolean. Jika nilainya 'true', FILE telah dibuat dengan benar di penyimpanan JICS yang mendasarinya.

Buat TDQUEUE

- Metode yang didukung: POST
- Membutuhkan otentikasi
- Jalan: `/api/services/rest/jicsservice/createTDQueue`
- Argumen: payload JSON, mewakili JICS TDQUEUE untuk membuat. Ini adalah serialisasi JSON dari objek `com.netfective.bluage.jac.entities.jactdQueue`.
- Mengembalikan nilai boolean. Jika nilainya 'true', TDQUEUE telah dibuat dengan benar di penyimpanan JICS yang mendasarinya.

Buat TSMODEL

- Metode yang didukung: POST
- Membutuhkan otentikasi
- Jalan: `/api/services/rest/jicsservice/createTSModel`
- Argumen: muatan JSON, mewakili JICS TSMODEL untuk dibuat. Ini adalah serialisasi JSON dari sebuah `com.netfective.bluage.jac.entities.JACTSModel` objek.
- Mengembalikan nilai boolean. Jika nilainya 'true', TSMODEL telah dibuat dengan benar di penyimpanan JICS yang mendasarinya.

Memperbarui DAFTAR

- Metode yang didukung: POST
- Membutuhkan otentikasi
- Jalan: `/api/services/rest/jicsservice/updateList`
- Argumen: payload JSON, mewakili DAFTAR JICS untuk memperbarui. Ini adalah serialisasi JSON dari sebuah `com.netfective.bluage.jac.entities.JACList` objek. Tidak perlu menyediakan anak-anak dari LIST, mekanisme pembaruan LIST tidak akan memperhitungkan hal ini.
- Mengembalikan nilai boolean. Jika nilainya 'true', LIST telah diperbarui dengan benar di penyimpanan JICS yang mendasarinya.

Memperbarui bendera LIST 'isActive' akan menyebar ke semua elemen yang dimiliki LIST, yaitu, semua GRUP yang dimiliki oleh LIST dan semua SUMBER DAYA yang dimiliki oleh GRUP tersebut. Ini adalah cara mudah untuk menonaktifkan banyak sumber daya dengan satu operasi, melalui beberapa GRUP.

Memperbarui GRUP

- Metode yang didukung: POST
- Membutuhkan otentikasi
- Jalan: `/api/services/rest/jicsservice/updateGroup`
- Argumen: payload JSON, mewakili JICS GROUP untuk memperbarui. Ini adalah serialisasi JSON dari sebuah `com.netfective.bluage.jac.entities.JACGroup` objek. Tidak perlu mendukung anak-anak dari GROUP, mekanisme pembaruan GROUP tidak akan memperhitungkan hal ini.
- Mengembalikan nilai boolean. Jika nilainya 'true', GROUP telah diperbarui dengan benar di penyimpanan JICS yang mendasarinya.

> **Note**
>
> Memperbarui flag GROUP 'isActive' akan menyebar ke semua elemen yang dimiliki GROUP, yaitu, semua SUMBER DAYA yang dimiliki oleh GROUP. Ini adalah cara mudah untuk menonaktifkan banyak sumber daya dengan satu operasi dalam GROUP tertentu.

Pertimbangan pembaruan SUMBER DAYA umum

Semua titik akhir berikut adalah tentang memperbarui JICS RESOURCES. Dengan menggunakan `groupName` bidang ini, Anda dapat mengubah GROUP yang memiliki sumber daya JICS apa pun, asalkan nilai bidang menunjuk ke GRUP yang keluar di penyimpanan JICS yang mendasarinya (jika tidak, Anda akan mendapatkan respons PERMINTAAN BURUK (HTTP STATUS 400) dari titik akhir).

Memperbarui TRANSAKSI

- Metode yang didukung: POST
- Membutuhkan otentikasi
- Jalan: `/api/services/rest/jicsservice/updateTransaction`
- Argumen: payload JSON, mewakili TRANSAKSI JICS untuk memperbarui. Ini adalah serialisasi JSON dari sebuah `com.netfective.bluage.jac.entities.JACTransaction` objek.
- Mengembalikan nilai boolean. Jika nilainya 'benar', TRANSAKSI telah diperbarui dengan benar di penyimpanan JICS yang mendasarinya.

Perbarui PROGRAM

- Metode yang didukung: POST
- Membutuhkan otentikasi
- Jalan: `/api/services/rest/jicsservice/updateProgram`
- Argumen: payload JSON, mewakili PROGRAM JICS untuk memperbarui. Ini adalah serialisasi JSON dari sebuah `com.netfective.bluage.jac.entities.JACProgram` objek.
- Mengembalikan nilai boolean. Jika nilainya 'true', PROGRAM telah diperbarui dengan benar di penyimpanan JICS yang mendasarinya.

Memperbarui FILE

- Metode yang didukung: POST
- Membutuhkan otentikasi
- Jalan: `/api/services/rest/jicsservice/updateFile`
- Argumen: payload JSON, mewakili JICS FILE untuk memperbarui. Ini adalah serialisasi JSON dari sebuah `com.netfective.bluage.jac.entities.JACFile` objek.

- Mengembalikan nilai boolean. Jika nilainya 'true', FILE telah diperbarui dengan benar di penyimpanan JICS yang mendasarinya.

Memperbarui TDQUEUE

- Metode yang didukung: POST
- Membutuhkan otentikasi
- Jalan: `/api/services/rest/jicsservice/updateTDQueue`
- Argumen: payload JSON, mewakili JICS TDQUEUE untuk diperbarui. Ini adalah serialisasi JSON dari sebuah `com.netfective.bluage.jac.entities.JACTDQueue` objek.
- Mengembalikan nilai boolean. Jika nilainya 'true', tdQueue telah diperbarui dengan benar di penyimpanan JICS yang mendasarinya.

Perbarui TSMODEL

- Metode yang didukung: POST
- Membutuhkan otentikasi
- Jalan: `/api/services/rest/jicsservice/updateTSModel`
- Argumen: payload JSON, mewakili JICS TSMODEL untuk diperbarui. Ini adalah serialisasi JSON dari sebuah `com.netfective.bluage.jac.entities.JACTSModel` objek.
- Mengembalikan nilai boolean. Jika nilainya 'true', TSMODEL telah diperbarui dengan benar di penyimpanan JICS yang mendasarinya.

Titik akhir manajemen Pengguna JAC

Topik

- Menghapus pengguna
- Mencatat pengguna
- Menguji jika setidaknya ada pengguna dalam sistem
- Merekam pengguna baru
- Memerinci pengguna
- Memerinci pengguna
- Keluar dari pengguna saat ini

- Status kesehatan server Jics

Menghapus pengguna

- Metode yang didukung: POST
- Memerlukan otentikasi dan hak ADMIN
- Jalan: `/api/services/security/servicelogin/deleteuser`
- Argumen: serialisasi JSON `com.netfective.bluage.jac.entities.SignOn` objek, mewakili pengguna yang akan dihapus dari penyimpanan.
- Mengembalikan nilai boolean. Jika nilainya 'true', pengguna telah dihapus dengan benar dari sistem JICS.

> **Note**
>
> Tindakan ini tidak dapat dibatalkan, pengguna yang dihapus tidak akan dapat terhubung ke aplikasi JAC lagi. Berhati-hatilah saat menggunakannya.

Mencatat pengguna

- Metode yang didukung: POST
- Memerlukan otentikasi dan hak ADMIN
- Jalan: `/api/services/security/servicelogin/login`
- Mengembalikan serialisasi JSON `com.netfective.bluage.jac.entities.SignOn` objek, mewakili pengguna yang kredensialnya disediakan dalam permintaan saat ini. Kata sandi disembunyikan dari tampilan di objek yang dikembalikan. Peran yang diberikan kepada yang digunakan sedang terdaftar.

Contoh respons:

```
{
    "login": "sadmin",
    "password": null,
    "roles": [
      {
        "id": 0,
```

```
        "roleName": "ROLE_SUPER_ADMIN"
      }
    ]
  }
```

Menguji jika setidaknya ada pengguna dalam sistem

- Metode yang didukung: GET
- Jalan: `/api/services/security/servicelogin/hasAccount`
- Mengembalikan nilai boolean, yang nilainya benar jika setidaknya pengguna - selain pengguna admin super default - telah dibuat dalam sistem JICS.

Merekam pengguna baru

- Metode yang didukung: POST
- Memerlukan otentikasi dan hak ADMIN
- Jalan: `/api/services/security/servicelogin/recorduser`
- Argumen: serialisasi JSON `com.netfective.bluage.jac.entities.Sign0n` objek, mewakili pengguna yang akan ditambahkan ke penyimpanan. Peran untuk pengguna harus ditentukan, jika tidak, pengguna mungkin tidak dapat menggunakan fasilitas JAC dan titik akhir.

Permintaan sampel:

```
{
    "login": "simpleuser",
    "password": "simpleuser",
    "roles": [
      {
        "id": 2,
        "roleName": "ROLE_USER"
      }
    ]
  }
```

Hanya peran berikut yang dapat digunakan saat merekam pengguna baru:

- ROLE_ADMIN: dapat mengelola sumber daya dan pengguna JICS.
- ROLE_USER: dapat mengelola sumber daya JICS tetapi bukan pengguna.

Memerinci pengguna

- Metode yang didukung: GET
- Memerlukan otentikasi dan hak ADMIN
- Jalan: `/api/services/security/servicelogin/listusers`
- Mengembalikan daftar`com.netfective.bluage.jac.entities.SignOn`, serial sebagai JSON.

Memerinci pengguna

- Metode yang didukung: GET
- Memerlukan otentikasi dan hak ADMIN
- Jalan: `/api/services/security/servicelogin/listusers`
- Mengembalikan daftar`com.netfective.bluage.jac.entities.SignOn`, serial sebagai JSON.

Keluar dari pengguna saat ini

- Metode yang didukung: GET
- Jalan: `/api/services/security/servicelogout/logout`
- Mengembalikan pesan JSON berikut: “{" success” :true}” jika logout dari pengguna saat ini berhasil (sesi HTTP terkait akan dibatalkan).

Status kesehatan server Jics

- Metode yang didukung: GET
- Jalan: `/api/services/rest/jicsserver/serverIsUp`
- Menunjukkan bahwa server JICS aktif dan berjalan, jika Anda mendapatkan respons yang memiliki HTTP STATUS 200.

Struktur Data

Bagian ini menjelaskan rincian dari berbagai struktur data.

Topik

Struktur pesan Detail Eksekusi Job

Setiap detail pelaksanaan pekerjaan akan memiliki bidang berikut:

ScriptID

pengenal dari skrip yang disebut.

pemanggil

I.P. alamat penelepon.

pengenal

pengidentifikasi eksekusi pekerjaan yang unik.

startTime

tanggal dan waktu pelaksanaan pekerjaan dimulai.

endTime

tanggal dan waktu di mana eksekusi pekerjaan berakhir.

status

status untuk pelaksanaan pekerjaan. Satu nilai yang mungkin di antara:

- `DONE`: eksekusi pekerjaan berakhir secara normal.
- `TRIGGERED`: eksekusi pekerjaan dipicu tetapi belum diluncurkan.
- `RUNNING`: eksekusi pekerjaan sedang berjalan.
- `KILLED`: eksekusi pekerjaan telah terbunuh.
- `FAILED`: eksekusi pekerjaan telah gagal.

Hasil Eksekusi

pesan untuk meringkas hasil eksekusi pekerjaan. Pesan ini dapat berupa pesan sederhana jika eksekusi pekerjaan belum selesai atau struktur JSON dengan bidang berikut:

- ExitCode: kode keluar numerik; nilai negatif menunjukkan situasi kegagalan.
- program: program terbaru yang diluncurkan oleh pekerjaan.
- status: satu nilai yang mungkin di antara:
  - `Error`: ketika ExitCode = -1; ini sesuai dengan kesalahan (teknis) yang terjadi selama eksekusi pekerjaan.
  - `Failed`: ketika exitcode = -2; Ini sesuai dengan kegagalan yang terjadi selama eksekusi program layanan (seperti situasi ABEND).
  - `Succeeded`: ketika ExitCode >= 0;
- StepName: nama langkah terbaru yang dijalankan dalam pekerjaan.

ExecutionMode

baik SYNCHRONOUS atau ASYNCHRONOUS, tergantung pada cara pekerjaan diluncurkan.

Contoh output:

```
{
    "scriptId": "INTCALC",
    "caller": "127.0.0.1",
    "identifier": "97d410be-efa7-4bd3-b7b9-d080e5769771",
    "startTime": "06-09-2023 11:42:41",
    "endTime": "06-09-2023 11:42:42",
    "status": "DONE",
    "executionResult": "{ \"exitCode\": -1, \"stepName\": \"STEP15\", \"program\":
 \"CBACT04C\", \"status\": \"Error\" }",
    "executionMode": "ASYNCHRONOUS"
  }
```

Struktur hasil peluncuran transaksi

Struktur mungkin berisi bidang-bidang berikut:

Hasil

string yang mewakili hasil eksekusi transaksi. Kemungkinan nilainy adalah:

- `Success`: eksekusi transaksi berjalan sampai akhir dengan benar.
- `Failure`: eksekusi transaksi gagal berakhir dengan benar, beberapa masalah ditemui.

commarea

string yang mewakili nilai akhir COMMAREA, sebagai array byte yang dikodekan byte64. Mungkin string kosong.

ContainerRecord

(opsional) string yang mewakili konten rekaman CONTAINER sebagai array byte yang dikodekan byte64.

Deskripsi Server

Mungkin berisi informasi tentang server yang melayani permintaan (untuk tujuan debugging). Mungkin string kosong.

AbendCode

(opsional) jika program direferensikan oleh transaksi yang diluncurkan abended, nilai kode abend akan dikembalikan sebagai string di bidang ini.

Sampel tanggapan:

Berhasil

```
{
    "outCome": "Success",
    "commarea": "",
    "serverDescription": ""
  }
```

Kegagalan

```
{
    "outCome": "Failure",
    "commarea": "",
    "serverDescription": "",
    "abendCode": "AEIA"
  }
```

Struktur hasil catatan peluncuran transaksi

Struktur mungkin berisi bidang-bidang berikut:

RecordContent

string yang mewakili konten rekaman COMMAREA sebagai array byte yang dikodekan byte64.

ContainerRecord

string yang mewakili konten rekaman CONTAINER sebagai array byte yang dikodekan byte64.

Deskripsi Server

Mungkin berisi informasi tentang server yang melayani permintaan (untuk tujuan debugging). Mungkin string kosong.

Sampel tanggapan:

Berhasil

```
{
    "recordContent": "",
    "serverDescription": ""
}
```

Kemungkinan status pekerjaan dalam antrian

Pada antrian, pekerjaan dapat memiliki status berikut:

AKTIF

Pekerjaan saat ini sedang dijalankan di antrian.

EKSEKUSI_TUNGGU

Pekerjaan sedang menunggu utas tersedia.

DIJADWALKAN

Pekerjaan dijadwalkan untuk dieksekusi pada tanggal dan waktu tertentu.

TAHAN

Job sedang menunggu untuk dibebaskan sebelum dijalankan.

DISELESAIKAN

Job telah berhasil dieksekusi.

Failed

Job Execution telah gagal.

TIDAK DIKETAHUI

Status tidak diketahui.

Kirim masukan pekerjaan

Masukan tugas submit adalah serialisasi JSON dari sebuah `com.netfective.bluage.gapwalk.rt.jobqueue.SubmitJobMessage` objek. Masukan sampel di bawah ini menunjukkan semua bidang untuk kacang seperti itu.

Masukan sampel:

```
{
    "messageQueueName":null,
    "scheduleDate":null,
    "scheduleTime":null,
    "programName":"PTA0044",
    "programParams":
     {"wmind":"B"},
    "localDataAreaValue":"",
    "userName":"USER1",
    "jobName":"PTA0044",
    "jobNumber":9,
    "jobPriority":5,
    "executionDate":"20181231",
    "jobQueue":"queue1",
    "jobOnHold":false
}
```

JobNumber

jika jobnumber adalah 0, nomor pekerjaan akan dibuat secara otomatis menggunakan nomor berikutnya dalam urutan nomor pekerjaan. Nilai itu harus diatur ke 0 (kecuali untuk tujuan pengujian).

JobPriority

Prioritas pekerjaan default di AS400 adalah 5. Rentang yang valid adalah 0-9, 0 menjadi prioritas tertinggi.

jobOnHold

Jika pekerjaan ditunda, itu tidak akan langsung dieksekusi tetapi hanya ketika seseorang "melepaskannya". Pekerjaan dapat dirilis menggunakan REST API (/release atau /release-all).

ScheduleDate dan ScheduleTime

Jika nilai-nilai ini tidak nol, pekerjaan akan dieksekusi pada tanggal dan waktu yang ditentukan.

Tanggal

dapat disediakan dengan format mmddyy atau ddmMyyyy (ukuran input akan menentukan format apa yang digunakan)

Waktu

dapat disediakan dengan format HHmm atau HHMMSS (ukuran input akan menentukan format apa yang digunakan)

ProgramParams

ini akan diteruskan ke program sebagai peta.

Daftar respon pekerjaan terjadwal

Ini adalah struktur titik akhir antrian pekerjaan daftar-pekerjaan. Harap dicatat bahwa pesan kirim pekerjaan yang digunakan untuk mengirimkan pekerjaan itu adalah bagian dari tanggapan. Ini dapat digunakan untuk tujuan pelacakan atau pengujian/pengiriman ulang. Ketika pekerjaan selesai, tanggal mulai dan tanggal akhir juga akan diisi.


```
[
    {
      "qrtzJobGroup": "PTA0044-PTA0044",
      "qrtzJobName": "PTA0044-168156109957800",
      "status": "HOLD",
      "qrtzDelay": 0,
      "startDate": null,
      "endDate": null,
      "jobName": "PTA0044",
      "userName": "USER1",
```

```
      "jobNumber": 9,
      "jobPriority": 5,
      "jobQueue": "queue1",
      "message": {
        "messageQueueName": null,
        "scheduleDate": null,
        "scheduleTime": null,
        "programName": "PTA0044",
        "programParams": {
          "wmind": "B"
        },
        "localDataAreaValue": "",
        "userName": "USER1",
        "jobName": "PTA0044",
        "jobNumber": 9,
        "jobPriority": 5,
        "executionDate": "20181231",
        "jobQueue": "queue1",
        "jobOnHold": true
      }
    },
    {
      "qrtzJobGroup": "PTA0044-PTA0044",
      "qrtzJobName": "PTA0044-166196954877600",
      "status": "COMPLETED",
      "qrtzDelay": 0,
      "startDate": "2022-10-13T22:48:34.025+00:00",
      "endDate": "2022-10-13T22:52:54.475+00:00",
      "jobName": "PTA0044",
      "userName": "USER1",
      "jobNumber": 9,
      "jobPriority": 5,
      "jobQueue": "queue1",
      "message": {
        "messageQueueName": null,
        "scheduleDate": null,
        "scheduleTime": null,
        "programName": "PTA0044",
        "programParams": {
          "wmind": "B"
        },
        "localDataAreaValue": "",
        "userName": "USER1",
        "jobName": "PTA0044",
```

```
        "jobNumber": 9,
        "jobPriority": 5,
        "executionDate": "20181231",
        "jobQueue": "queue1",
        "jobOnHold": true
      }
    }
  ]
```

Daftar respons pekerjaan “ditahan”

Strukturnya mirip dengan yang sebelumnya, kecuali bahwa semua pekerjaan yang dikembalikan akan “ditahan”.

```
[
    {
      "qrtzJobGroup": "PTA0044-PTA0044",
      "qrtzJobName": "PTA0044-168156109957800",
      "status": "HOLD",
      "qrtzDelay": 0,
      "startDate": null,
      "endDate": null,
      "jobName": "PTA0044",
      "userName": "USER1",
      "jobNumber": 9,
      "jobPriority": 5,
      "jobQueue": "queue1",
      "message": {
        "messageQueueName": null,
        "scheduleDate": null,
        "scheduleTime": null,
        "programName": "PTA0044",
        "programParams": {
          "wmind": "B"
        },
        "localDataAreaValue": "",
        "userName": "USER1",
        "jobName": "PTA0044",
        "jobNumber": 9,
        "jobPriority": 5,
        "executionDate": "20181231",
        "jobQueue": "queue1",
```

```
        "jobOnHold": true
      }
    }
  ]
```

AWS Pengaturan Blu Age Runtime (tidak dikelola)

Bagian ini menjelaskan langkah-langkah untuk mengatur AWS Blu Age Runtime (tidak dikelola) pada infrastruktur Anda. AWS

Topik

- AWS Prasyarat Runtime Blu Age
- AWS Orientasi Blu Age Runtime
- Persyaratan penyiapan infrastruktur untuk AWS Blu Age Runtime (tidak dikelola)
- AWS Penerapan Blu Age Runtime di Amazon ECS dikelola oleh AWS Fargate
- AWS Penerapan Blu Age Runtime di Amazon EC2

AWS Prasyarat Runtime Blu Age

AWS Blu Age Runtime (non-managed) tersedia dalam beberapa versi rilis. Jika Anda memiliki proyek modernisasi yang sedang berlangsung, Anda mungkin memerlukan versi tambahan dari runtime untuk tujuan implementasi dan pengujian. Untuk menentukan kebutuhan Anda, hubungi manajer pengiriman AWS Blu Age Anda.

Sebelum Anda memulai proses orientasi AWS Blu Age Runtime (non-managed), lakukan hal berikut:

- Pastikan Anda memiliki AWS akun.
- Pastikan Anda memiliki aplikasi modern yang difaktorkan ulang dengan Blu Age. AWS
- Pilih AWS Wilayah dan komputasi (Amazon EC2 atau Amazon ECS aktif). AWS Fargate
- Pilih versi AWS Blu Age Runtime yang ingin Anda gunakan.
- Tinjau the section called “Persyaratan pengaturan infrastruktur” dan validasi komponen tambahan yang diperlukan untuk menjalankan AWS Blu Age Runtime (tidak dikelola).

> **Note**
>
> Jika Anda ingin menguji fitur AWS Blu Age Runtime (non-managed), Anda dapat menggunakan aplikasi demo`Planets Demo`, yang dapat Anda unduh dari -v1.zip. PlanetsDemo

AWS Orientasi Blu Age Runtime

Untuk memulai, buat AWS Support kasus untuk meminta orientasi untuk mengakses AWS Blu Age Runtime. Sertakan dalam permintaan Akun AWS ID Anda, AWS Wilayah yang ingin Anda gunakan, dan pilihan komputasi dan versi runtime. Jika Anda tidak yakin versi mana yang Anda butuhkan, hubungi manajer pengiriman AWS Blu Age Anda.

AWS Blu Age Runtime (tidak dikelola) di Amazon EC2

Kami menyimpan artefak AWS Blu Age Runtime (tidak dikelola) di bucket Amazon S3 yang berbeda berdasarkan Wilayah dan dengan pilihan komputasi. Untuk mengakses bucket Wilayah AWS untuk AWS Blu Age Runtime (tidak dikelola) di Amazon EC2, gunakan nama yang tercantum dalam tabel berikut.

Wilayah AWS	Pelepasan ember	Bucket membangun
AS Timur (Ohio)	aws-bluage-runtime-artifacts-055777665268-us-timur-2	aws-bluage-runtime-artifacts-dev-055777665268-us-timur-2
AS Timur (Virginia Utara)	aws-bluage-runtime-artifacts-139023371234-kita-timur-1	aws-bluage-runtime-artifacts-dev-139023371234-kita-timur-1
AS Barat (California Utara)	aws-bluage-runtime-artifacts-788454048782-kita-barat-1	aws-bluage-runtime-artifacts-dev-788454048782-kita-barat-1
AS Barat (Oregon)	aws-bluage-runtime-artifacts-836771190483-kita-barat-2	aws-bluage-runtime-artifacts-dev-836771190483-kita-barat-2

Wilayah AWS	Pelepasan ember	Bucket membangun
Eropa (Irlandia)	aws-bluage-runtime-artifacts-925278190477-eu-barat-1	aws-bluage-runtime-artifacts-dev-925278190477-eu-barat-1
Eropa (Paris)	aws-bluage-runtime-artifacts-673009995881-eu-barat-3	aws-bluage-runtime-artifacts-dev-673009995881-eu-barat-3
Eropa (Frankfurt)	aws-bluage-runtime-artifacts-485196800481-eu-pusat-1	aws-bluage-runtime-artifacts-dev-485196800481-eu-pusat-1
Amerika Selatan (Sao Paulo)	aws-bluage-runtime-artifacts-737536804457-sa-timur-1	aws-bluage-runtime-artifacts-dev-737536804457-sa-timur-1
Asia Pasifik (Tokyo)	aws-bluage-runtime-artifacts-445578176276-ap-timur laut-1	aws-bluage-runtime-artifacts-dev-445578176276-ap-timur laut-1
Asia Pasifik (Sydney)	aws-bluage-runtime-artifacts-726160321909-ap-tenggara-2	aws-bluage-runtime-artifacts-dev-726160321909-ap-tenggara-2

AWS Blu Age Runtime (tidak dikelola) di Amazon ECS yang dikelola oleh Fargate

Kami menyimpan artefak AWS Blu Age Runtime (tidak dikelola) di bucket Amazon S3 yang berbeda berdasarkan Wilayah dan dengan pilihan komputasi. Untuk mengakses bucket Wilayah AWS untuk AWS Blu Age Runtime (tidak dikelola) di Amazon ECS yang dikelola oleh Fargate, gunakan nama yang tercantum dalam tabel berikut.

Wilayah AWS	Pelepasan ember	Bucket membangun
AS Timur (Ohio)	aws-bluage-runtime-fargate-rel-483416914331-us-timur-2	aws-bluage-runtime-fargate-dev-483416914331-kita-timur-2

Wilayah AWS	Pelepasan ember	Bucket membangun
AS Timur (Virginia Utara)	aws-bluage-runtime-fargate-rel-308472162679-kita-timur-1	aws-bluage-runtime-fargate-dev-308472162679-kita-timur-1
AS Barat (California Utara)	aws-bluage-runtime-fargate-rel-343763094578-kita-barat-1	aws-bluage-runtime-fargate-dev-343763094578-kita-barat-1
AS Barat (Oregon)	aws-bluage-runtime-fargate-rel-688933007849-kita-barat-2	aws-bluage-runtime-fargate-dev-688933007849-kita-barat-2
Eropa (Irlandia)	aws-bluage-runtime-fargate-rel-140138033705-eu-barat-1	aws-bluage-runtime-fargate-dev-140138033705-eu-barat-1
Eropa (Paris)	aws-bluage-runtime-fargate-rel-339712948211-eu-barat-3	aws-bluage-runtime-fargate-dev-339712948211-eu-barat-3
Eropa (Frankfurt)	aws-bluage-runtime-fargate-rel-339712918892-eu-pusat-1	aws-bluage-runtime-fargate-dev-339712918892-eu-pusat-1
Amerika Selatan (Sao Paulo)	aws-bluage-runtime-fargate-rel-767397998881-sa-timur-1	aws-bluage-runtime-fargate-dev-767397998881-sa-timur-1
Asia Pasifik (Tokyo)	aws-bluage-runtime-fargate-rel-891377400849-ap-timur laut-1	aws-bluage-runtime-fargate-dev-891377400849-ap-timur laut-1
Asia Pasifik (Sydney)	aws-bluage-runtime-fargate-rel-533267435478-ap-tenggara	aws-bluage-runtime-fargate-dev-533267435478-ap-tenggara-2

Menggunakan baris perintah untuk membuat daftar isi ember

Setelah Anda onboard, Anda dapat membuat daftar isi bucket dengan menjalankan perintah berikut di terminal.

```
aws s3 ls bucket-name
```

Ganti `bucket-name` dengan nama ember untuk Anda Wilayah AWS dari tabel sebelumnya.

Perintah ini mengembalikan daftar folder yang sesuai dengan versi yang berbeda dari runtime AWS Blu Age Runtime (non-managed), seperti

```
PRE 3.3.0.1/
PRE 3.3.0.2/
```

Kami menyarankan Anda menggunakan versi terbaru yang tersedia. Jika itu tidak memungkinkan, gunakan versi runtime yang divalidasi selama fase refactoring aplikasi. Untuk membuat daftar kerangka kerja yang tersedia untuk versi tertentu, jalankan perintah berikut:

```
aws s3 ls s3://bucket-name/version/Framework/
```

Ganti `bucket-name` dengan nama ember untuk Anda Wilayah AWS dan `version` dengan versi yang Anda inginkan. Berikut sebuah contoh.

```
aws s3 ls s3://aws-bluage-runtime-artifacts-139023371234-us-east-1/3.4.0/
Framework/
```

Perintah akan mengembalikan daftar kerangka kerja, seperti:

```
2023-06-05 10:26:52   97960225 aws-bluage-runtime-3.4.0.tar.gz
2023-06-05 10:27:12         45 aws-bluage-runtime-3.4.0.tar.gz.checksumSHA256
2023-06-05 10:27:14  138497123 aws-bluage-webapps-3.4.0.tar.gz
2023-06-05 10:27:44         45 aws-bluage-webapps-3.4.0.tar.gz.checksumSHA256
```

Unduh kerangka kerja

Anda dapat mengunduh kerangka kerja misalnya untuk memutakhirkan versi runtime kecepatan pada instans Amazon EC2 yang ada.

```
aws s3 cp s3://bucket-name/version/Framework/ folder-of-your-choice --
recursive
```

Di mana:

folder-of-your-choice

jalur folder tempat Anda ingin mengunduh kerangka kerja.

Sebagai contoh: `aws s3 cp s3://aws-bluage-runtime-artifacts-139023371234-us-east-1/3.4.0/Framework/ . --recursive`

Perintah ini menghasilkan output berikut ini:

```
download: s3://aws-bluage-runtime-artifacts-139023371234-us-east-1/3.4.0/
Framework/aws-bluage-runtime-3.4.0.tar.gz.checksumSHA256 to ./aws-bluage-
runtime-3.4.0.tar.gz.checksumSHA256
download: s3://aws-bluage-runtime-artifacts-139023371234-us-east-1/3.4.0/
Framework/aws-bluage-webapps-3.4.0.tar.gz.checksumSHA256 to ./aws-bluage-
webapps-3.4.0.tar.gz.checksumSHA256
download: s3://aws-bluage-runtime-artifacts-139023371234-us-east-1/3.4.0/Framework/aws-
bluage-webapps-3.4.0.tar.gz to ./aws-bluage-webapps-3.4.0.tar.gz
download: s3://aws-bluage-runtime-artifacts-139023371234-us-east-1/3.4.0/Framework/aws-
bluage-runtime-3.4.0.tar.gz to ./aws-bluage-runtime-3.4.0.tar.gz
```

Anda dapat membuat daftar file kerangka kerja sebagai berikut:

`ls -l`

Perintah ini menghasilkan output berikut ini:

```
total 230928
-rw-rw-r-- 1 cloudshell-user cloudshell-user  97960225 Jun  5 10:26 aws-bluage-
runtime-3.4.0.tar.gz
-rw-rw-r-- 1 cloudshell-user cloudshell-user        45 Jun  5 10:27 aws-bluage-
runtime-3.4.0.tar.gz.checksumSHA256
-rw-rw-r-- 1 cloudshell-user cloudshell-user 138497123 Jun  5 10:27 aws-bluage-
webapps-3.4.0.tar.gz
-rw-rw-r-- 1 cloudshell-user cloudshell-user        45 Jun  5 10:27 aws-bluage-
webapps-3.4.0.tar.gz.checksumSHA256
```

Persyaratan penyiapan infrastruktur untuk AWS Blu Age Runtime (tidak dikelola)

Topik ini menjelaskan konfigurasi infrastruktur minimum yang diperlukan untuk menjalankan AWS Blu Age Runtime (tidak dikelola). Prosedur berikut menjelaskan cara mengatur AWS Blu Age Runtime (tidak dikelola) pada komputasi pilihan Anda untuk menerapkan aplikasi modern pada Blu Age Runtime. AWS Sumber daya yang Anda buat harus berada di VPC Amazon yang memiliki subnet yang didedikasikan untuk domain aplikasi Anda.

Membuat grup keamanan

1. Buka konsol Amazon VPC di https://console.aws.amazon.com/vpc/.
2. Di panel navigasi kiri, di bawah Keamanan, pilih Grup keamanan.
3. Di panel tengah, pilih Buat grup keamanan.
4. Di bidang Nama grup keamanan, masukkan**`M2BluagePrivateLink-SG`**.
5. Di bagian Aturan masuk, pilih Tambahkan aturan.
6. Untuk Type, pilih HTTPS.
7. Untuk Sumber, masukkan CIDR VPC Anda.
8. Di bagian Aturan keluar, pilih Tambahkan aturan.
9. Untuk Type, pilih HTTPS.
10. Untuk Tujuan, masukkan**`0.0.0.0/0`**.
11. Pilih Buat grup keamanan.

Buat titik akhir VPC Amazon

1. Buka konsol Amazon VPC di https://console.aws.amazon.com/vpc/.
2. Di panel navigasi kiri, di bawah Virtual private cloud, pilih Endpoints.
3. Di panel tengah, pilih Buat titik akhir.
4. Di bagian Layanan, masukkan **`SQS`** di bidang pencarian, lalu pilih layanan Amazon SQS yang sesuai dengan Wilayah Anda.
5. Di bagian VPC, pilih VPC Amazon yang Anda buat di langkah sebelumnya.
6. Di bagian Subnet, pilih subnet yang Anda buat untuk domain aplikasi Anda.
7. Di bagian Grup keamanan, pilih M2 BluagePrivateLink -SG.

8. Pilih Buat titik akhir.

Buat kebijakan IAM

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi kiri, di bawah Manajemen akses, pilih Kebijakan.
3. Di panel tengah, pilih Buat kebijakan.
4. Di bagian Editor kebijakan, pilih JSON.
5. Ganti semua JSON yang Anda lihat di editor dengan JSON berikut.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "sqs:GetQueueUrl",
                "sqs:ReceiveMessage",
                "sqs:SendMessage"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> Jika Anda memerlukan detail lebih lanjut untuk menyesuaikan kebijakan Anda, hubungi manajer pengiriman atau manajer akun AWS Blu Age Anda.

6. Pilih Berikutnya.
7. Masukkan nama untuk kebijakan tersebut, lalu pilih Buat kebijakan.

Membuat peran IAM

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi kiri, di bawah Manajemen akses, pilih Peran.

3. Di panel tengah, pilih Buat peran.
4. Di bagian Use case, pilih EC2 atau Elastic Container Service (dan kemudian Elastic Container Service Task), tergantung pada pilihan komputasi Anda.
5. Pilih Berikutnya.
6. Di kotak pencarian, masukkan nama kebijakan yang Anda buat sebelumnya.
7. Pilih kotak centang di sebelah kiri kebijakan Anda.
8. Pilih Berikutnya.
9. Masukkan nama peran, lalu pilih Buat peran.

Menjalankan AWS Blu Age Runtime di Amazon EC2

Membuat instans Amazon EC2

1. Buka konsol Amazon EC2 di https://console.aws.amazon.com/ec2/.
2. Pilih Luncurkan instans.
3. Untuk jenis Instance, pilih salah satu jenis yang tercantum dithe section called “Jenis instans Amazon EC2 untuk AWS Blu Age Runtime (di Amazon EC2)”.
4. Di bagian Key pair, pilih key pair yang ada atau buat yang baru.
5. Di bagian Pengaturan jaringan, pilih Pilih grup keamanan yang ada.
6. Untuk grup keamanan umum, pilih M2 BluagePrivateLink -SG.
7. Perluas bagian Detail lanjutan.
8. Untuk profil instans IAM, pilih peran IAM yang Anda buat sebelumnya.
9. Pilih Luncurkan instans.

Saat status instans Amazon EC2 berubah menjadi Running, sambungkan ke instans dan instal komponen perangkat lunak berikut:

- Lingkungan Runtime Java (JRE) 11.
- Apache Tomcat 9.
- AWS Blu Age Runtime (di Amazon EC2). Instal runtime AWS Blu Age di root folder instalasi Apache Tomcat (beberapa file akan ditambahkan sementara yang lain akan ditimpa).

Jika Anda ingin menginstal aplikasi web tambahan, instal di folder terpisah. Dalam hal ini, ulangi proses menginstal Apache Tomcat, lalu arsip di atasnya.

Menjalankan AWS Blu Age Runtime di Amazon ECS dikelola oleh AWS Fargate

Buat klaster Amazon ECS dan peran tugas yang akan digunakan saat meluncurkan AWS Fargate tugas nanti. Untuk peran tugas, sertakan kebijakan yang Anda buat sebelumnya. Bergantung pada skenario Anda, Anda mungkin perlu melampirkan kebijakan IAM tambahan ke peran tugas.

Jenis instans Amazon EC2 untuk AWS Blu Age Runtime (di Amazon EC2)

Berikut ini adalah daftar jenis instans Amazon EC2 yang dapat Anda gunakan untuk AWS Blu Age Runtime (di Amazon EC2).

```
t3.xlarge
t3.small
t3.large
t2.small
t2.large
r6i.xlarge
r6i.large
r6i.4xlarge
r6i.2xlarge
r5b.xlarge
r5b.large
r5b.2xlarge
r3.xlarge
m6i.xlarge
m6i.large
m6i.8xlarge
m6i.4xlarge
m6i.2xlarge
m6i.16xlarge
m5zn.xlarge
m5zn.large
m5zn.3xlarge
m5zn.2xlarge
m5.xlarge
m5.large
m5.8xlarge
m5.4xlarge
m5.2xlarge
m5.16xlarge
```

```
m5.12xlarge
c6i.xlarge
c6i.large
c6i.8xlarge
c6i.4xlarge
c6i.2xlarge
c6i.16xlarge
c5.xlarge
c5.large
c5.9xlarge
c5.4xlarge
c5.2xlarge
c5.18xlarge
c5.12xlarge
```

AWS Penerapan Blu Age Runtime di Amazon ECS dikelola oleh AWS Fargate

Topik di bagian ini menjelaskan cara mengatur AWS Blu Age Runtime di Amazon ECS yang dikelola oleh AWS Fargate, cara memperbarui versi runtime, cara memantau penerapan Anda menggunakan alarm CloudWatch Amazon, dan cara menambahkan dependensi berlisensi.

Topik

Menyiapkan AWS Blu Age Runtime di Amazon ECS yang dikelola oleh AWS Fargate

Topik ini menjelaskan cara menyiapkan dan menerapkan aplikasi PlanetsDemo sampel menggunakan AWS Blu Age Runtime di Amazon ECS yang dikelola oleh. AWS Fargate

AWS Blu Age Runtime di Amazon ECS yang dikelola oleh tersedia untuk AWS Fargate Linux/X86.

Topik

- Menyiapkan
- Uji PlanetsDemo aplikasinya

Prasyarat

Sebelum Anda mulai, pastikan Anda menyelesaikan prasyarat berikut.

- Konfigurasikan AWS CLI dengan mengikuti langkah-langkah dalam Mengonfigurasi AWS CLI.
- Lengkap the section called “AWS Prasyarat Runtime Blu Age” danthe section called “AWS Orientasi Blu Age Runtime”.
- Unduh AWS Blu Age Runtime di Amazon ECS yang dikelola oleh binari. AWS Fargate Untuk petunjuk, lihat the section called “AWS Orientasi Blu Age Runtime”.
- Unduh binari Apache Tomcat 9.
- Unduh arsip PlanetsDemo aplikasi.
- Buat database Amazon Aurora PostgreSQL untuk JICS, dan jalankan kueri di atasnya. `PlanetsDemo-v1/jics/sql/initJics.sql` Untuk informasi tentang cara membuat database PostgreSQL Amazon Aurora, lihat, Membuat dan menghubungkan ke klaster DB PostgreSQL Aurora.

Menyiapkan

Untuk mengatur aplikasi PlanetsDemo sampel, selesaikan langkah-langkah berikut.

1. Setelah mengunduh binari Apache Tomcat, ekstrak isinya, dan buka folder. `conf` Buka `catalina.properties` file untuk diedit dan ganti baris yang dimulai `common.loader` dengan baris berikut.

   ```
   common.loader="${catalina.base}/lib","${catalina.base}/lib/
   *.jar","${catalina.home}/lib","${catalina.home}/lib/*.jar","${catalina.home}/
   shared","${catalina.home}/shared/*.jar","${catalina.home}/extra","${catalina.home}/
   extra/*.jar"
   ```

2. Kompres folder Apache Tomcat dengan menggunakan perintah tar untuk membangun arsip `tar.gz`.
3. Siapkan Dockerfile untuk membangun gambar kustom Anda berdasarkan binari runtime yang disediakan dan binari server Apache Tomcat. Lihat contoh berikut Dockerfile. Tujuannya adalah untuk menginstal Apache Tomcat 9, diikuti oleh AWS Blu Age Runtime (untuk Amazon ECS

dikelola oleh AWS Fargate) diekstraksi di root direktori instalasi Apache Tomcat 9, dan kemudian untuk menginstal contoh aplikasi modern bernama. PlanetsDemo

> **Note**
>
> Isi skrip install-gapwalk.sh dan install-app.sh, yang digunakan dalam contoh ini Dockerfile, terdaftar setelah Dockerfile.

```
FROM --platform=linux/x86_64 amazonlinux:2

RUN mkdir -p /workdir/apps
WORKDIR /workdir
COPY install-gapwalk.sh .
COPY install-app.sh .
RUN chmod +x install-gapwalk.sh
RUN chmod +x install-app.sh

# Install Java and AWS CLI v2-y
RUN yum install sudo java-17-amazon-corretto unzip tar -y
RUN sudo yum remove awscli -y
RUN curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
 "awscliv2.zip"
RUN sudo unzip awscliv2.zip
RUN sudo ./aws/install

#·Installation dir
RUN mkdir -p /usr/local/velocity/installation/gapwalk
# Copy PlanetsDemo archive to a dedicated apps dir
COPY PlanetsDemo-v1.zip /workdir/apps/

# Copy resources (tomcat, blu age runtime) to installation dir
COPY tomcat.tar.gz /usr/local/velocity/installation/tomcat.tar.gz
COPY aws-bluage-on-fargate-runtime-3.10.0.15.tar.gz /usr/local/velocity/
installation/gapwalk/gapwalk-bluage-on-fargate.tar.gz

# run relevant installation scripts
RUN ./install-gapwalk.sh
RUN ./install-app.sh

EXPOSE 8080
EXPOSE 8081
```

```
# ...

# Run Command to start Tomcat server
CMD ["sh", "-c", "sudo /bluage-on-fargate/tomcat.gapwalk/velocity/startup.sh
 $ECS_CONTAINER_METADATA_URI_V4 $AWS_CONTAINER_CREDENTIALS_RELATIVE_URI"]
```

Berikut ini adalah isi dari install-gapwalk.sh.

```
#!/bin/sh

# Vars
TEMP_DIR=/bluage-on-fargate/tomcat.gapwalk/temp

# Install
echo "Installing Gapwalk and Tomcat"
sudo rm -rf /bluage-on-fargate
mkdir -p ${TEMP_DIR}
# Copy Blu Age runtime and tomcat archives to temporary extraction dir
sudo cp /usr/local/velocity/installation/gapwalk/gapwalk-bluage-on-fargate.tar.gz
 ${TEMP_DIR}
sudo cp /usr/local/velocity/installation/tomcat.tar.gz ${TEMP_DIR}
#·Create velocity dir
mkdir -p /bluage-on-fargate/tomcat.gapwalk/velocity
#·Extract tomcat files
tar -xvf ${TEMP_DIR}/tomcat.tar.gz -C ${TEMP_DIR}
# Copy all tomcat files to velocity dir
cp -fr ${TEMP_DIR}/apache-tomcat-9.x.x/* /bluage-on-fargate/tomcat.gapwalk/velocity
# Remove default webapps of Tomcat
rm -f /bluage-on-fargate/tomcat.gapwalk/velocity/webapps/*
# Extract Blu Age runtime at velocity dir
tar -xvf ${TEMP_DIR}/gapwalk-bluage-on-fargate.tar.gz -C /bluage-on-fargate/
tomcat.gapwalk
# Remove temporary extraction dir
sudo rm -rf ${TEMP_DIR}
```

Berikut ini adalah isi dari install-app.sh.

```
#!/bin/sh

APP_DIR=/workdir/apps
TOMCAT_GAPWALK_DIR=/bluage-on-fargate/tomcat.gapwalk
```

```
unzip ${APP_DIR}/PlanetsDemo-v1.zip -d ${APP_DIR}
cp -r ${APP_DIR}/webapps/* ${TOMCAT_GAPWALK_DIR}/velocity/webapps/
cp -r ${APP_DIR}/config/* ${TOMCAT_GAPWALK_DIR}/velocity/config/
```

4. Berikan informasi koneksi untuk database yang Anda buat sebagai bagian dari prasyarat dalam cuplikan berikut dalam `application-main.yml` file, yang terletak di folder. `{TOMCAT_GAPWALK_DIR}/config` Untuk informasi selengkapnya lihat, Membuat dan menghubungkan ke klaster DB PostgreSQL Aurora.

```
datasource:
   jicsDs:
     driver-class-name :
     url:
     username:
     password:
     type :
```

5. Buat dan dorong gambar ke repositori Amazon ECR Anda. Untuk petunjuknya, lihat Mendorong gambar Docker di Panduan Pengguna Amazon Elastic Container Registry.
6. Buka konsol di https://console.aws.amazon.com/ecs/v2.
7. Di panel navigasi kiri, pilih Definisi tugas.
8. Untuk jenis Peluncuran, pilih AWS Fargate.
9. Pilih peran tugas yang Anda buat sebagai bagian darithe section called “Persyaratan pengaturan infrastruktur”.
10. Lampirkan gambar Anda ke wadah.
11. Selesai mengisi formulir, lalu pilih Buat.
12. Di panel navigasi kiri, pilih Cluster, lalu pilih klaster Anda dari daftar.
13. Pada halaman detail klaster Anda, pada tab Layanan, pilih Buat.
14. Pilih definisi tugas.
15. Perluas bagian Jaringan, dan konfigurasikan VPC, subnet, dan grup keamanan yang Anda buat sebagai bagian darinya. the section called “Persyaratan pengaturan infrastruktur”
16. Terapkan layanan Amazon ECS Anda.

Jika penerapan gagal, periksa log. Untuk menemukannya, buka halaman tugas di Amazon ECS yang dikelola oleh AWS Fargate, lalu pilih tab Log. Jika Anda menemukan kode kesalahan yang dimulai dengan C diikuti oleh angka, seperti CXXXX, perhatikan pesan kesalahan. Misalnya, kode kesalahan

C5102 adalah kesalahan umum yang menunjukkan konfigurasi infrastruktur yang salah. Anda juga dapat menavigasi di dalam tugas yang sedang berjalan dan menjalankan beberapa perintah, mirip dengan AWS Blu Age Runtime (di Amazon EC2). Untuk informasi selengkapnya, lihat Menggunakan Amazon ECS Exec untuk men-debug di Panduan Pengembang Layanan Amazon Elastic Container.

Untuk membuka shell interaktif, jalankan perintah berikut dari mesin lokal Anda.

```
aws ecs execute-command --cluster your_cluster_name --container your_container_name --
task task_id --interactive --command /bin/sh
```

Uji PlanetsDemo aplikasinya

Untuk memeriksa status PlanetsDemo aplikasi yang digunakan, jalankan perintah berikut setelah Anda mengganti `load-balancer-DNS-namelistener-port`, dan `web-binary-name` dengan nilai yang benar untuk pengaturan Anda.

```
curl http://load-balancer-DNS-name:listener-port/gapwalk-application/
```

Jika aplikasi berjalan, Anda melihat pesan output berikut:`Jics application is running`.

Selanjutnya, jalankan perintah berikut.

```
curl http://load-balancer-DNS-name:listener-port/jac/api/services/rest/jicsservice/
```

Jika aplikasi berjalan, Anda melihat pesan output berikut:`Jics application is running`.

```
Jics application is running
```

Jika Anda telah mengkonfigurasi Blusam, Anda dapat mengharapkan respons kosong ketika Anda menjalankan perintah berikut.

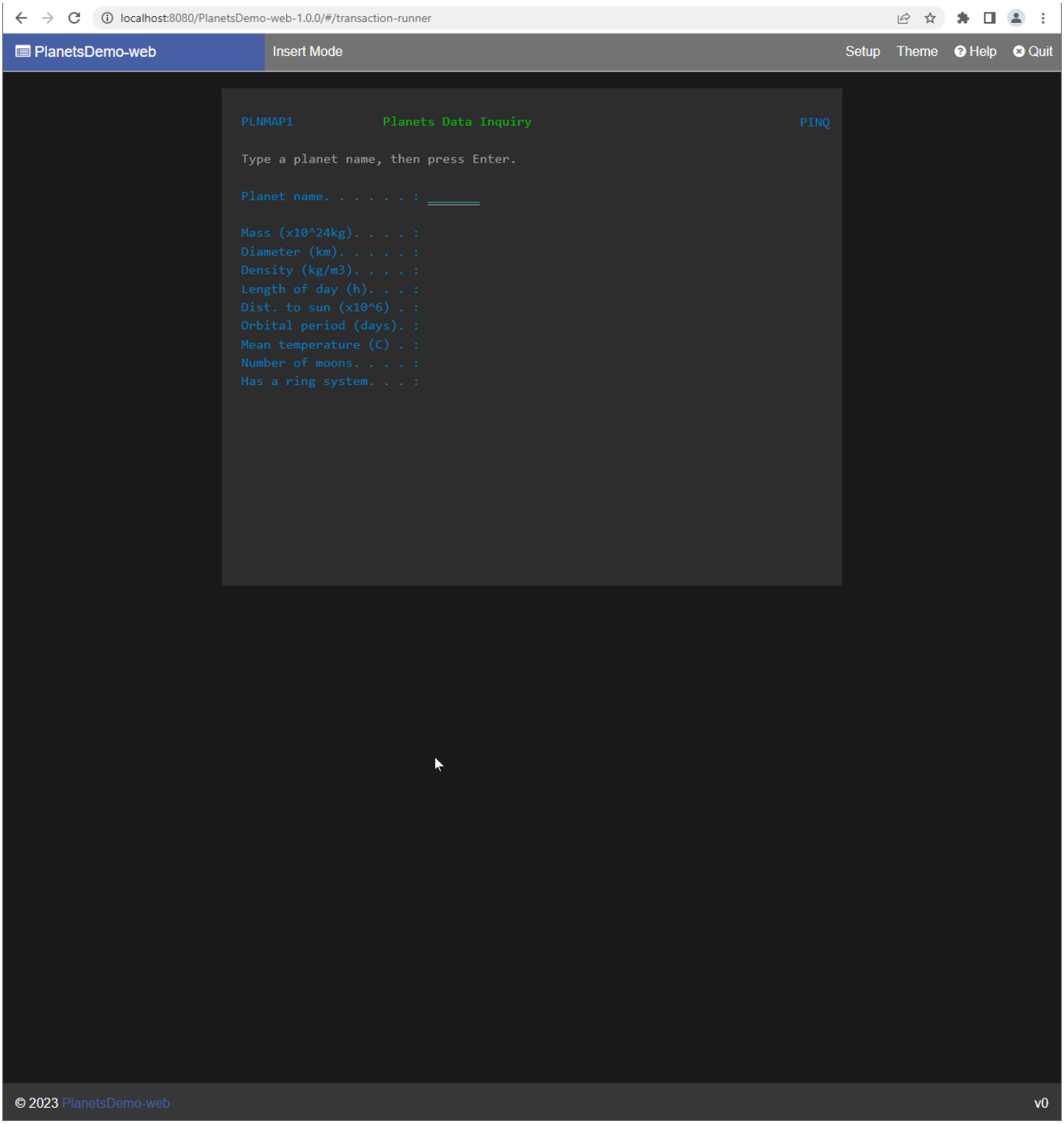
```
curl http://load-balancer-DNS-name:listener-port/bac/api/services/rest/bluesamserver/
serverIsUp
```

Perhatikan nama biner web (PlanetsDemo-web-1.0.0, jika tidak berubah). Untuk mengakses PlanetsDemo aplikasi, gunakan URL dengan format berikut.

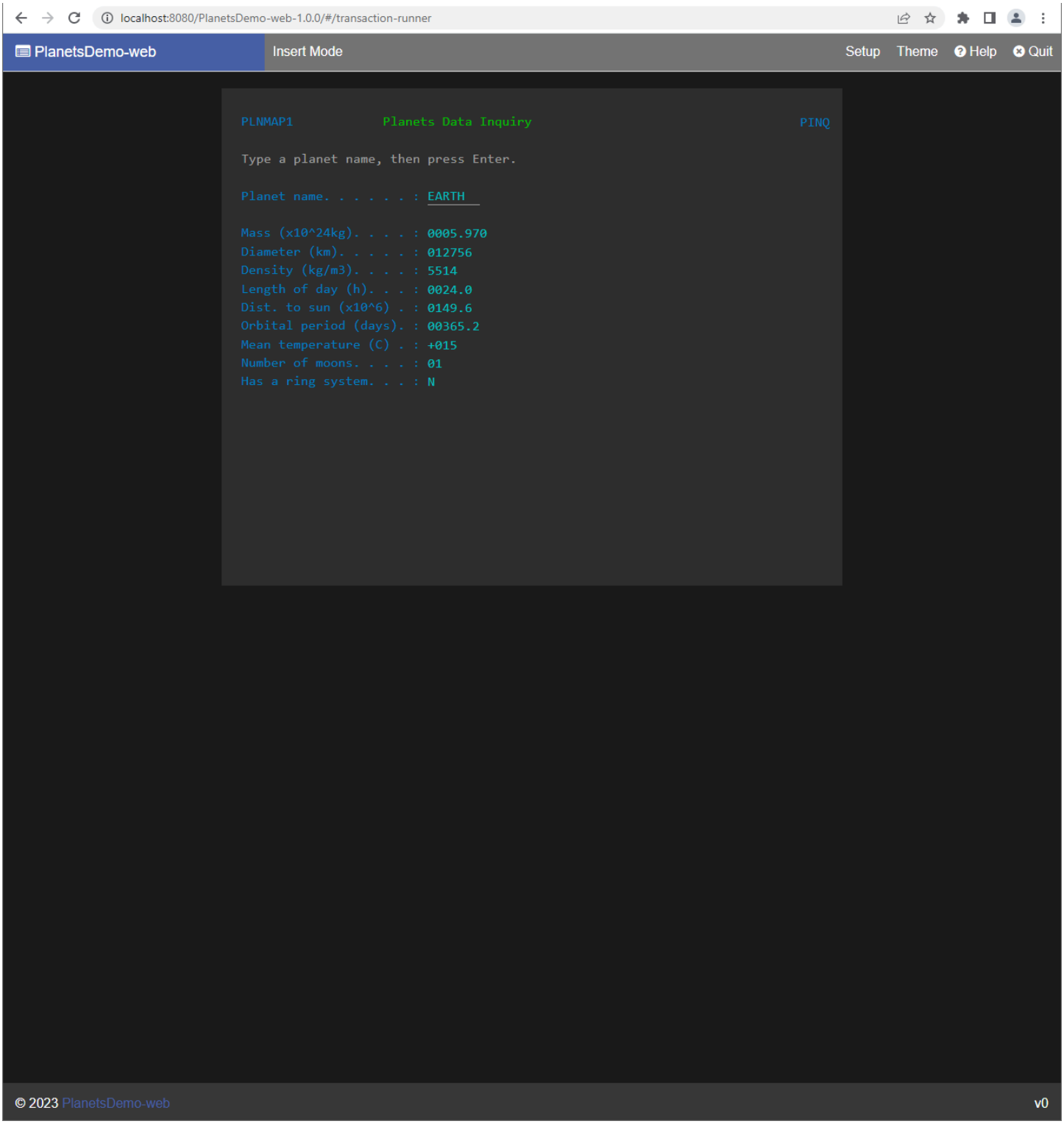
```
https://load-balancer-DNS-name:listener-port/web-binary-name
```

Setelah PlanetsDemo aplikasi dimulai, halaman beranda ditampilkan.

Masukkan PINQ di kotak teks dan kemudian tekan Enter. Halaman pertanyaan data ditampilkan.



Misalnya, masukkan EARTH di bidang PlanetsDemo nama, lalu tekan Enter. Halaman untuk planet yang Anda masukkan ditampilkan.


localhost:8080/PlanetsDemo-web-1.0.0/#/transaction-runner
PlanetsDemo-web
Insert Mode
Setup
Theme
Help
Quit
PLNMAP1 Planets Data Inquiry PINQ
Type a planet name, then press Enter.
Planet name. . . . . : EARTH
Mass (x10^24kg). . . . : 0005.970
Diameter (km). . . . . : 012756
Density (kg/m3). . . . : 5514
Length of day (h). . . : 0024.0
Dist. to sun (x10^6) . : 0149.6
Orbital period (days). : 00365.2
Mean temperature (C) . : +015
Number of moons. . . . : 01
Has a ring system. . . : N
© 2023 PlanetsDemo-web
v0

Memutakhirkan AWS Blu Age Runtime di Amazon ECS yang dikelola oleh AWS Fargate

Panduan ini menjelaskan cara meningkatkan AWS Blu Age Runtime di Amazon ECS yang dikelola oleh. AWS Fargate

Topik

Prasyarat

Sebelum Anda mulai, pastikan Anda memenuhi prasyarat berikut.

- Lengkap the section called “AWS Prasyarat Runtime Blu Age” danthe section called “AWS Orientasi Blu Age Runtime”.
- Unduh versi AWS Blu Age Runtime yang ingin Anda tingkatkan. Untuk informasi selengkapnya, lihat the section called “AWS Orientasi Blu Age Runtime”. Kerangka kerja terdiri dari dua file biner: `aws-bluage-runtime-x.x.x.x.tar.gz` dan`aws-bluage-webapps-x.x.x.x.tar.gz`.

Tingkatkan runtime kecepatan

Selesaikan langkah-langkah berikut untuk meningkatkan runtime kecepatan.

1. Bangun kembali gambar Docker Anda dengan versi AWS Blu Age Runtime yang diinginkan. Untuk petunjuk, lihat the section called “Menyiapkan AWS Blu Age Runtime di Amazon ECS yang dikelola oleh AWS Fargate”.
2. Dorong gambar Docker Anda ke repositori Amazon ECR Anda.
3. Hentikan dan mulai ulang layanan Amazon ECS Anda.
4. Verifikasi log.

AWS Blu Age Runtime berhasil ditingkatkan.

Amazon CloudWatch Alarm untuk AWS Blu Age Runtime di Amazon ECS dikelola oleh AWS Fargate

Untuk memiliki notifikasi yang lebih terlihat setiap kali aplikasi yang Anda gunakan menemukan pengecualian, siapkan CloudWatch untuk menerima log aplikasi Anda, dan tambahkan alarm untuk memperingatkan Anda tentang kemungkinan kesalahan.

Pengaturan alarm

Dengan CloudWatch log, Anda dapat mengonfigurasi sejumlah metrik dan alarm, tergantung pada aplikasi dan kebutuhan Anda.

Secara khusus, Anda dapat mengatur alarm proaktif untuk peringatan penggunaan secara langsung selama pembuatan klaster Amazon ECS, sehingga Anda mendapatkan pemberitahuan saat terjadi kesalahan. Untuk menyoroti kesalahan dalam koneksi ke sistem kontrol AWS Blu Age, tambahkan metrik mengenai string “Kesalahan C” di log. Anda kemudian dapat menentukan alarm yang bereaksi terhadap metrik ini.

Menyiapkan dependensi berlisensi di AWS Blu Age Runtime di Amazon ECS yang dikelola oleh AWS Fargate

Topik ini menjelaskan cara menyiapkan dependensi berlisensi tambahan yang dapat Anda gunakan dengan AWS Blu Age Runtime di Amazon ECS yang dikelola oleh. AWS Fargate

Topik

Prasyarat

Sebelum Anda mulai, pastikan Anda menyelesaikan prasyarat berikut.

- Lengkap the section called “AWS Prasyarat Runtime Blu Age” danthe section called “AWS Orientasi Blu Age Runtime”.
- Dapatkan dependensi berikut dari sumbernya.

Basis data Oracle

Menyediakan driver database Oracle. Misalnya, ojdbc8-19.8.0.0.jar.

Koneksi IBM MQ

Menyediakan klien IBM MQ. Misalnya, com.ibm.mq.allclient-9.3.0.15.jar.

Dengan versi dependensi ini, berikan juga dependensi transitif berikut:

- javax.jms-api-2.0.1.jar
- json-20080701.jar

File Printer DDS

Menyediakan perpustakaan laporan Jasper. Misalnya, jasperreports-6.16.0.jar, tetapi versi yang lebih baru mungkin kompatibel.

Dengan versi dependensi ini, berikan juga dependensi transitif berikut:

- castor-core-1.4.1.jar
- kastor-xml-1.4.1.jar
- commons-digester-2.1.jar
- ecj-3.21.0.jar
- itext-2.1.7.js8.jar
- javax.inject-1.jar
- jcommon-1.0.23.jar
- jfreechart-1.0.19.jar

Ikhtisar

Untuk menginstal dependensi, selesaikan langkah-langkah berikut.

1. Salin salah satu dependensi di atas seperti yang diperlukan ke folder build image Docker Anda.
2. Jika database JICS atau Blusam Anda di-host di Oracle, berikan driver database Oracle. *your-tomcat-path*`/extra`
3. Di Dockerfile Anda, salin dependensi ini ke. *your-tomcat-path*`/extra`
4. Buat image Docker Anda, lalu dorong ke Amazon ECR.
5. Hentikan dan mulai ulang layanan Amazon ECS Anda.

6. Periksa log.

AWS Penerapan Blu Age Runtime di Amazon EC2

Topik di bagian ini menjelaskan cara mengatur AWS Blu Age Runtime (tidak dikelola) di Amazon EC2, cara memperbarui versi runtime, cara memantau penerapan Anda menggunakan CloudWatch alarm Amazon, dan cara menambahkan dependensi berlisensi.

Topik

Menyiapkan AWS Blu Age Runtime (tidak dikelola) di Amazon EC2

Topik ini menjelaskan cara menyiapkan dan menerapkan aplikasi PlanetsDemo sampel menggunakan AWS Blu Age Runtime (tidak dikelola) di Amazon EC2.

Topik

Prasyarat

Sebelum Anda mulai, pastikan Anda menyelesaikan prasyarat berikut.

- Konfigurasikan AWS CLI dengan mengikuti langkah-langkah dalam Mengonfigurasi AWS CLI.
- Lengkap the section called “AWS Prasyarat Runtime Blu Age” danthe section called “AWS Orientasi Blu Age Runtime”.
- Buat instans Amazon EC2 yang berisi AWS Blu Age Runtime terbaru (di Amazon EC2). Untuk informasi selengkapnya, lihat Memulai instans Amazon EC2 Linux.
- Pastikan Anda dapat terhubung ke instans Amazon EC2 dengan sukses, misalnya dengan menggunakan SSM.

- Unduh dan ekstrak AWS Blu Age Runtime (di Amazon EC2) di. *your-tomcat-path*/* Untuk petunjuk, lihat the section called “AWS Orientasi Blu Age Runtime”.
- Unduh arsip PlanetsDemo aplikasi.
- Buka zip arsip dan unggah aplikasi ke ember Amazon S3 pilihan Anda.
- Buat database Amazon Aurora PostgreSQL untuk JICS, dan jalankan kueri di atasnya. `PlanetsDemo-v1/jics/sql/initJics.sql` Untuk informasi tentang cara membuat database PostgreSQL Amazon Aurora, lihat Membuat dan menghubungkan ke klaster DB PostgreSQL Aurora.

Menyiapkan

Untuk mengatur aplikasi PlanetsDemo sampel, selesaikan langkah-langkah berikut.

1. Hubungkan ke instans Amazon EC2 Anda dan buka `conf` folder di bawah folder instalasi Apache Tomcat 9 Anda. Buka `catalina.properties` file untuk diedit dan ganti baris yang dimulai `common.loader` dengan baris berikut.

```
common.loader="${catalina.base}/lib","${catalina.base}/lib/
*.jar","${catalina.home}/lib","${catalina.home}/lib/*.jar","${catalina.home}/
shared","${catalina.home}/shared/*.jar","${catalina.home}/extra","${catalina.home}/
extra/*.jar"
```

2. Arahkan ke *<your-tomcat-path>*`/webapps` folder.
3. Salin PlanetsDemo binari yang tersedia di PlanetsDemo folder -v1/webapps/ dari bucket Amazon S3 menggunakan perintah berikut.

```
aws s3 cp s3://path-to-demo-app-webapps/ . --recursive
```

> **Note**
>
> Ganti `path-to-demo-app-webapps` dengan URI Amazon S3 yang benar untuk bucket tempat Anda membuka ritsleting arsip sebelumnya. PlanetsDemo

4. Salin konten `PlanetsDemo-v1/config/` folder ke*<your-tomcat-path>*`/config/`.
5. Berikan informasi koneksi untuk database yang Anda buat sebagai bagian dari prasyarat dalam cuplikan berikut dalam file. `application-main.yml` Untuk informasi selengkapnya lihat, Membuat dan menghubungkan ke klaster DB PostgreSQL Aurora.

```
datasource:
   jicsDs:
     driver-class-name :
     url:
     username:
     password:
     type :
```

6. Mulai server Apache Tomcat Anda dan verifikasi log.

```
your-tomcat-path/startup.sh

tail -f your-tomcat-path/logs/catalina.log
```

Jika Anda menemukan kode kesalahan yang dimulai dengan C diikuti oleh angka, seperti CXXXX, perhatikan pesan kesalahan. Misalnya, kode kesalahan C5102 adalah kesalahan umum yang menunjukkan konfigurasi infrastruktur yang salah.

Uji PlanetsDemo aplikasinya

Untuk memeriksa status PlanetsDemo aplikasi yang digunakan, jalankan perintah berikut setelah Anda mengganti `load-balancer-DNS-namelistener-port`, dan `web-binary-name` dengan nilai yang benar untuk pengaturan Anda.

```
curl http://load-balancer-DNS-name:listener-port/gapwalk-application/
```

Jika aplikasi berjalan, Anda melihat pesan output berikut:`Jics application is running`.

Selanjutnya, jalankan perintah berikut.

```
curl http://load-balancer-DNS-name:listener-port/jac/api/services/rest/jicsservice/
```

Jika aplikasi berjalan, Anda melihat pesan output berikut:`Jics application is running`.

```
Jics application is running
```

Jika Anda telah mengkonfigurasi Blusam, Anda dapat mengharapkan respons kosong ketika Anda menjalankan perintah berikut.

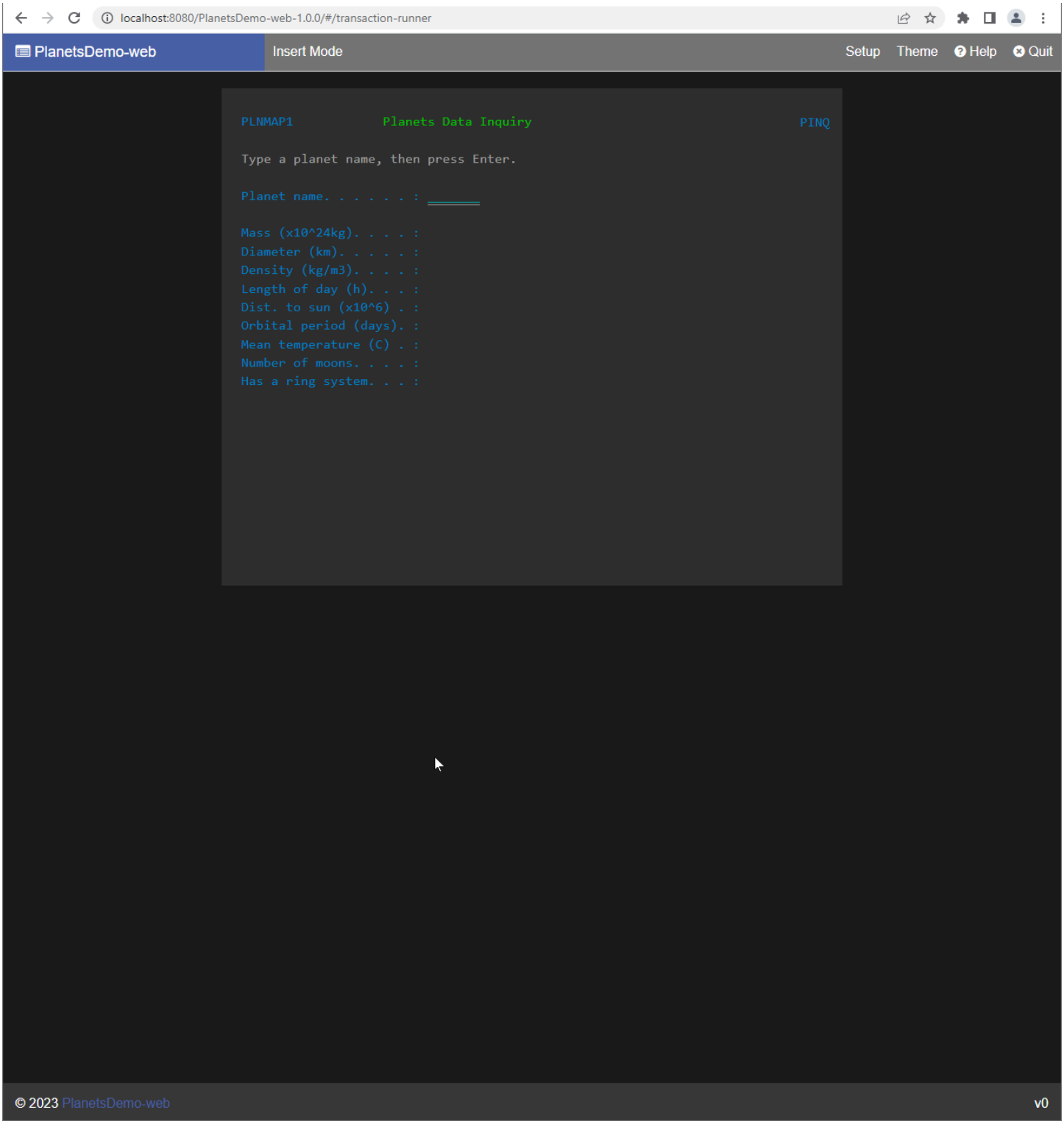
```
curl http://load-balancer-DNS-name:listener-port/bac/api/services/rest/bluesamserver/
serverIsUp
```

Perhatikan nama biner web (PlanetsDemo-web-1.0.0, jika tidak berubah). Untuk mengakses PlanetsDemo aplikasi, gunakan URL dengan format berikut.

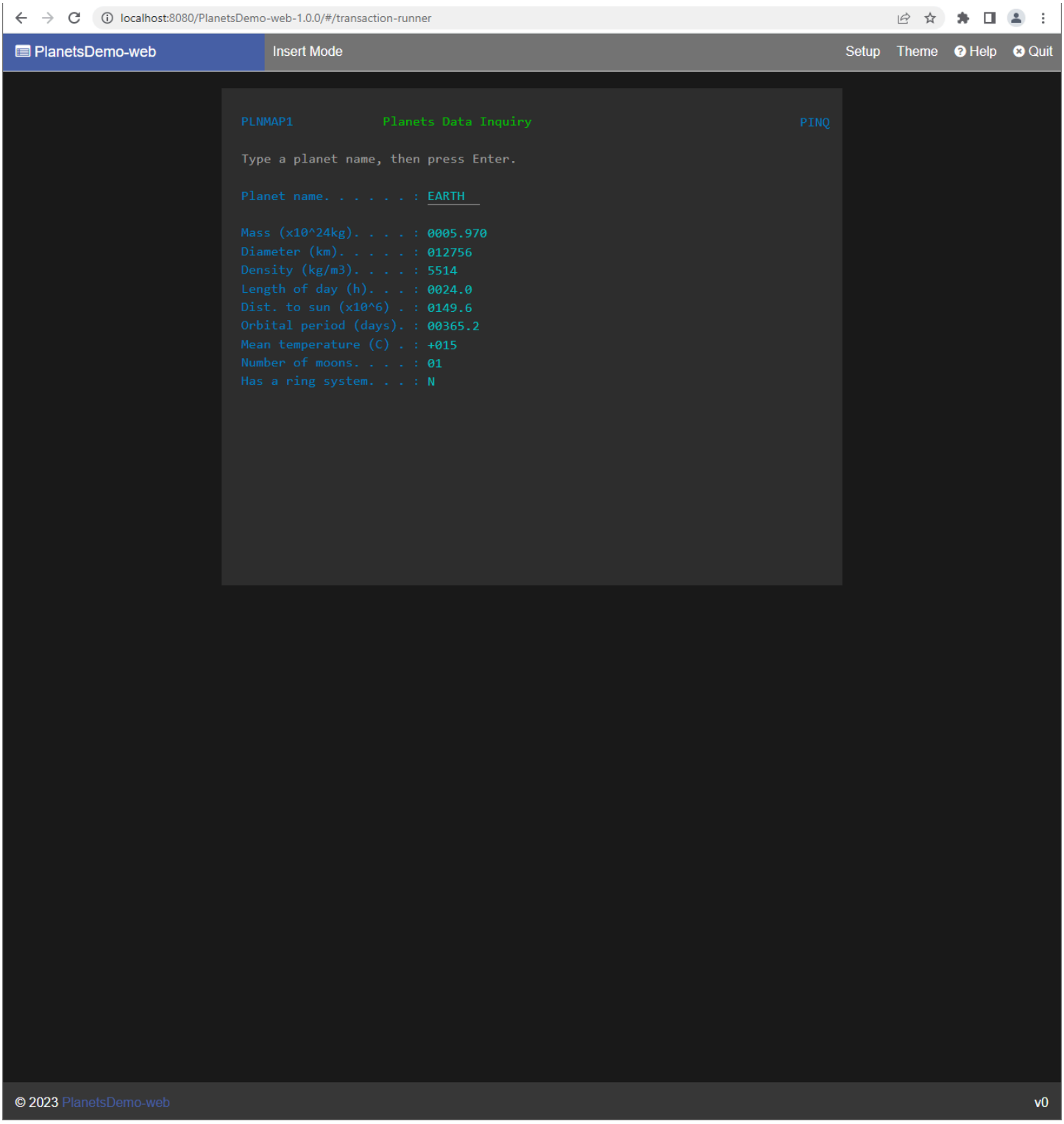
```
https://load-balancer-DNS-name:listener-port/web-binary-name
```

Setelah PlanetsDemo aplikasi dimulai, halaman beranda ditampilkan.

Masukkan PINQ di kotak teks dan kemudian tekan Enter. Halaman pertanyaan data ditampilkan.



Misalnya, masukkan EARTH di bidang PlanetsDemo nama, lalu tekan Enter. Halaman untuk planet yang Anda masukkan ditampilkan.



Memutakhirkan Runtime AWS Blu Age di Amazon EC2

Panduan ini menjelaskan cara memutakhirkan AWS Blu Age Runtime di Amazon EC2.

Topik

Prasyarat

Sebelum Anda mulai, pastikan Anda memenuhi prasyarat berikut.

- Lengkap the section called “AWS Prasyarat Runtime Blu Age” danthe section called “AWS Orientasi Blu Age Runtime”.
- Pastikan Anda memiliki instans Amazon EC2 yang berisi runtime AWS Blu Age terbaru. Untuk informasi selengkapnya, lihat Memulai instans Amazon EC2 Linux.
- Pastikan Anda dapat terhubung ke instans Amazon EC2 dengan sukses, misalnya, dengan menggunakan SSM.
- Unduh versi AWS Blu Age Runtime (di Amazon EC2) yang ingin Anda tingkatkan. Untuk informasi lebih lanjut, lihat the section called “AWS Pengaturan Blu Age Runtime (tidak dikelola)” Kerangka kerja terdiri dari dua file biner: `aws-bluage-runtime-x.x.x.x.tar.gz` dan`aws-bluage-webapps-x.x.x.x.tar.gz`.

Ikhtisar

Selesaikan langkah-langkah berikut untuk meningkatkan runtime kecepatan.

1. Connect ke instans Amazon EC2 Anda dan ubah pengguna ke su dengan menjalankan perintah berikut.

   ```
   sudo su
   ```

   Anda memerlukan hak Superuser untuk menjalankan perintah dalam tutorial ini.
2. Buat dua folder, satu untuk setiap file biner.
3. Beri nama setiap folder dengan nama yang sama dengan file biner.
4. Salin setiap file biner ke folder yang sesuai.

> **Warning**
>
> Mengekstrak setiap biner menghasilkan folder dengan nama yang sama. Oleh karena itu, jika Anda mengekstrak kedua file biner di lokasi yang sama satu demi satu, Anda akan menimpa konten.

5. Untuk mengekstrak binari, gunakan perintah berikut. Jalankan perintah di setiap folder.

```
tar xvf aws-bluage-runtime-x.x.x.x.tar.gz
tar xvf aws-bluage-webapps-x.x.x.x.tar.gz
```

6. Hentikan layanan Apache Tomcat dengan menggunakan perintah berikut.

```
systemctl stop tomcat.service
systemctl stop tomcat-webapps.service
```

7. Ganti konten `<your-tomcat-path>/shared/` dengan kontenaws-bluage-runtime-x.x.x.x/velocity/shared/.
8. Ganti `<your-tomcat-path>/webapps/gapwalk-application.war` denganaws-bluage-runtime-x.x.x.x/velocity/webapps/gapwalk-application.war.
9. Ganti file perang di`<your-tomcat-path>/webapps/`, yaitu `bac.war` dan`jac.war`, dengan file yang sama dariaws-bluage-webapps-x.x.x.x/velocity/webapps/.
10. Mulai layanan Apache Tomcat dengan menjalankan perintah berikut.

```
systemctl start tomcat.service
systemctl start tomcat-webapps.service
```

11. Periksa log.

Untuk memeriksa status aplikasi yang digunakan, jalankan perintah berikut.

```
curl http://localhost:8080/gapwalk-application/
```

Pesan berikut akan muncul.

```
Jics application is running
```

```
curl http://localhost:8181/jac/api/services/rest/jicsservice/
```

Pesan berikut akan muncul.

```
Jics application is running
```

```
curl http://localhost:8181/bac/api/services/rest/bluesamserver/serverIsUp
```

Responsnya harus kosong.

Runtime AWS Blu Age berhasil ditingkatkan.

AWS Blu Age Runtime (di Amazon EC2) Alarm Amazon CloudWatch

Agar Anda memiliki pemberitahuan yang lebih terlihat ketika aplikasi yang Anda gunakan menemukan pengecualian yang akan menempatkan aplikasi Anda dalam masa tenggang, Anda dapat mengatur CloudWatch untuk menerima log aplikasi Anda, dan menambahkan alarm untuk memperingatkan Anda tentang kemungkinan kesalahan.

Penyebaran Logging CloudWatch

Secara default, `application-main.yml` file tersebut menyertakan referensi ke file konfigurasi logging lain bernama`logback-cloudwatch.yml`.

```
logging:
  config: classpath:logback-cloudwatch.xml
```

Kedua file berada di folder konfigurasi dan ini adalah bagaimana CloudWatch logging dikonfigurasi, seperti yang dijelaskan di bagian berikut.

Konfigurasi CloudWatch logging

`logback-cloudwatch.xml`File default memiliki konten berikut.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE configuration>
<configuration>
```

```
    <appender name="console" class="ch.qos.logback.core.ConsoleAppender">
        <encoder>
            <pattern>%date{yyyy-MM-dd HH:mm:ss.SSS,UTC}  %level --- [%thread{15}]
 %logger{40} : %msg%n%xThrowable</pattern>
        </encoder>
    </appender>

    <appender name="cloudwatch"
 class="com.netfective.bluage.runtime.cloudwatchlogger.CloudWatchAppender">
        <logGroup>BluAgeRuntimeOnEC2-Logs</logGroup>
        <logStream>%date{yyyy-MM-dd,UTC}.%instanceId.%uuid</logStream>
        <layout>
            <pattern>%date{yyyy-MM-dd HH:mm:ss.SSS,UTC}  %level --- [%thread{15}]
 %logger{40} : %msg%n%xThrowable</pattern>
        </layout>
        <appender-ref ref="console" />
    </appender>

    <root level="INFO">
        <appender-ref ref="cloudwatch" />
    </root>
</configuration>
```

Segala sesuatu di luar <appender name="cloudwatch"/> elemen adalah konfigurasi logback standar. Ada dua appender dalam file ini: appender konsol untuk mengirim log ke konsol dan CloudWatch appender untuk mengirim log ke. CloudWatch

`level`Atribut dalam `root` elemen menentukan tingkat logging dari seluruh aplikasi.

Nilai yang diperlukan di dalam tag <appender name="cloudwatch"/> adalah:

- <logGroup/>:Menetapkan nama grup log di CloudWatch. Jika nilai tidak ditentukan, defaultnya. `BluAgeRuntimeOnEC2-Logs` Jika grup log tidak ada maka akan dibuat secara otomatis. Perilaku ini dapat diubah melalui konfigurasi, yang dibahas di bawah ini.
- <logStream/>: Menetapkan nama LogStream (di dalam grup log) di. CloudWatch

Nilai opsional:

- <region/>: Mengganti Wilayah tempat aliran log akan ditulis. Secara default, log masuk ke Wilayah yang sama dengan instans EC2.
- <layout/>: Pola pesan log akan digunakan.

- <maxbatchsize/>: Jumlah maksimum pesan log untuk dikirim CloudWatch per operasi.
- <maxbatchtimemillis/>: Waktu dalam milidetik untuk memungkinkan CloudWatch log ditulis.
- <maxqueuewaittimemillis/>: Waktu dalam milidetik untuk mencoba menyisipkan permintaan dalam antrian log internal.
- <internalqueuesize/>: Ukuran maksimum antrian internal.
- <createlogdests/>: Buat grup log dan aliran log jika tidak ada.
- <initialwaittimemillis/>: Jumlah waktu yang Anda inginkan thread untuk tidur saat startup. Penantian awal ini memungkinkan akrual awal log.
- <maxeventmessagesize/>: Ukuran maksimum peristiwa log. Log yang melebihi ukuran ini tidak akan dikirim.
- <truncateeventmessages/>: Memutus pesan yang terlalu panjang.
- <printrejectedevents/>: Aktifkan appender darurat.

CloudWatch penyiapan

Agar konfigurasi di atas dapat mendorong log dengan benar CloudWatch, perbarui peran profil instans Amazon EC2 IAM Anda untuk memberinya izin tambahan untuk grup log BluAgeRuntimeOn `EC2-logs` dan aliran lognya:

- `logs:CreateLogStream`
- `logs:DescribeLogStreams`
- `logs:CreateLogGroup`
- `logs:PutLogEvents`
- `logs:DescribeLogGroups`

Pengaturan alarm

Berkat CloudWatch log, Anda kemudian dapat mengonfigurasi metrik dan alarm yang berbeda, tergantung pada aplikasi dan kebutuhan Anda. Secara khusus, Anda dapat mengatur alarm proaktif untuk peringatan penggunaan, agar diperingatkan jika terjadi kesalahan yang mungkin menempatkan aplikasi Anda dalam masa tenggang (dan pada akhirnya, mencegahnya berfungsi sama sekali). Untuk mencapai ini, Anda dapat menambahkan metrik mengenai string “Error C5001" di log, yang menyoroti kesalahan dalam koneksi ke sistem kontrol AWS Blu Age. Anda kemudian dapat menentukan alarm yang bereaksi terhadap metrik ini.

Menyiapkan dependensi berlisensi di AWS Blu Age Runtime di Amazon EC2

Panduan ini menjelaskan cara mengatur dependensi berlisensi tambahan yang dapat Anda gunakan dengan AWS Blu Age Runtime di Amazon EC2.

Topik

Prasyarat

Sebelum Anda mulai, pastikan Anda menyelesaikan prasyarat berikut.

- Lengkap the section called “AWS Prasyarat Runtime Blu Age” danthe section called “AWS Orientasi Blu Age Runtime”.
- Pastikan Anda memiliki instans Amazon EC2 yang berisi AWS Blu Age Runtime terbaru (di Amazon EC2). Untuk informasi selengkapnya, lihat Memulai instans Amazon EC2 Linux.
- Pastikan Anda dapat terhubung ke instans Amazon EC2 dengan sukses, misalnya, dengan menggunakan SSM.
- Dapatkan dependensi berikut dari sumbernya.

Basis data Oracle

Menyediakan driver database Oracle. Kami menguji fungsionalitas AWS Blu Age Runtime (di Amazon EC2) dengan versi ojdbc8-19.8.0.0.jar, tetapi versi yang lebih baru mungkin kompatibel.

Koneksi IBM MQ

Menyediakan klien IBM MQ. Kami menguji fungsionalitas AWS Blu Age Runtime (di Amazon EC2) dengan versi com.ibm.mq.allclient-9.3.0.15.jar, tetapi versi yang lebih baru mungkin kompatibel.

Dengan versi dependensi ini, berikan juga dependensi transitif berikut:

- javax.jms-api-2.0.1.jar
- json-20080701.jar

File Printer DDS

Menyediakan perpustakaan laporan Jasper. Kami menguji fungsionalitas AWS Blu Age Runtime (di Amazon EC2) dengan jasperreports-6.16.0.jar, tetapi versi yang lebih baru mungkin kompatibel.

Dengan versi dependensi ini, berikan juga dependensi transitif berikut:

- castor-core-1.4.1.jar
- kastor-xml-1.4.1.jar
- commons-digester-2.1.jar
- ecj-3.21.0.jar
- itext-2.1.7.js8.jar
- javax.inject-1.jar
- jcommon-1.0.23.jar
- jfreechart-1.0.19.jar

Ikhtisar

Untuk menginstal dependensi, selesaikan langkah-langkah berikut.

1. Connect ke instans Amazon EC2 Anda dan ubah pengguna ke su dengan menjalankan perintah berikut.

   ```
   sudo su
   ```

   Anda memerlukan hak Superuser untuk menjalankan perintah dalam tutorial ini.
2. Arahkan ke `<your-tomcat-path>/extra/` folder.

   ```
   cd <your-tomcat-path>/extra/
   ```
3. Salin salah satu dependensi di atas seperti yang diperlukan di folder ini.
4. Hentikan dan mulai tomcat.service dengan menjalankan perintah berikut.

   ```
   systemctl stop tomcat.service
   ```

   ```
   systemctl start tomcat.service
   ```

5. Periksa status layanan untuk memastikannya berjalan.

```
systemctl status tomcat.service
```

6. Verifikasi log.

Siapkan dependensi untuk aplikasi web JAC dan BAC

1. Jika database JICS atau Blusam Anda di-host di Oracle maka Anda perlu menyediakan driver database Oracle. `<your-tomcat-path>/extra`
2. Buat folder jika belum ada.
3. Berhenti dan restart server Apache Tomcat Anda.
4. Verifikasi log.

Ubah kode sumber dengan Blu Age Developer IDE

Jika Anda menggunakan mesin runtime AWS Blu Age yang AWS dikelola, Anda dapat menggunakan Blu Age Developer untuk memodifikasi kode sumber yang dihasilkan. Anda mungkin ingin melakukan ini jika Anda perlu memperbarui kode modern untuk beberapa alasan, atau jika sebagian dari kode sumber lama tidak dapat dimodernisasi. Anda mengakses Blu Age Developer melalui Amazon AppStream 2.0. Bagian ini menjelaskan cara mengatur Blu Age Developer di AppStream 2.0. Ini juga menjelaskan cara menggunakan Blu Age Developer untuk memperbarui kode sumber, menggunakan aplikasi PlanetsDemo sampel.

Topik

- Tutorial: Mengatur AppStream 2.0 untuk IDE Pengembang AWS Blu Age
- Tutorial: Gunakan AWS Blu Age Developer di 2.0 AppStream

Tutorial: Mengatur AppStream 2.0 untuk IDE Pengembang AWS Blu Age

AWS Modernisasi Mainframe menyediakan beberapa alat melalui Amazon 2.0. AppStream AppStream 2.0 adalah layanan streaming aplikasi yang dikelola sepenuhnya dan aman yang memungkinkan Anda melakukan streaming aplikasi desktop ke pengguna tanpa menulis ulang aplikasi. AppStream 2.0 memberi pengguna akses instan ke aplikasi yang mereka butuhkan dengan pengalaman pengguna yang responsif dan lancar pada perangkat pilihan mereka. Menggunakan AppStream 2.0 untuk meng-host alat khusus mesin runtime memberi tim aplikasi pelanggan

kemampuan untuk menggunakan alat langsung dari browser web mereka, berinteraksi dengan file aplikasi yang disimpan di bucket Amazon S3 atau repositori. CodeCommit

Untuk informasi tentang dukungan browser di AppStream 2.0, lihat Persyaratan Sistem dan Dukungan Fitur (Browser Web) di Panduan Administrasi Amazon AppStream 2.0. Jika Anda memiliki masalah saat menggunakan AppStream 2.0, lihat Memecahkan Masalah Pengguna AppStream 2.0 di Panduan Administrasi Amazon AppStream 2.0.

Dokumen ini menjelaskan cara mengatur IDE Pengembang AWS Blu Age pada armada AppStream 2.0.

Topik

Prasyarat

Unduh file arsip yang berisi artefak yang Anda perlukan untuk mengatur AWS Blu Age Developer IDE di bawah AppStream 2.0.

> **Note**
>
> Ini adalah file besar. Jika Anda memiliki masalah dengan waktu operasi habis, sebaiknya gunakan instans Amazon EC2 untuk meningkatkan kinerja unggahan dan unduhan.

Langkah 1: Buat bucket Amazon S3.

Buat bucket Amazon S3 Wilayah AWS sama dengan armada AppStream 2.0 yang akan Anda buat. Bucket ini akan berisi artefak yang Anda butuhkan untuk menyelesaikan tutorial ini.

Langkah 2: Lampirkan kebijakan ke bucket S3

Lampirkan kebijakan berikut ke bucket yang Anda buat untuk tutorial ini. Pastikan untuk mengganti `MYBUCKET` dengan nama sebenarnya dari bucket yang Anda buat.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "AllowAppStream2.0ToRetrieveObjects",
        "Effect": "Allow",
        "Principal": {
            "Service": "appstream.amazonaws.com"
        },
        "Action": "s3:GetObject",
        "Resource": "arn:aws:s3:::MYBUCKET/*"
    }]
}
```

Langkah 3: Unggah file ke bucket Amazon S3

Buka zip file yang Anda unduh di Prasyarat dan unggah `appstream` folder ke ember Anda. Mengunggah folder ini akan menciptakan struktur yang benar di bucket Anda. Untuk informasi selengkapnya, lihat Mengunggah objek di Panduan Pengguna Amazon S3.

Langkah 4: Unduh AWS CloudFormation templat

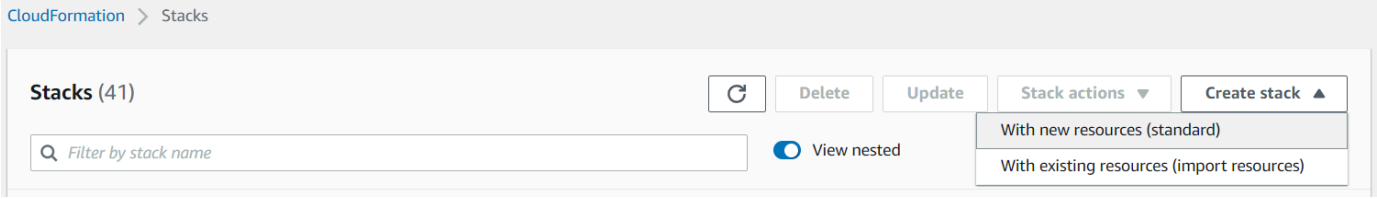
Unduh AWS CloudFormation templat berikut. Anda memerlukan template ini untuk membuat dan mengisi armada AppStream 2.0.

- cfn-m2- .yaml appstream-elastic-fleet-linux
- cfn-m2- -linux.yaml appstream-bluage-dev-tools
- cfn-m2- .yaml appstream-bluage-shared-linux
- cfn-m2- .yaml appstream-chrome-linux
- cfn-m2- .yaml appstream-eclipse-jee-linux
- cfn-m2- .yaml appstream-pgadmin-linux

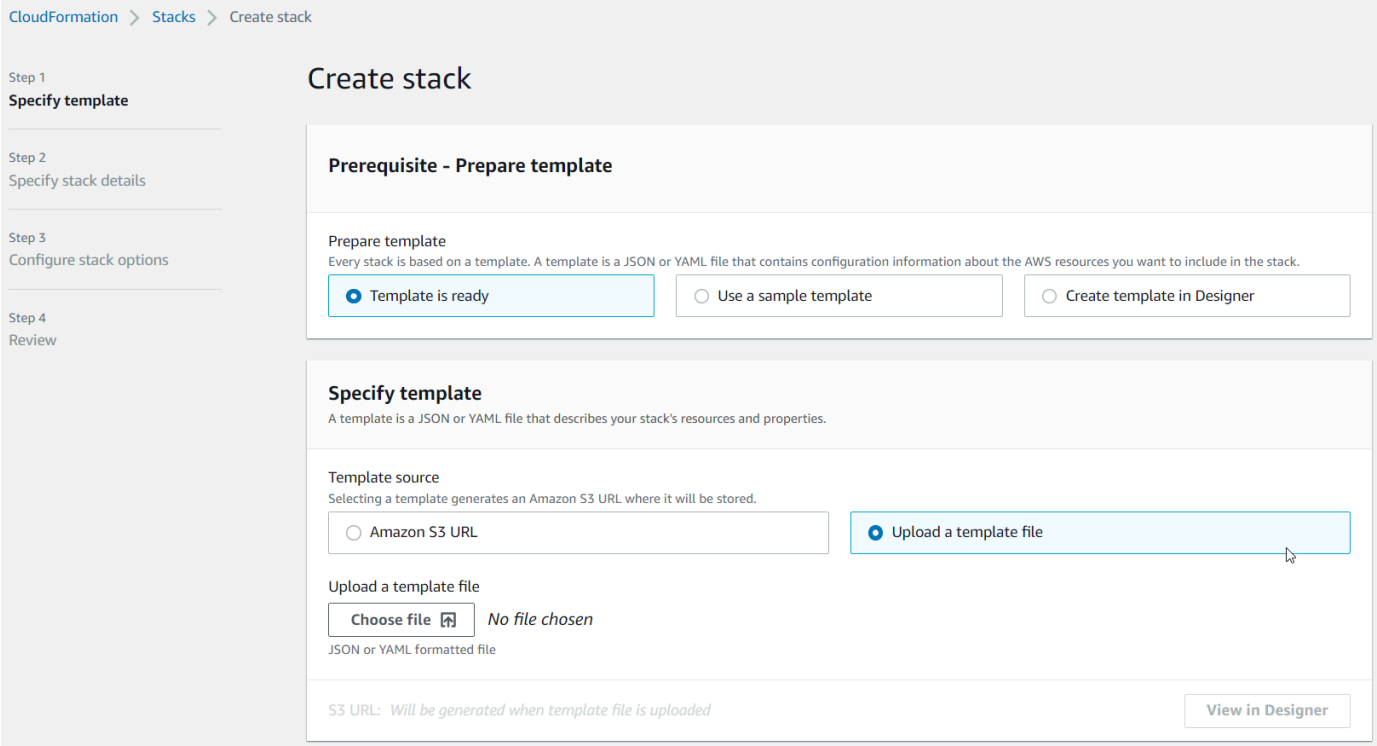
Langkah 5: Buat armada dengan AWS CloudFormation

Pada langkah ini, Anda menggunakan `cfn-m2-appstream-elastic-fleet-linux.yaml` AWS CloudFormation template untuk membuat armada AppStream 2.0 dan tumpukan untuk meng-host IDE Pengembang AWS Blu Age. Setelah Anda membuat armada dan tumpukan, Anda akan menjalankan AWS CloudFormation template lain yang Anda unduh pada langkah sebelumnya untuk menginstal IDE Pengembang dan alat lain yang diperlukan.

1. Arahkan ke AWS CloudFormation di konsol AWS Manajemen, dan pilih Tumpukan.
2. Di Stacks, pilih Create stack dan With new Resources (standar):



3. Di Buat tumpukan, pilih Template siap dan Upload file template:



4. Pilih Pilih file, dan navigasikan ke file`cfn-m2-appstream-elastic-fleet-linux.yaml`. Pilih Berikutnya.
5. Di Tentukan detail tumpukan, berikan informasi berikut:

   - Sebuah nama untuk tumpukan.

- Grup keamanan default Anda dan dua subnet dari grup keamanan tersebut.

> **Note**
>
> Dua subnet kelompok keamanan harus berada di zona ketersediaan yang berbeda.

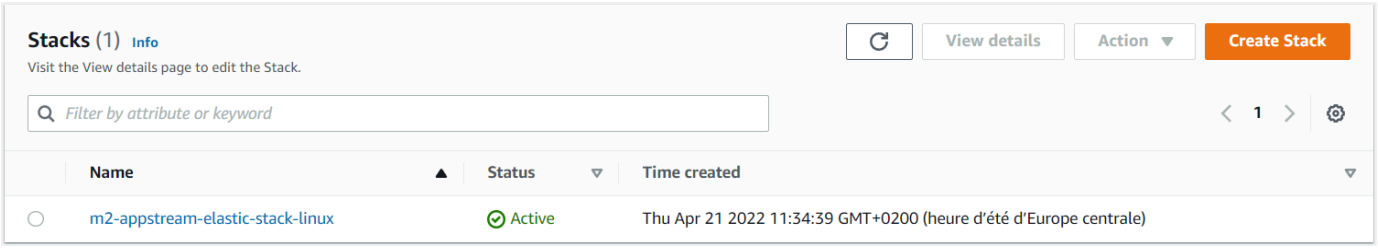
6. Pilih Berikutnya, lalu pilih Berikutnya lagi.
7. Pilih Saya mengakui yang AWS CloudFormation mungkin membuat sumber daya IAM dengan nama khusus. , dan kemudian pilih Submit.
8. Setelah Anda membuat armada, buat CloudFormation tumpukan dengan templat lain yang diunduh untuk menyelesaikan pengaturan aplikasi. Pastikan untuk memperbarui BucketNamesetiap kali untuk menunjuk ke bucket S3 yang benar. Anda dapat mengedit BucketNamedi CloudFormation konsol. Atau, Anda dapat mengedit file template secara langsung dan memperbarui `S3Bucket` properti.

> **Note**
>
> Template yang diunduh berharap dapat menemukan aset dalam bucket S3 dengan struktur folder yang disebut`appstream/bluage/developer-ide/`. Ember harus Wilayah AWS sama dengan armada yang Anda buat.

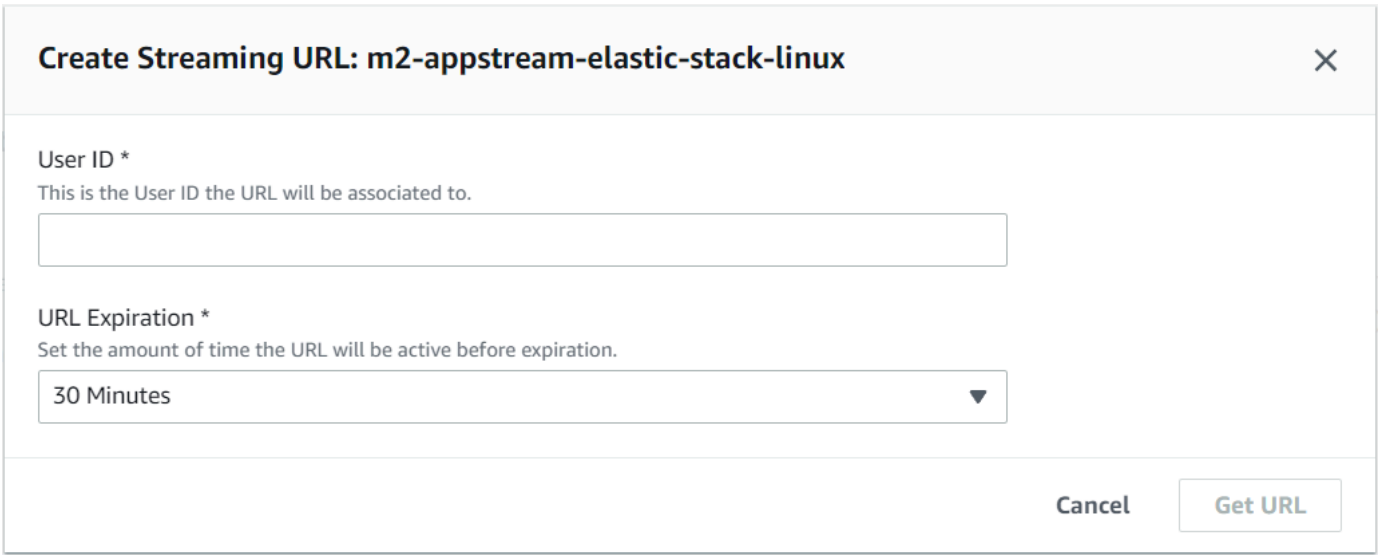
Langkah 6: Akses sebuah instance

Setelah Anda membuat dan memulai armada, Anda dapat membuat tautan sementara untuk mengakses armada melalui klien asli.

1. Arahkan ke AppStream 2.0 di AWS Management Console dan pilih tumpukan yang dibuat sebelumnya:



2. Pada halaman detail tumpukan, pilih Tindakan, lalu pilih Buat URL Streaming:



3. Di Buat URL Streaming, masukkan ID Pengguna arbitrer dan waktu kedaluwarsa URL, lalu pilih Dapatkan URL. Anda mendapatkan URL yang dapat Anda gunakan untuk streaming ke browser atau ke klien asli. Kami menyarankan Anda melakukan streaming ke klien asli.

Pembersihan sumber daya

Untuk prosedur membersihkan tumpukan dan armada yang dibuat, lihat Membuat Armada AppStream 2.0 dan Tumpukan.

Ketika Anda telah menghapus objek AppStream 2.0, Anda atau administrator akun juga dapat membersihkan bucket S3 untuk Pengaturan Aplikasi dan Folder Rumah.

> **Note**
>
> Folder home untuk pengguna tertentu unik di semua armada, jadi Anda mungkin perlu menyimpannya jika tumpukan AppStream 2.0 lainnya aktif di akun yang sama.

Anda tidak dapat menggunakan konsol AppStream 2.0 untuk menghapus pengguna. Sebagai gantinya, Anda harus menggunakan API layanan dengan file AWS CLI. Untuk informasi selengkapnya, lihat Administrasi Kumpulan Pengguna di Panduan Administrasi Amazon AppStream 2.0.

Tutorial: Gunakan AWS Blu Age Developer di 2.0 AppStream

Tutorial ini menunjukkan cara mengakses AWS Blu Age Developer di AppStream 2.0 dan menggunakannya dengan contoh aplikasi sehingga Anda dapat mencoba fitur-fiturnya. Ketika Anda menyelesaikan tutorial ini, Anda dapat menggunakan langkah yang sama dengan aplikasi Anda sendiri.

Topik

Langkah 1: Buat database

Pada langkah ini, Anda menggunakan Amazon RDS untuk membuat database PostgreSQL terkelola yang digunakan aplikasi demo untuk menyimpan informasi konfigurasi.

1. Buka konsol Amazon RDS.
2. Pilih Database > Buat database.
3. Pilih Standard create > PostgreSQL, tinggalkan versi default, lalu pilih Tingkat gratis.
4. Pilih pengidentifikasi instans DB.
5. Untuk Pengaturan Kredensi, pilih Kelola kredenal master di. AWS Secrets Manager Untuk informasi selengkapnya, lihat Manajemen kata sandi dengan Amazon RDS dan AWS Secrets Manager di Panduan Pengguna Amazon RDS.
6. Pastikan VPC sama dengan yang Anda gunakan untuk instance AppStream 2.0. Anda dapat meminta admin Anda untuk nilai ini.

7. Untuk grup keamanan VPC, pilih Buat Baru.
8. Tetapkan akses Publik ke Ya.
9. Tinggalkan semua nilai default lainnya. Tinjau nilai-nilai ini.
10. Pilih Buat basis data.

Untuk membuat server database dapat diakses dari instans Anda, pilih server database di Amazon RDS. Di bawah Konektivitas & keamanan, pilih grup keamanan VPC untuk server database. Grup keamanan ini sebelumnya dibuat untuk Anda dan harus memiliki deskripsi yang mirip dengan yang ada di Dibuat oleh konsol manajemen RDS. Pilih Tindakan > Edit aturan masuk, pilih Tambahkan aturan, dan buat aturan tipe PostgreSQL. Untuk sumber aturan, gunakan default grup keamanan Anda dapat mulai mengetik nama sumber di bidang Sumber dan kemudian menerima ID yang disarankan. Terakhir, pilih Simpan aturan.

Langkah 2: Akses lingkungan

Pada langkah ini, Anda mengakses lingkungan pengembangan AWS Blu Age di AppStream 2.0.

1. Hubungi administrator Anda untuk cara yang tepat untuk mengakses instans AppStream 2.0 Anda. Untuk informasi umum tentang kemungkinan klien dan konfigurasi, lihat Metode Akses AppStream 2.0 dan Klien di Panduan Administrasi Amazon AppStream 2.0. Pertimbangkan untuk menggunakan klien asli untuk pengalaman terbaik.
2. Di AppStream 2.0 pilih Desktop.

Langkah 3: Mengatur runtime

Pada langkah ini, Anda mengatur runtime AWS Blu Age. Anda harus mengatur runtime pada peluncuran pertama dan lagi jika Anda diberi tahu tentang peningkatan runtime. Langkah ini mengisi `.m2` folder Anda.

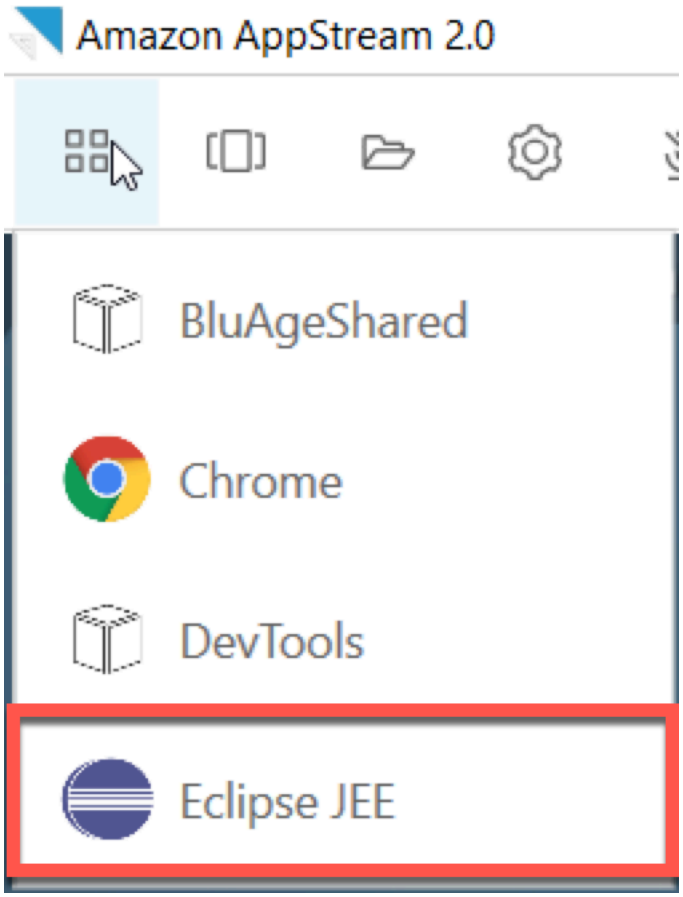
1. Pilih Aplikasi, dari bilah menu, lalu pilih Terminal.
2. Masukkan perintah berikut:

```
~/_install-velocity-runtime.sh
```

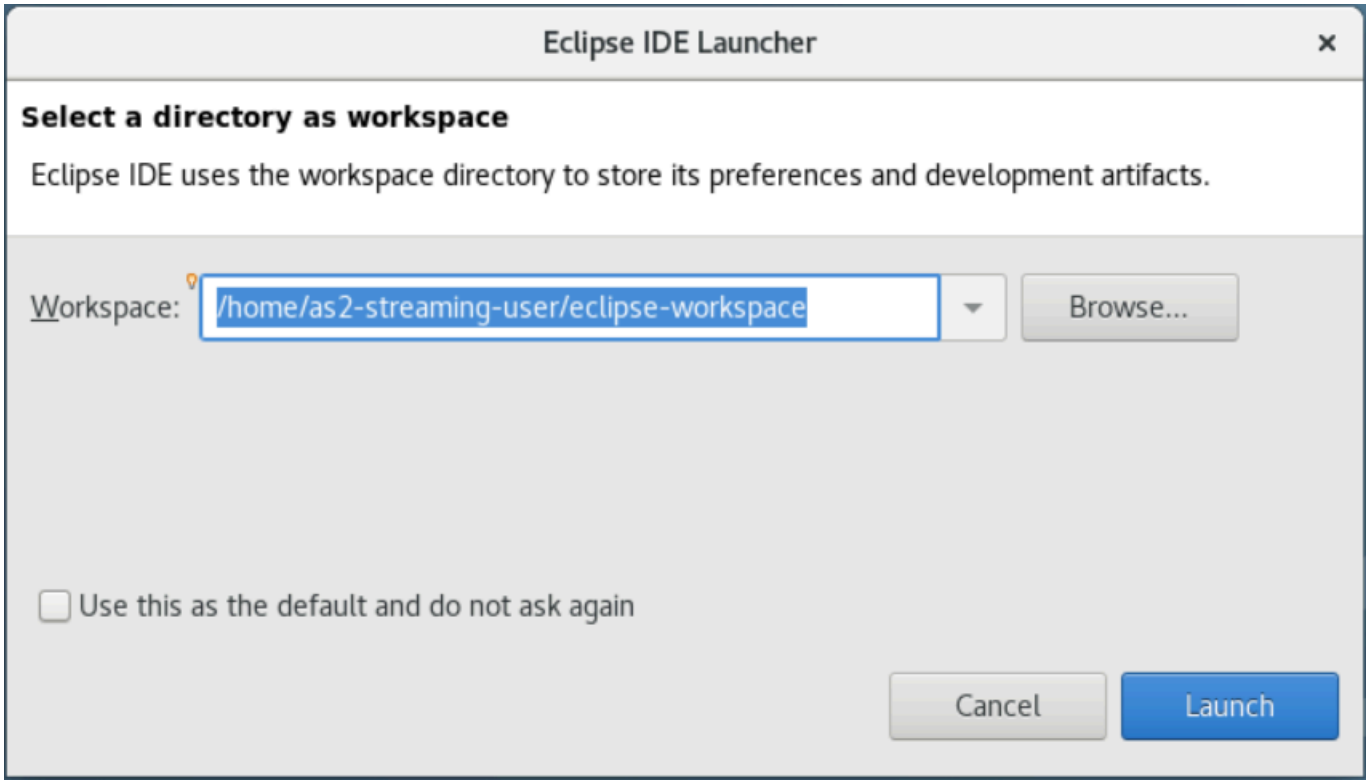
Langkah 4: Mulai IDE Eclipse

Pada langkah ini, Anda memulai Eclipse IDE dan memilih lokasi di mana Anda ingin membuat ruang kerja.

1. Di AppStream 2.0 pilih ikon Launch Application pada toolbar, dan kemudian pilih Eclipse JEE.



2. Saat peluncur terbuka, masukkan lokasi tempat Anda ingin membuat ruang kerja, dan pilih Luncurkan.



Secara opsional, Anda dapat meluncurkan Eclipse dari baris perintah, sebagai berikut:

```
~/eclipse &
```

Langkah 5: Siapkan proyek Maven

Pada langkah ini, Anda mengimpor proyek Maven untuk aplikasi demo Planets.

1. Unggah PlanetsDemo-pom.zip ke folder Home Anda. Anda dapat menggunakan fitur “File Saya” klien asli untuk melakukan ini.
2. Gunakan alat baris `unzip` perintah untuk mengekstrak file.
3. Arahkan ke dalam folder yang tidak di-zip dan buka root `pom.xml` proyek Anda di editor teks.
4. Edit `gapwalk.version` properti sehingga cocok dengan runtime AWS Blu Age yang diinstal.

   Jika Anda tidak yakin dengan versi yang diinstal, keluarkan perintah berikut di terminal:

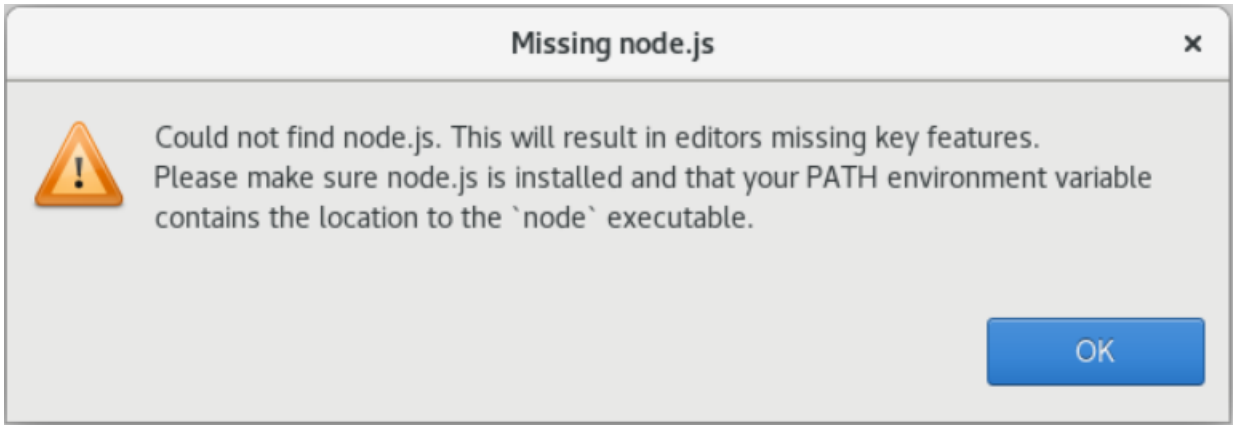
   ```
   cat ~/runtime-version.txt
   ```

   Perintah ini mencetak versi runtime yang tersedia saat ini, misalnya,`3.1.0-b3257-dev`.

> **Note**
>
> Jangan sertakan `-dev` sufiks dalam`gapwalk.version`. Misalnya, nilai yang valid adalah`<gapwalk.version>3.1.0-b3257</gapwalk.version>`.

5. Di Eclipse, pilih File, lalu Impor. Di jendela dialog Impor, perluas Maven dan pilih Proyek Maven yang Ada. Pilih Berikutnya.
6. Di Impor Proyek Maven, berikan lokasi file yang diekstrak dan pilih Selesai.

   Anda dapat dengan aman mengabaikan popup berikut. Maven mengunduh salinan lokal `node.js` untuk membangun bagian Angular (*-web) dari proyek:

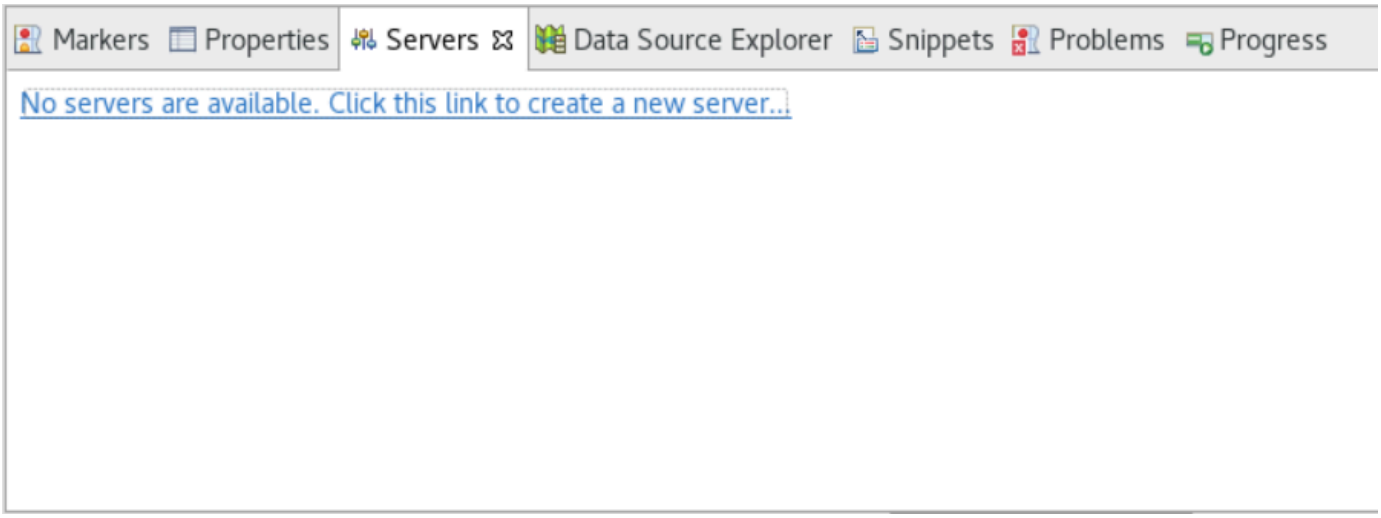
   

   Tunggu sampai akhir build. Anda dapat mengikuti build di tampilan Progress.
7. Di Eclipse, pilih proyek dan pilih Run as. Kemudian pilih Maven install. Setelah instalasi Maven berhasil, itu membuat file di `war` bawah. `PlanetsDemoPom/PlanetsDemo-web/target/PlanetsDemo-web-1.0.0.war`

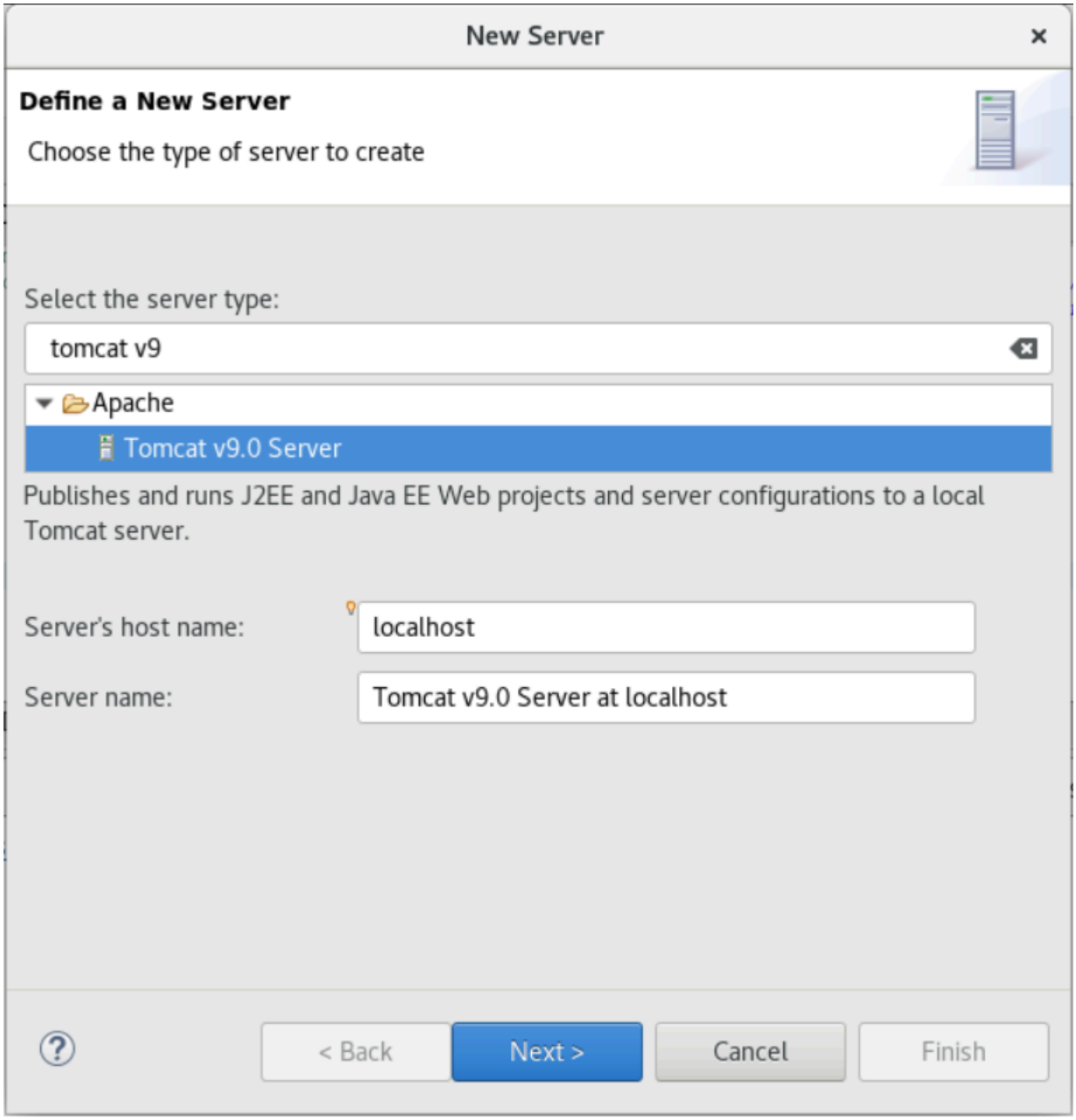
Langkah 6: Konfigurasikan server Tomcat

Pada langkah ini, Anda mengonfigurasi server Tomcat tempat Anda menyebarkan dan memulai aplikasi yang dikompilasi.

1. Di Eclipse, pilih Window > Show View > Server untuk menampilkan tampilan Server:

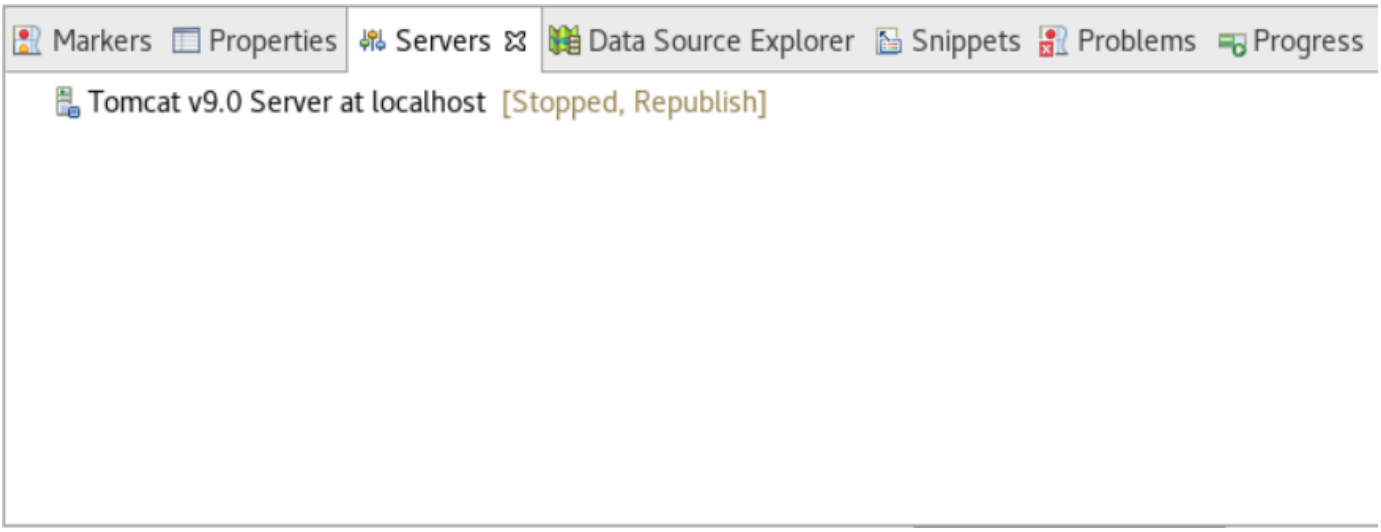


2. Pilih Tidak ada server yang tersedia. Klik tautan ini untuk membuat server baru... . Wizard Server Baru muncul. Di bidang Pilih jenis server wizard, masukkan tomcat v9, dan pilih Tomcat v9.0 Server. Lalu pilih Selanjutnya.



3. Pilih Browse, dan pilih folder tomcat di root folder Home. Biarkan JRE pada nilai defaultnya dan pilih Selesai.

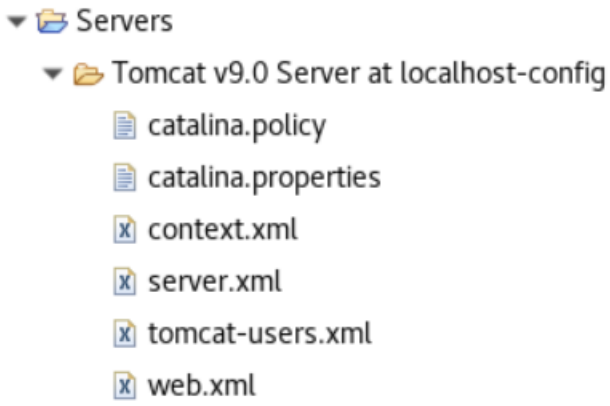
   Proyek Server dibuat di ruang kerja, dan server Tomcat v9.0 sekarang tersedia di tampilan Server. Di sinilah aplikasi yang dikompilasi akan digunakan dan dimulai:



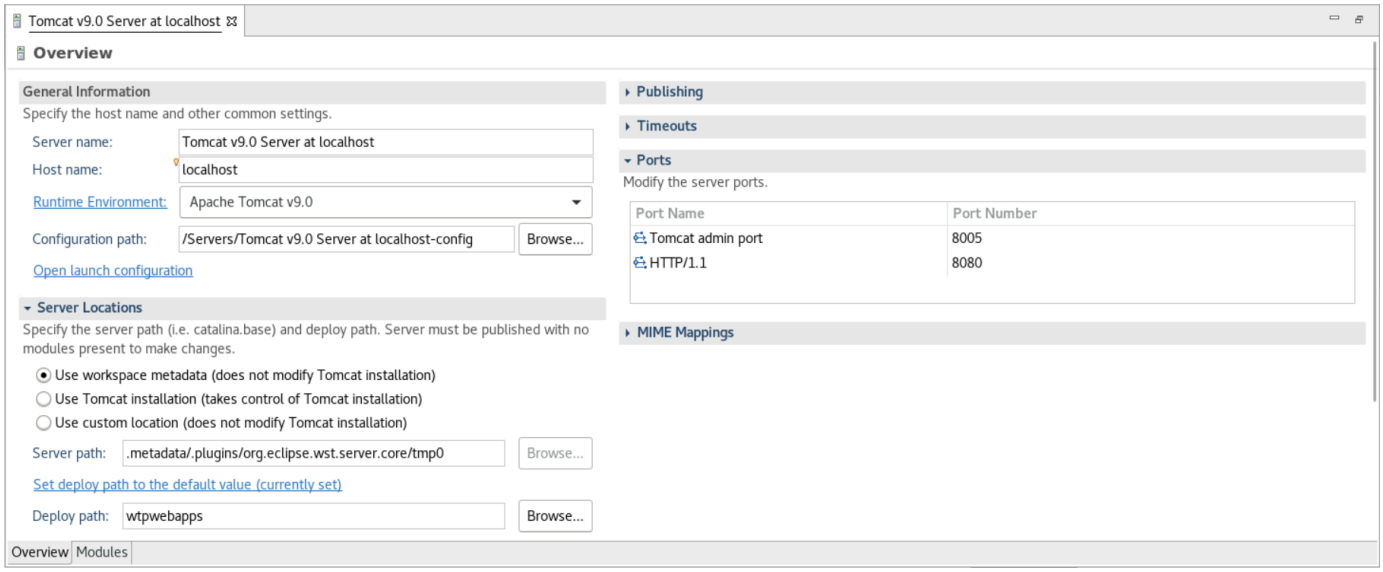
Langkah 7: Menyebarkan ke Tomcat

Pada langkah ini, Anda menyebarkan aplikasi demo Planets ke server Tomcat sehingga Anda dapat menjalankan aplikasi.

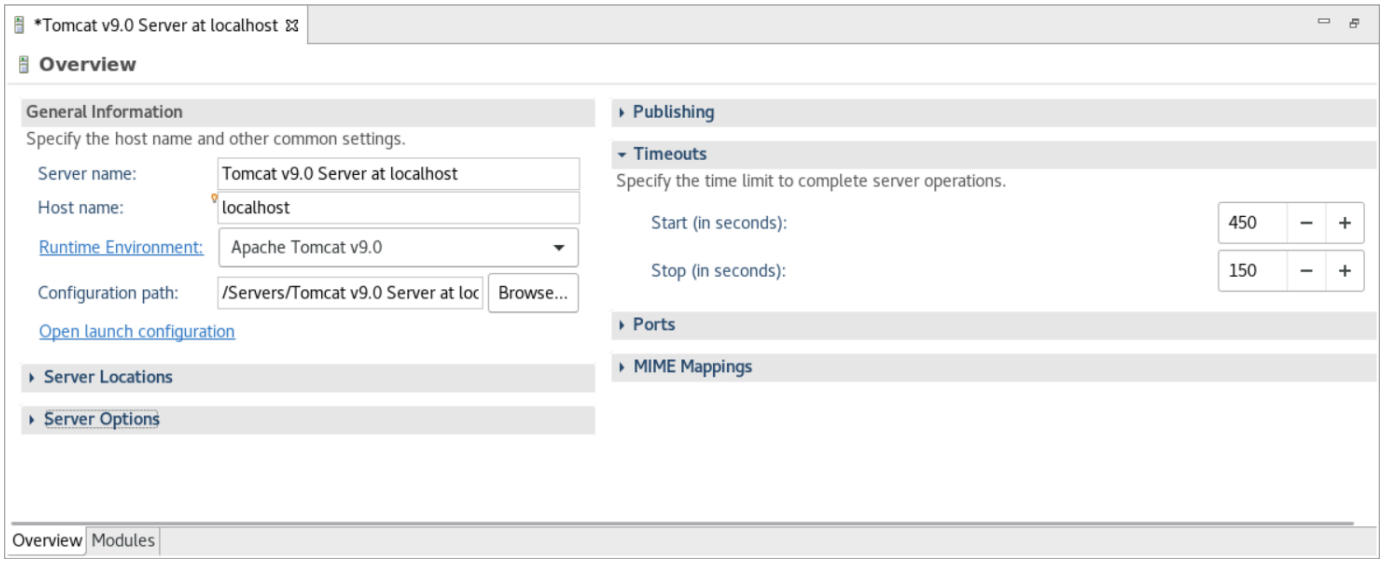
1. Pilih `PlanetsDemo-web` file dan pilih Run As > Maven install. Pilih `PlanetsDemo-web` lagi dan pilih Refresh untuk memastikan bahwa frontend yang dikompilasi npm dikompilasi dengan benar ke .war dan diperhatikan oleh Eclipse.
2. Unggah PlanetsDemo-runtime.zip ke instance, dan unzip file di lokasi yang dapat diakses. Ini memastikan bahwa aplikasi demo dapat mengakses folder konfigurasi dan file yang dibutuhkannya.
3. Salin konten `PlanetsDemo-runtime/tomcat-config` ke dalam `Servers/Tomcat v9.0...` subfolder yang Anda buat untuk server Tomcat Anda:



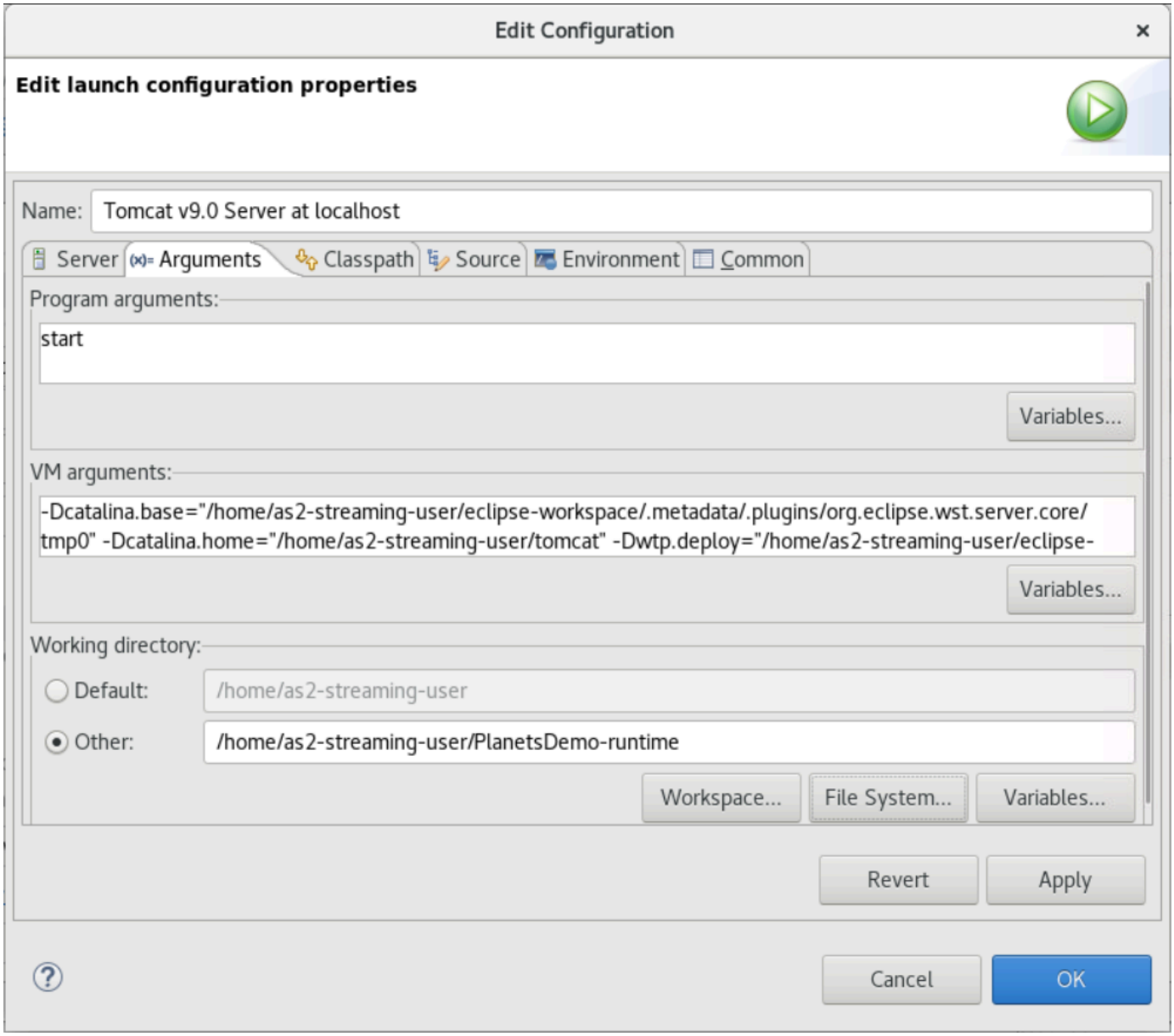
4. Buka entri `tomcat v9.0` server di tampilan Server. Editor properti server muncul:



5. Di tab Ikhtisar, tingkatkan nilai Timeout menjadi 450 detik untuk Mulai, dan 150 detik untuk Berhenti, seperti yang ditunjukkan di sini:



6. Pilih Buka konfigurasi peluncuran. Seorang penyihir muncul. Di wizard, arahkan ke folder Argumen dan, untuk direktori Kerja, pilih Lainnya. Pilih File System, dan navigasikan ke `PlanetsDemo-runtime` folder yang di-unzip sebelumnya. Folder ini harus berisi subfolder langsung yang disebut config.

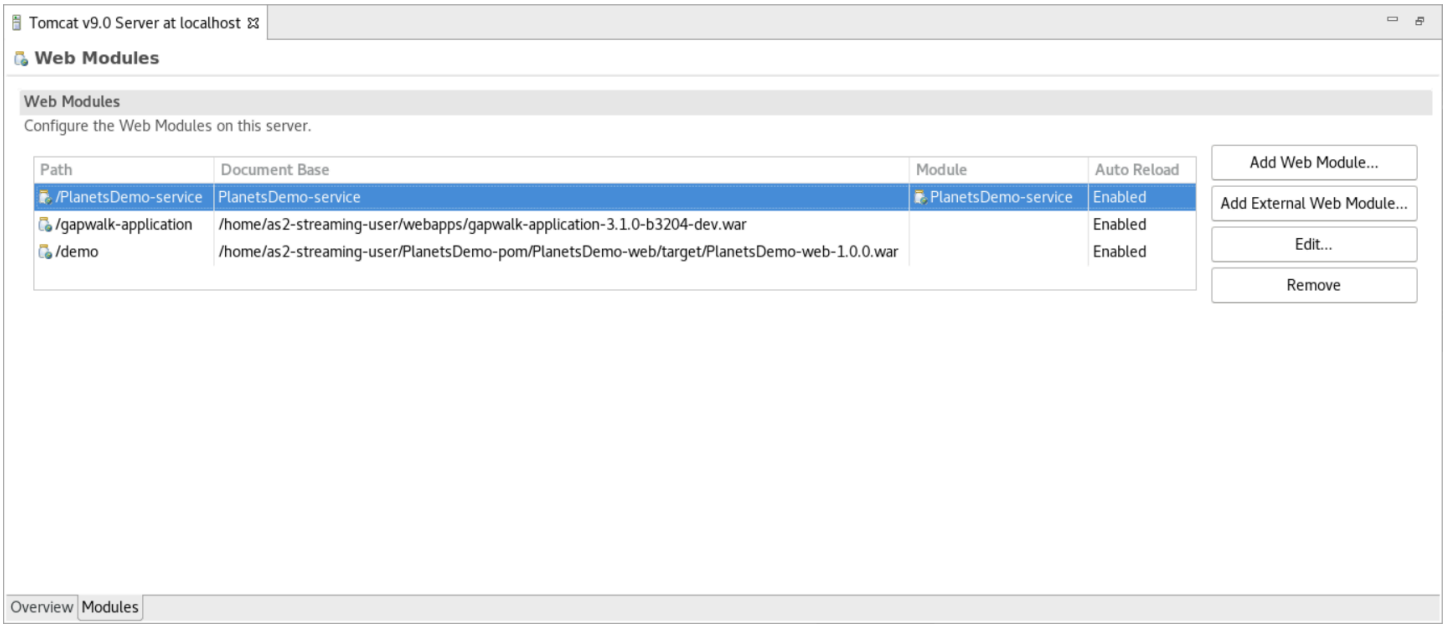


7. Pilih tab Modul editor properti server dan buat perubahan berikut:

   - Pilih Add Web Module dan tambahkan`PlanetsDemo-service`.
   - Pilih Tambahkan Modul Web Eksternal. Jendela dialog Add Web Module muncul. Lakukan perubahan berikut:
     - Di Dasar dokumen, pilih Browse dan navigasikan ke `~/webapps/gapwalk-application...war`
     - Di Jalan, masuk`/gapwalk-application`.
   - Pilih OK.
   - Pilih Add External Web Module lagi dan buat perubahan berikut:

    - Untuk basis Document, masukkan path ke frontend .war (in) `PlanetsDemo-web/target`
    - Untuk Path, masukkan `/demo`
  - Pilih OK
  - Simpan modifikasi editor (Ctrl+S).

Konten editor sekarang harus mirip dengan yang berikut ini.



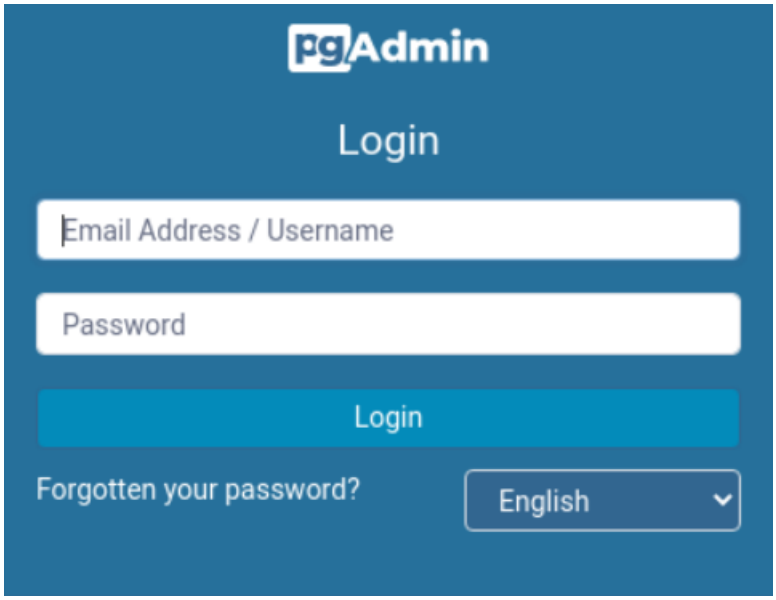
Langkah 8: Buat database JICS

Pada langkah ini, Anda terhubung ke database yang Anda buatLangkah 1: Buat database.

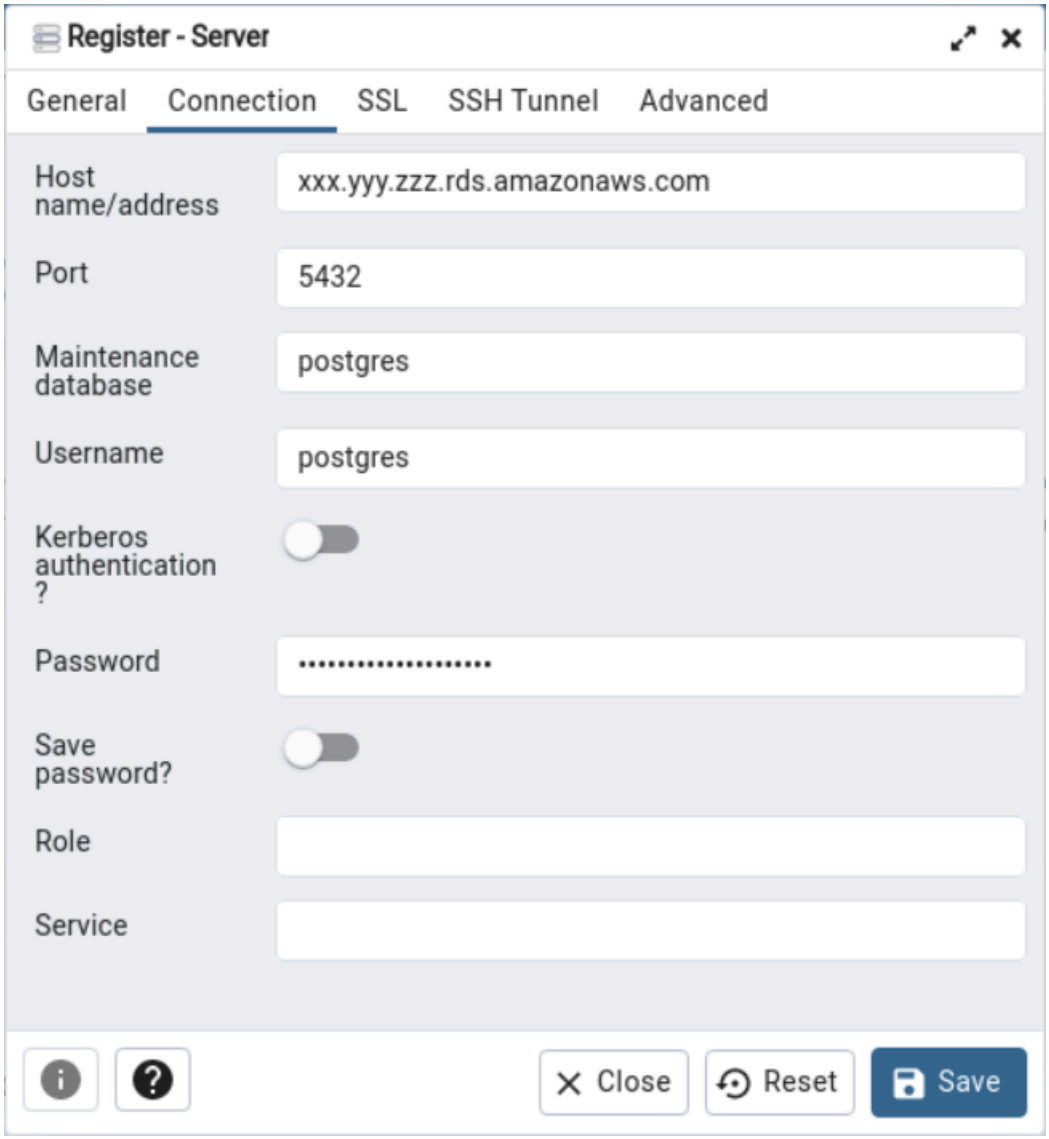
1. Dari instance AppStream 2.0, keluarkan perintah berikut di terminal untuk diluncurkanpgAdmin:

   ```
   ./pgadmin-start.sh
   ```

2. Pilih alamat email dan kata sandi sebagai pengidentifikasi login. Catat URL yang disediakan (biasanya http://127.0.0.1:5050). Luncurkan Google Chrome di instance, salin dan tempel URL ke browser, dan masuk dengan pengenal Anda.



3. Setelah Anda masuk, pilih Tambahkan Server Baru dan masukkan informasi koneksi ke database yang dibuat sebelumnya sebagai berikut.

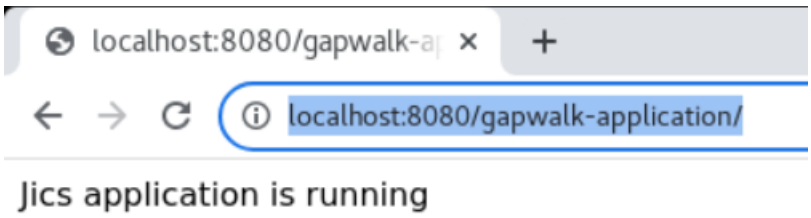


4. Saat Anda terhubung ke server database, gunakan Object > Create > Database dan buat database baru bernama jics.
5. Edit informasi koneksi database yang digunakan aplikasi demo. Informasi ini didefinisikan dalam`PlanetsDemo-runtime/config/application-main.yml`. Cari `jicsDs` entri. Untuk mengambil nilai untuk `username` dan`password`, di konsol Amazon RDS, navigasikan ke database. Pada tab Konfigurasi, di bawah Master Credentials ARN, pilih Manage in Secrets Manager. Kemudian, di konsol Secrets Manager, dalam rahasia, pilih Ambil nilai rahasia.

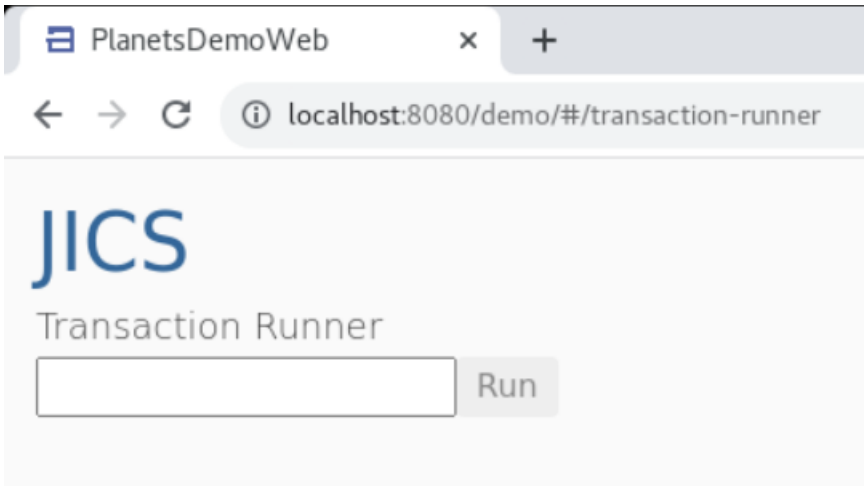
Langkah 9: Mulai dan uji aplikasi

Pada langkah ini, Anda memulai server Tomcat dan aplikasi demo sehingga Anda dapat mengujinya.

1. Untuk memulai server Tomcat dan aplikasi yang digunakan sebelumnya, pilih entri server di tampilan Server dan pilih Mulai. Konsol muncul yang menampilkan log startup.
2. Periksa status server di tampilan Server, atau tunggu server dimulai dalam pesan milidetik [xxx] di konsol. Setelah server dimulai, periksa apakah aplikasi gapwalk digunakan dengan benar. Untuk melakukan ini, akses URL http://localhost:8080/gapwalk-application di browser Google Chrome. Anda harus melihat yang berikut ini.



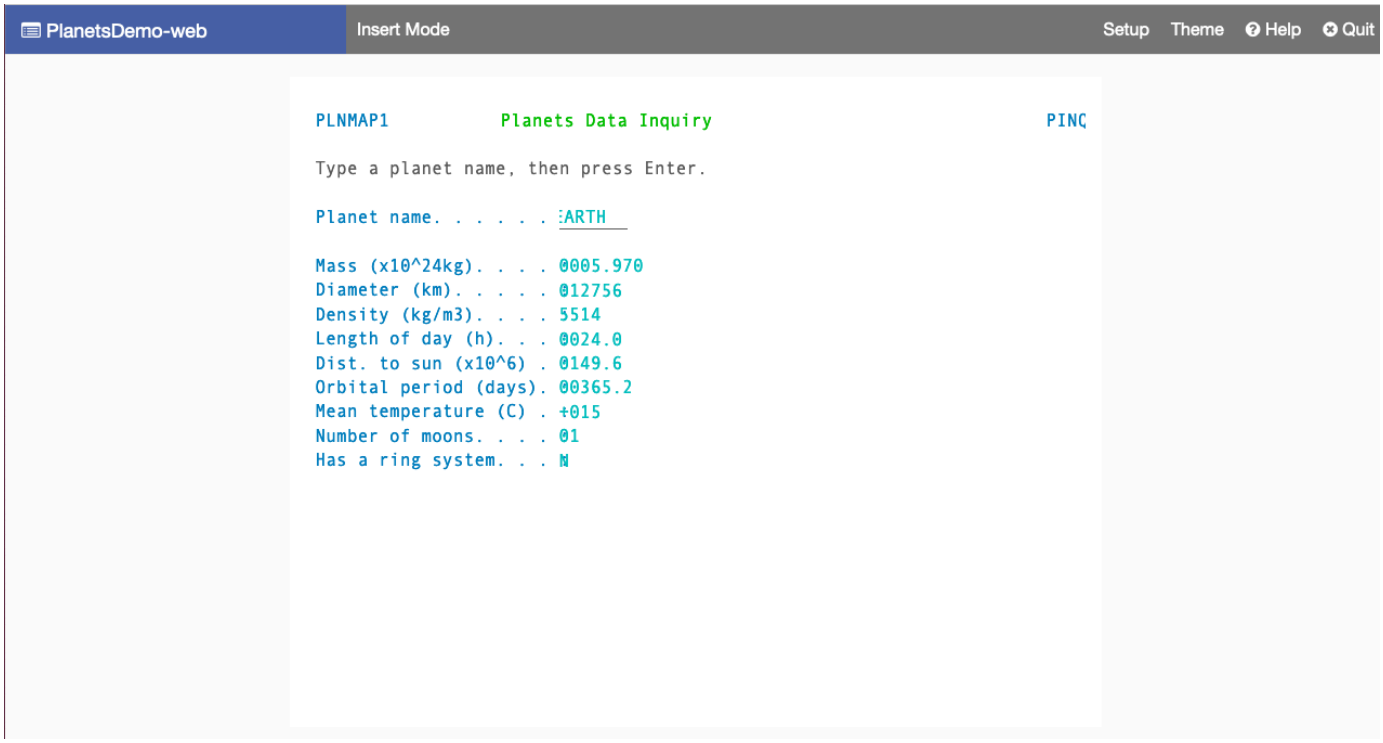
3. Akses frontend aplikasi yang diterapkan dari Google Chrome di http://localhost:8080/demo. Halaman Transaction Launcher berikut akan muncul.



4. Untuk memulai transaksi aplikasi, masukkan `PINQ` di kolom input, dan pilih Jalankan (atau tekan Enter).

   Layar aplikasi demo akan muncul.

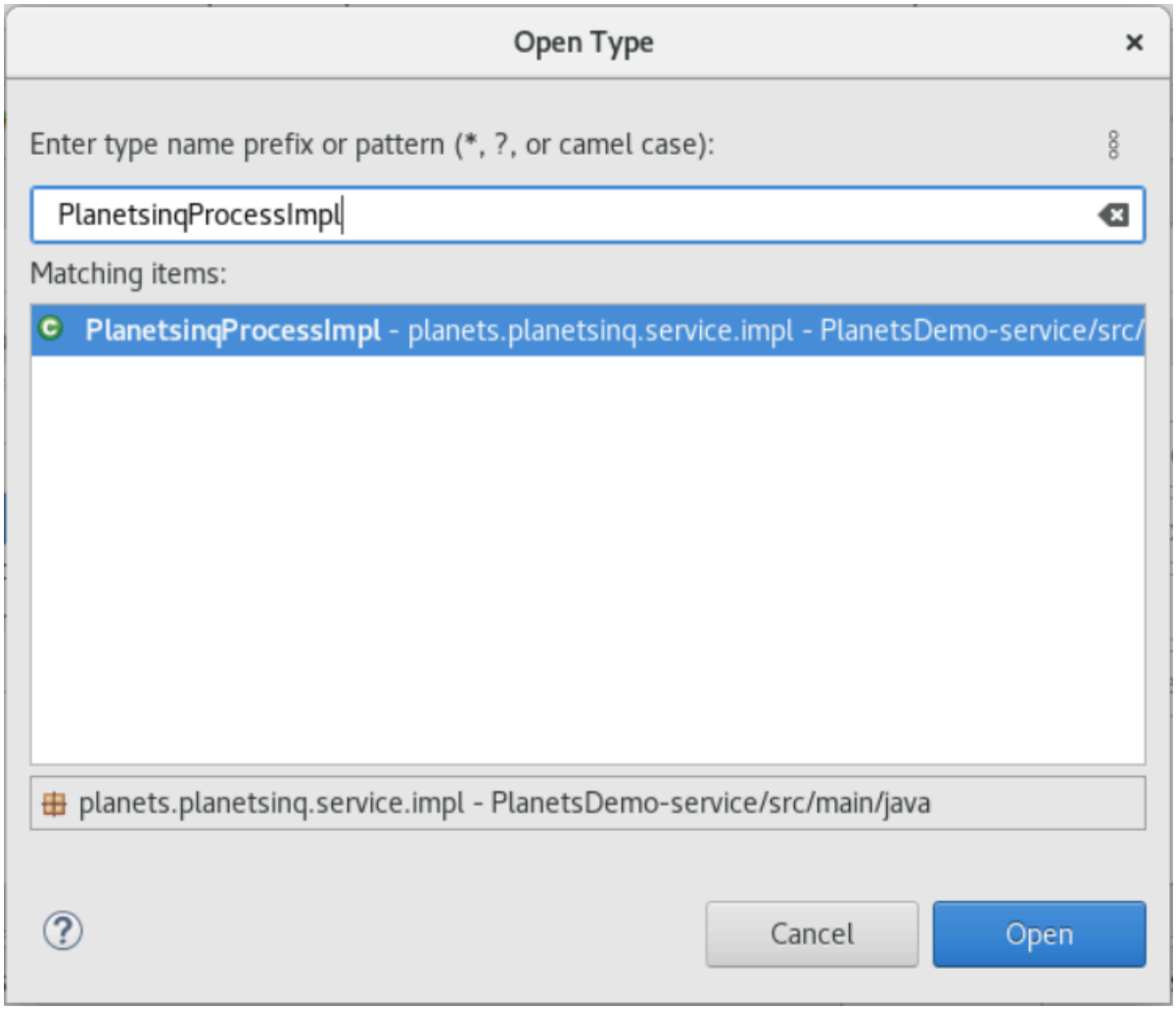
5. Ketik nama planet di bidang yang sesuai dan tekan Enter.



Langkah 10: Debug aplikasi

Pada langkah ini, Anda menguji menggunakan fitur debugging Eclipse standar. Fitur-fitur ini tersedia saat Anda mengerjakan aplikasi modern.

1. Untuk membuka kelas layanan utama, tekan Ctrl+Shift+T. Lalu masukkan. `PlanetsinqProcessImpl`



2. Arahkan ke `searchPlanet` metode, dan letakkan breakpoint di sana.
3. Pilih nama server dan pilih Restart di Debug.
4. Ulangi langkah sebelumnya. Artinya, akses aplikasi, masukkan nama planet, dan tekan Enter.

   Eclipse akan menghentikan aplikasi dalam metode ini`searchPlanet`. Sekarang Anda bisa memeriksanya.

Pembersihan sumber daya

Jika Anda tidak lagi membutuhkan sumber daya yang Anda buat untuk tutorial ini, hapus sehingga Anda tidak dikenakan biaya tambahan. Selesaikan langkah-langkah berikut:

- Jika aplikasi Planets masih berjalan, hentikan.
- Hapus database yang Anda buatLangkah 1: Buat database. Untuk informasi selengkapnya, lihat Menghapus instans DB.

Membentuk ulang aplikasi dengan Micro Focus

Bagian ini menjelaskan setiap langkah dalam proses replatforming. Ini menjelaskan semua tugas dan mencakup informasi tentang mengonfigurasi dan mengoperasikan runtime Modernisasi AWS Mainframe di Amazon EC2.

Topik

Pengaturan Waktu Proses Fokus Mikro (di Amazon EC2)

AWS Modernisasi Mainframe menyediakan beberapa Amazon Machine Images (AMI) yang mencakup produk berlisensi Micro Focus. AMI ini memungkinkan Anda menyediakan instans Amazon Elastic Compute Cloud (Amazon EC2) dengan cepat untuk mendukung lingkungan Fokus Mikro yang Anda kontrol dan kelola. Topik ini memberikan langkah-langkah yang diperlukan untuk mengakses dan meluncurkan AMI ini. Menggunakan AMI ini sepenuhnya opsional dan mereka tidak diharuskan untuk menyelesaikan tutorial dalam panduan pengguna ini.

Topik

- Menciptakan AWS Identity and Access Management peran
- Berikan License Manager izin yang diperlukan
- Berlangganan Gambar Mesin Amazon
- Luncurkan Instans AWS Mikro Fokus Modernisasi Mainframe
- Subnet atau VPC tanpa Akses Internet
- Memecahkan masalah lisensi

Prasyarat

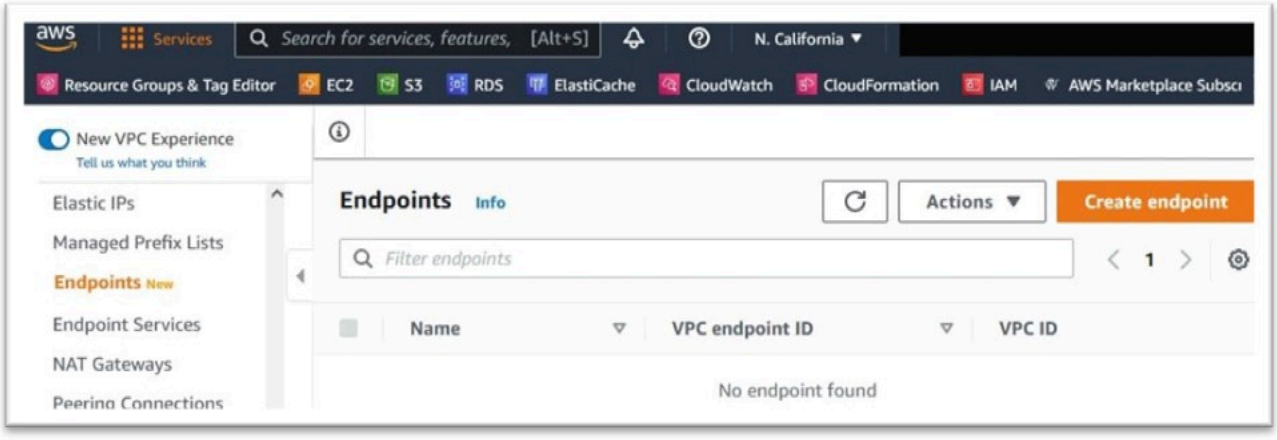
Pastikan Anda memenuhi prasyarat berikut.

- Akses administrator ke akun tempat instans Amazon EC2 akan dibuat.
- Identifikasi Wilayah AWS tempat instans Amazon EC2 akan dibuat dan verifikasi layanan Modernisasi AWS Mainframe tersedia. Lihat AWS Layanan berdasarkan Wilayah. Pastikan untuk memilih Wilayah di mana layanan tersedia.
- Identifikasi Amazon Virtual Private Cloud (Amazon VPC) tempat instans Amazon EC2 akan dibuat.

Buat Titik Akhir VPC Amazon untuk Amazon S3

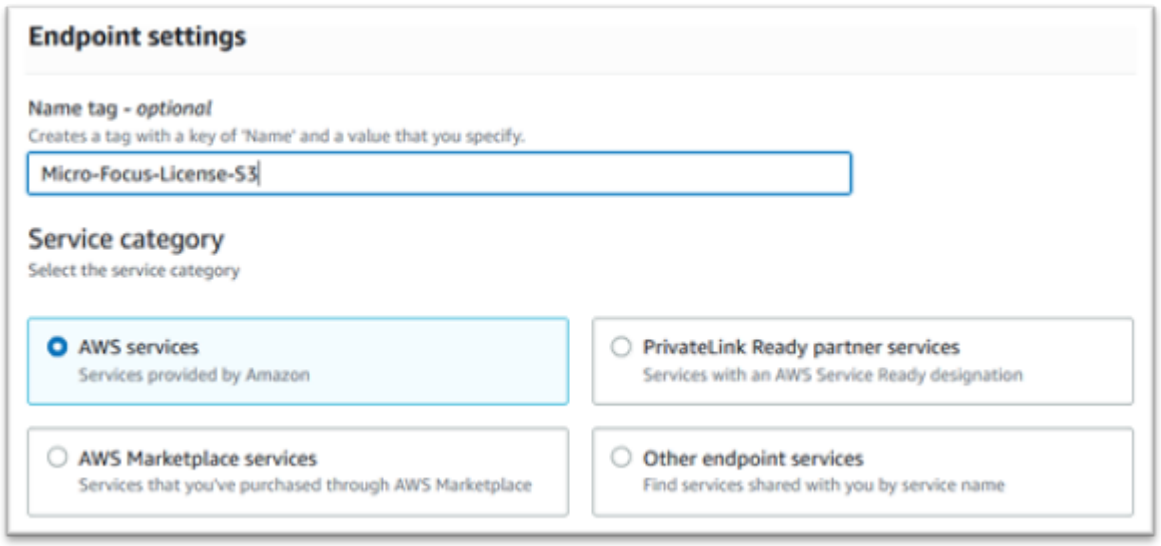
Di bagian ini, Anda membuat titik akhir Amazon VPC untuk Amazon S3 untuk digunakan.

1. Arahkan ke Amazon VPC di. AWS Management Console
2. Di panel navigasi, pilih Titik Akhir.
3. Pilih Buat Titik Akhir.



4. Masukkan tag nama yang berarti, misalnya: “Micro-Focus-License-S3".

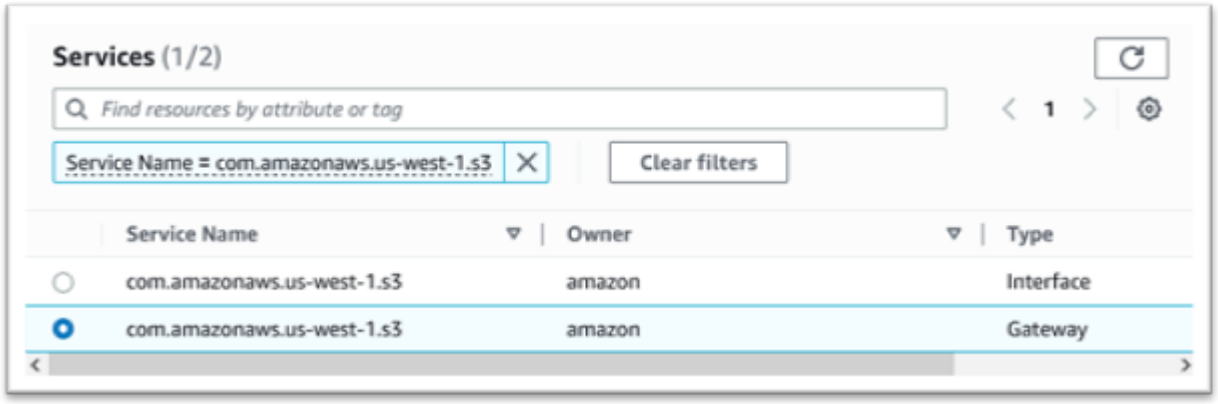
5. Pilih AWS Services sebagai Kategori Layanan.



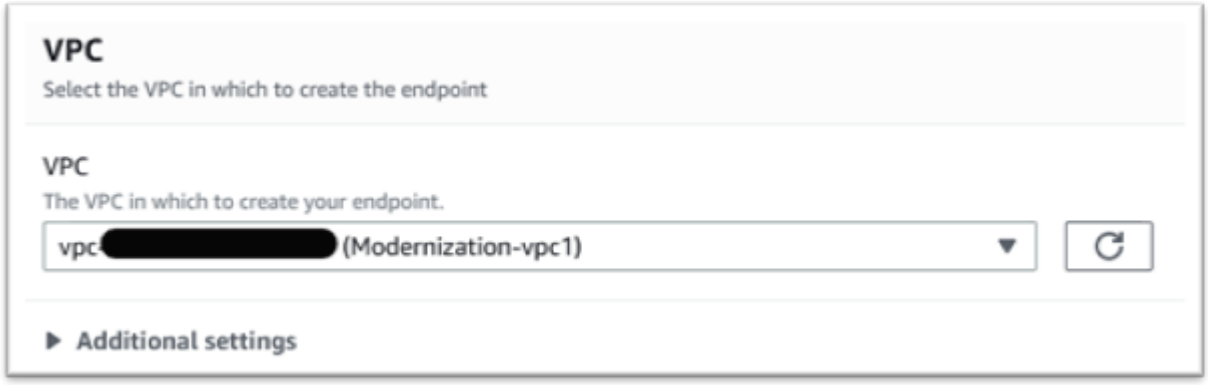
6. Di bawah Layanan, cari layanan Amazon S3 Gateway: com.amazonaws. [wilayah] .s3.

   Untuk `us-west-1` ini akan menjadi:`com.amazonaws.us-west-1.s3`.

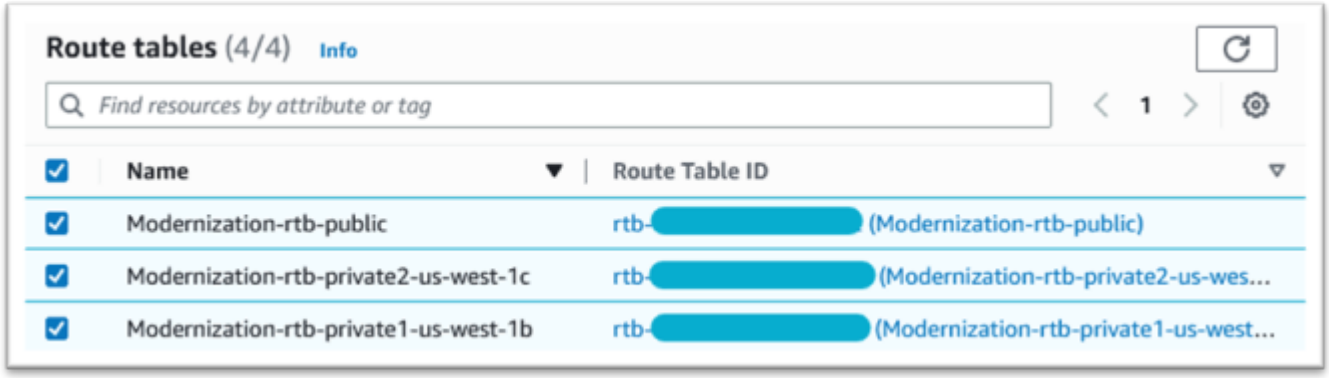
7. Pilih layanan Gateway.



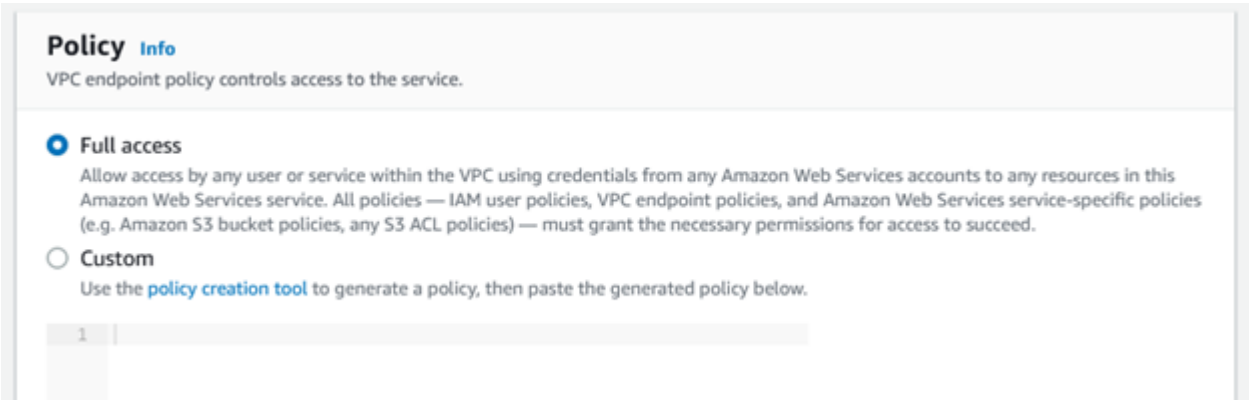
8. Untuk VPC pilih VPC yang akan Anda gunakan.



9. Pilih semua tabel rute untuk VPC.



10. Di bawah Kebijakan pilih Akses Penuh.



11. Pilih Buat Titik Akhir.

Minta pembaruan Daftar Izin untuk Akun

Bekerja dengan AWS perwakilan Anda agar akun Anda diizinkan terdaftar untuk AMI Modernisasi AWS Mainframe. Harap berikan informasi berikut:

- Akun AWS ID.
- Wilayah AWS Tempat titik akhir VPC Amazon dibuat.
- ID titik akhir Amazon VPC Amazon S3 Amazon dibuat di. Buat Titik Akhir VPC Amazon untuk Amazon S3 Ini adalah `vpce-xxxxxxxxxxxxxxxxx` id untuk com.amazonaws. [wilayah] .s3 Titik akhir Gateway.
- Jumlah lisensi yang diperlukan di semua instans Micro Focus Enterprise Suite AMI Amazon EC2.

  Satu lisensi diperlukan per inti CPU (per 2 vCPU untuk sebagian besar instans Amazon EC2).

  Untuk informasi selengkapnya, lihat Optimalkan opsi CPU.

Nomor yang diminta dapat disesuaikan di masa depan oleh AWS.

> **Note**
>
> AWS Perwakilan harus membuka tiket dukungan untuk permintaan Allowlist. Itu tidak dapat diminta secara langsung dan permintaan mungkin memakan waktu beberapa hari untuk diselesaikan.

Menciptakan AWS Identity and Access Management peran

Buat AWS Identity and Access Management kebijakan dan peran yang akan digunakan oleh instans Amazon EC2 Modernisasi AWS Mainframe. Membuat peran melalui konsol IAM akan membuat profil instance terkait dengan nama yang sama. Menetapkan profil instans ini ke instans Amazon EC2 memungkinkan Lisensi Fokus Mikro ditetapkan. Untuk informasi selengkapnya tentang profil instans, lihat Menggunakan peran IAM untuk memberikan izin ke aplikasi yang berjalan di instans Amazon EC2.

Buat Kebijakan IAM

Kebijakan IAM dibuat terlebih dahulu dan kemudian dilampirkan pada peran tersebut.

1. Arahkan ke AWS Identity and Access Management dalam AWS Management Console.
2. Pilih Kebijakan dan kemudian Buat Kebijakan.

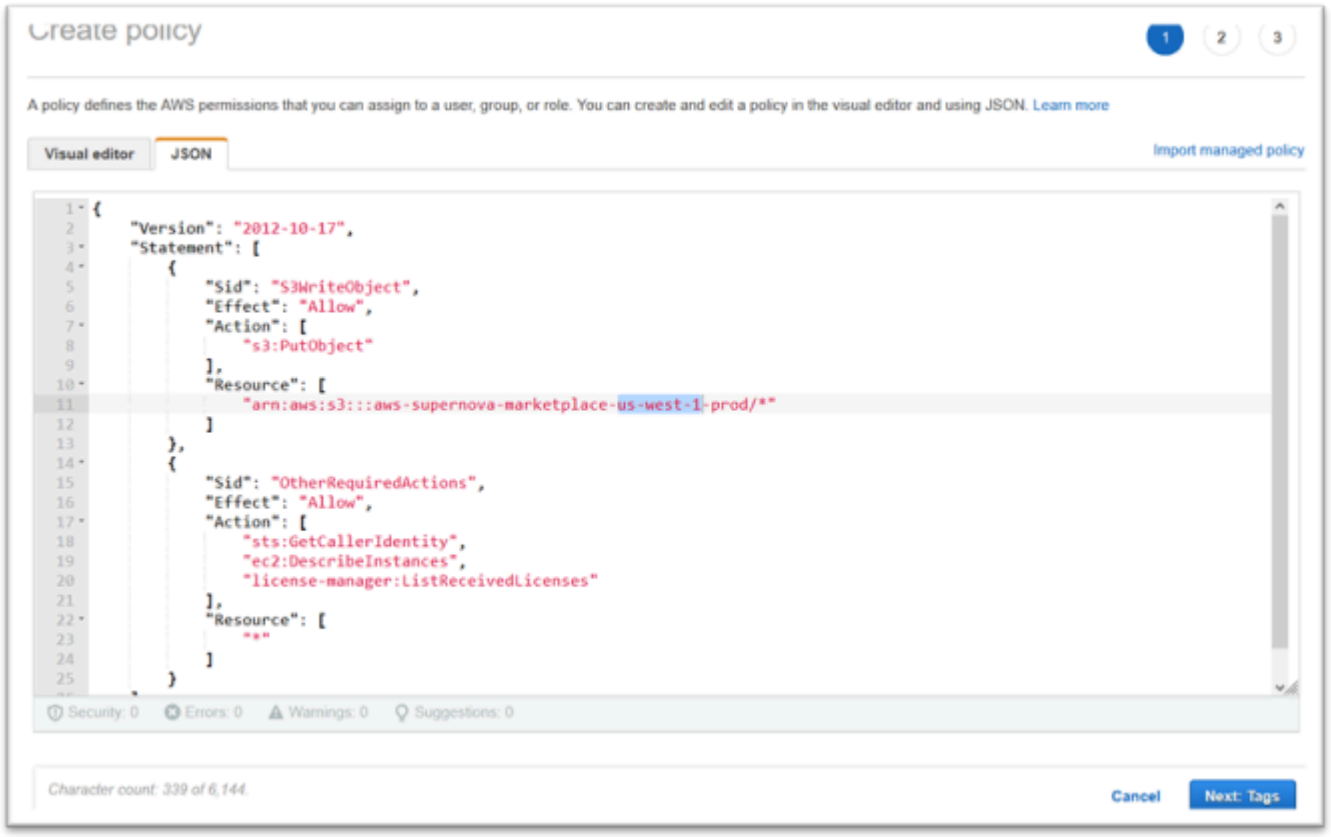
3. Pilih tab JSON.

4. Ganti `us-west-1` di JSON berikut dengan Wilayah AWS tempat titik akhir Amazon S3 ditentukan, lalu salin dan tempel JSON ke editor kebijakan.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3WriteObject",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::aws-supernova-marketplace-us-west-1-prod/*"
            ]
        },
        {
            "Sid": "OtherRequiredActions",
            "Effect": "Allow",
            "Action": [
                "sts:GetCallerIdentity",
                "ec2:DescribeInstances",
                "license-manager:ListReceivedLicenses"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
```

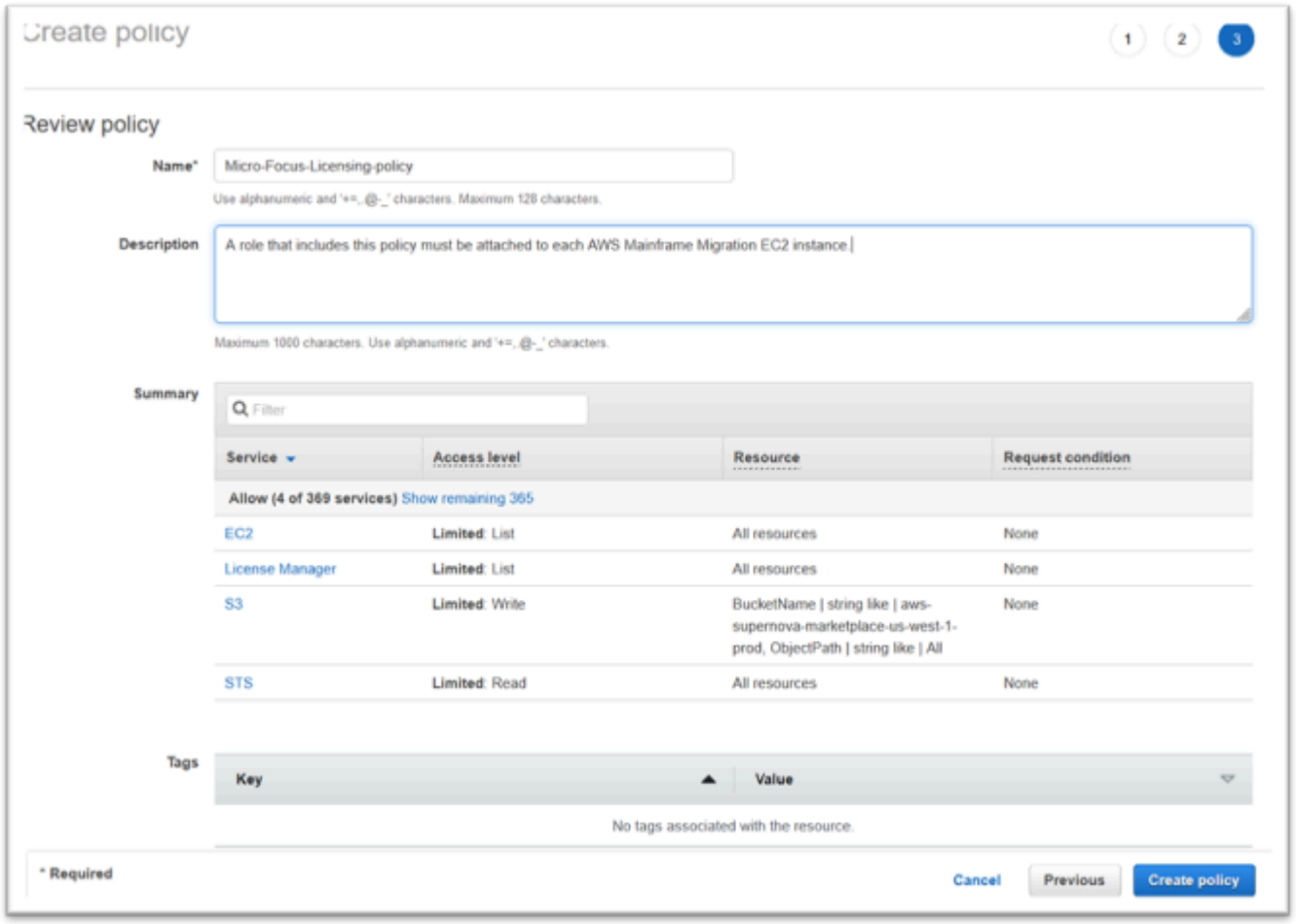
```
}
```

> **Note**
>
> Tindakan di bawah Sid `OtherRequiredActions` tidak mendukung izin tingkat sumber daya dan harus menentukan * dalam elemen sumber daya.



5. Pilih Berikutnya: Tanda.

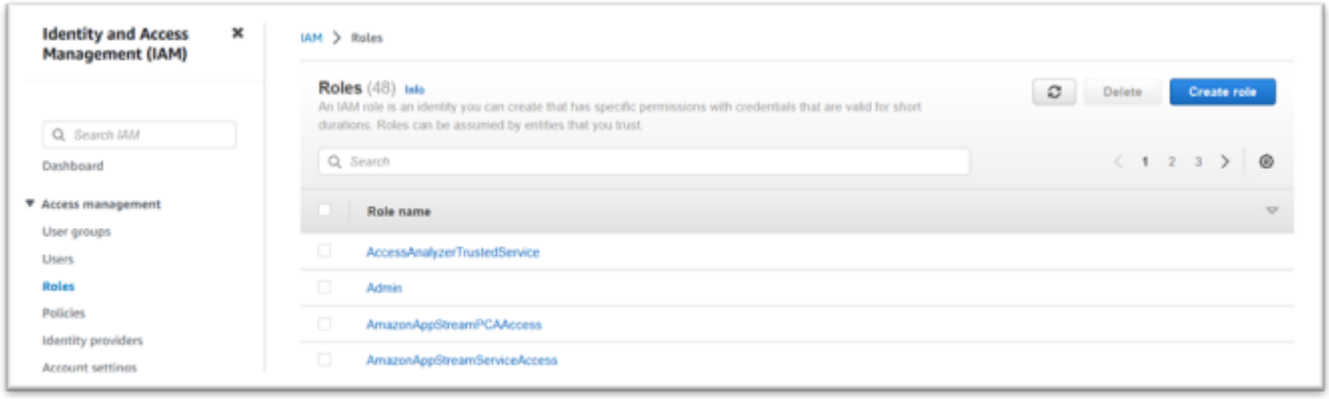
6. Secara opsional masukkan tag apa pun, lalu pilih Berikutnya: Tinjau.
7. Masukkan nama untuk kebijakan tersebut, misalnya “Micro-focus-Licensing-Policy”. Secara opsional masukkan deskripsi, misalnya “Peran yang menyertakan kebijakan ini harus dilampirkan ke setiap instance Modernisasi Amazon EC2 AWS Mainframe.”



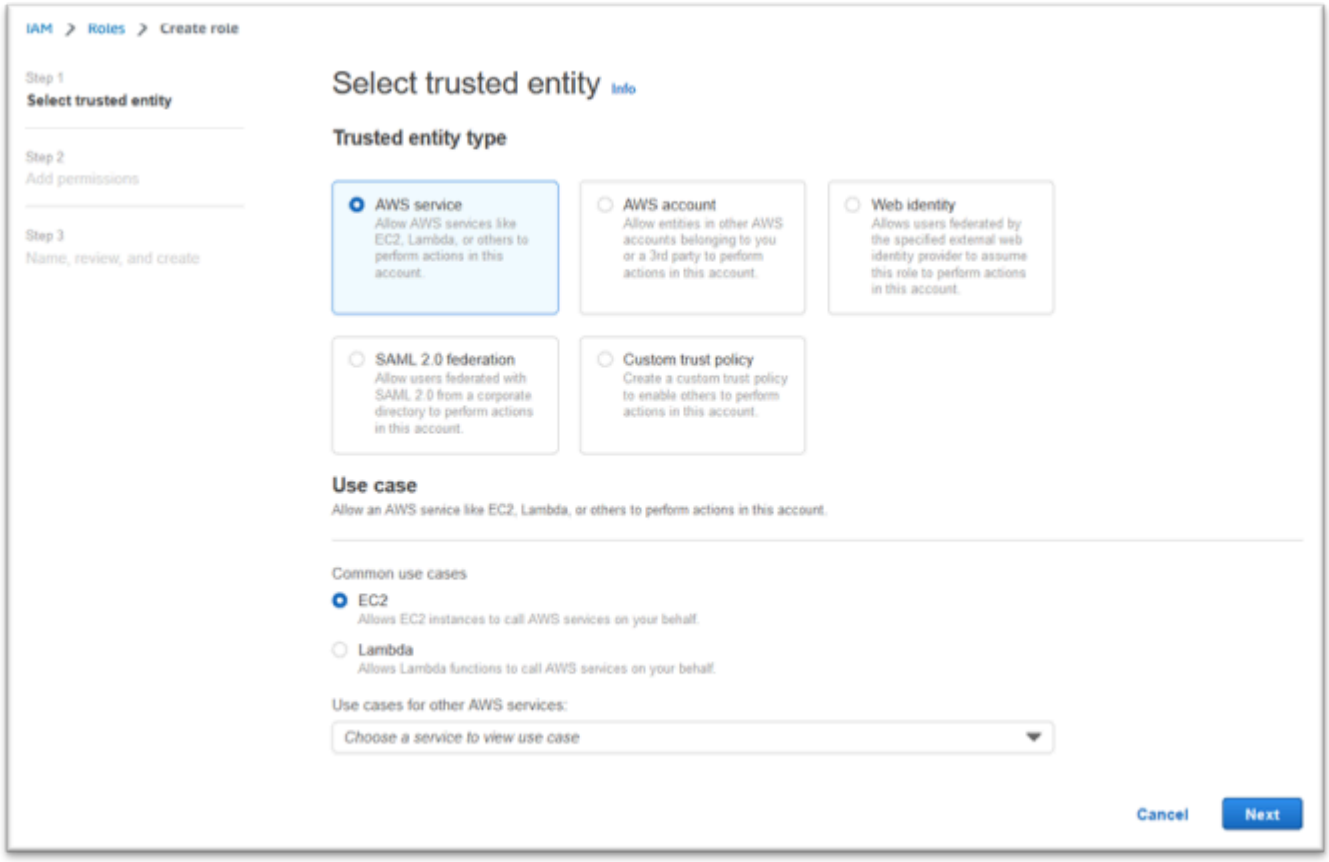
8. Pilih Buat Kebijakan.

Buat Peran IAM

1. Arahkan ke IAM di. AWS Management Console
2. Pilih Peran dan kemudian Buat Peran.

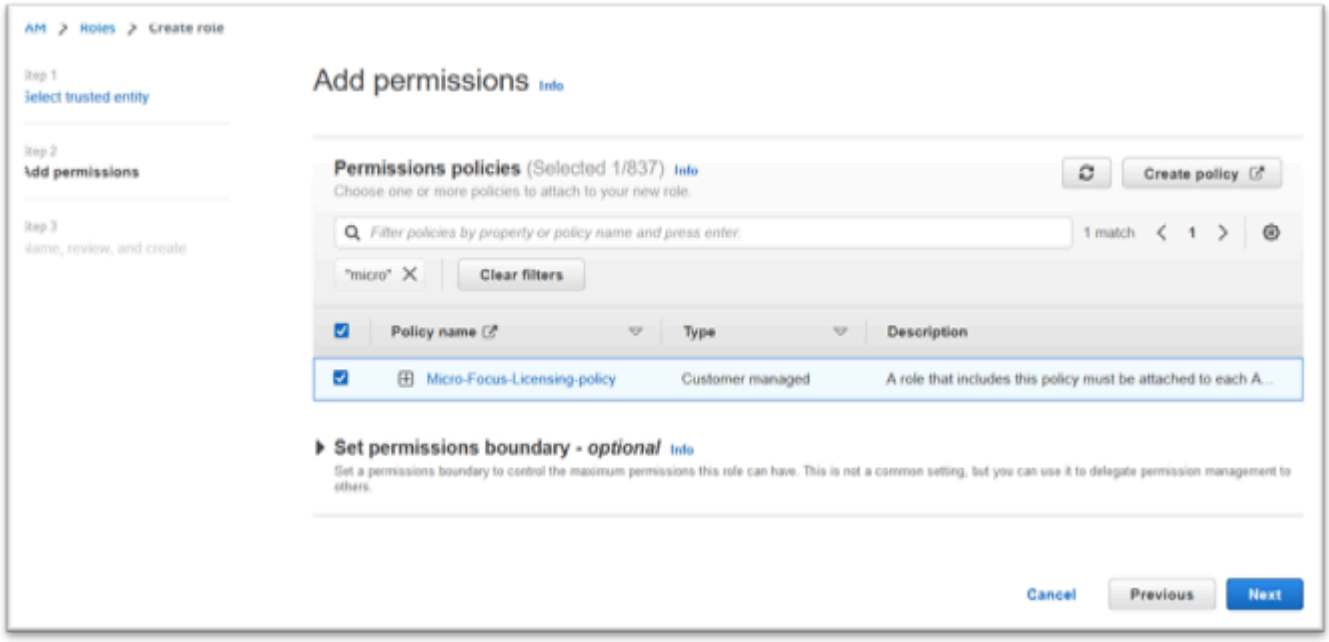


3. Tinggalkan jenis entitas Tepercaya sebagai AWS layanan dan pilih kasus penggunaan umum EC2.

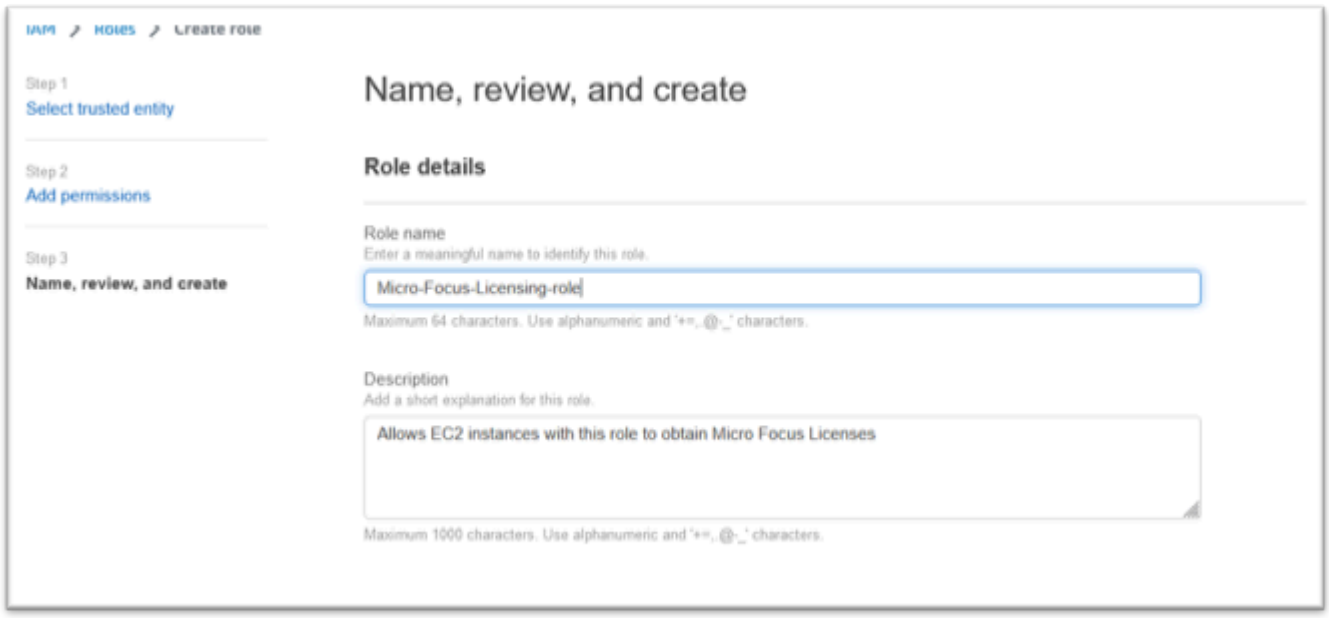


4. Pilih Berikutnya.
5. Masukkan “Mikro” ke dalam filter dan tekan enter untuk menerapkan filter.

6. Pilih kebijakan yang baru saja dibuat, misalnya “Micro-focus-Licensing-Policy”.
7. Pilih Berikutnya.



8. Masukkan nama Peran, misalnya “Micro-focus-Licensing-Role”.
9. Ganti deskripsi dengan salah satu deskripsi Anda sendiri, misalnya “Mengizinkan instans Amazon EC2 dengan peran ini untuk mendapatkan Lisensi Fokus Mikro”.

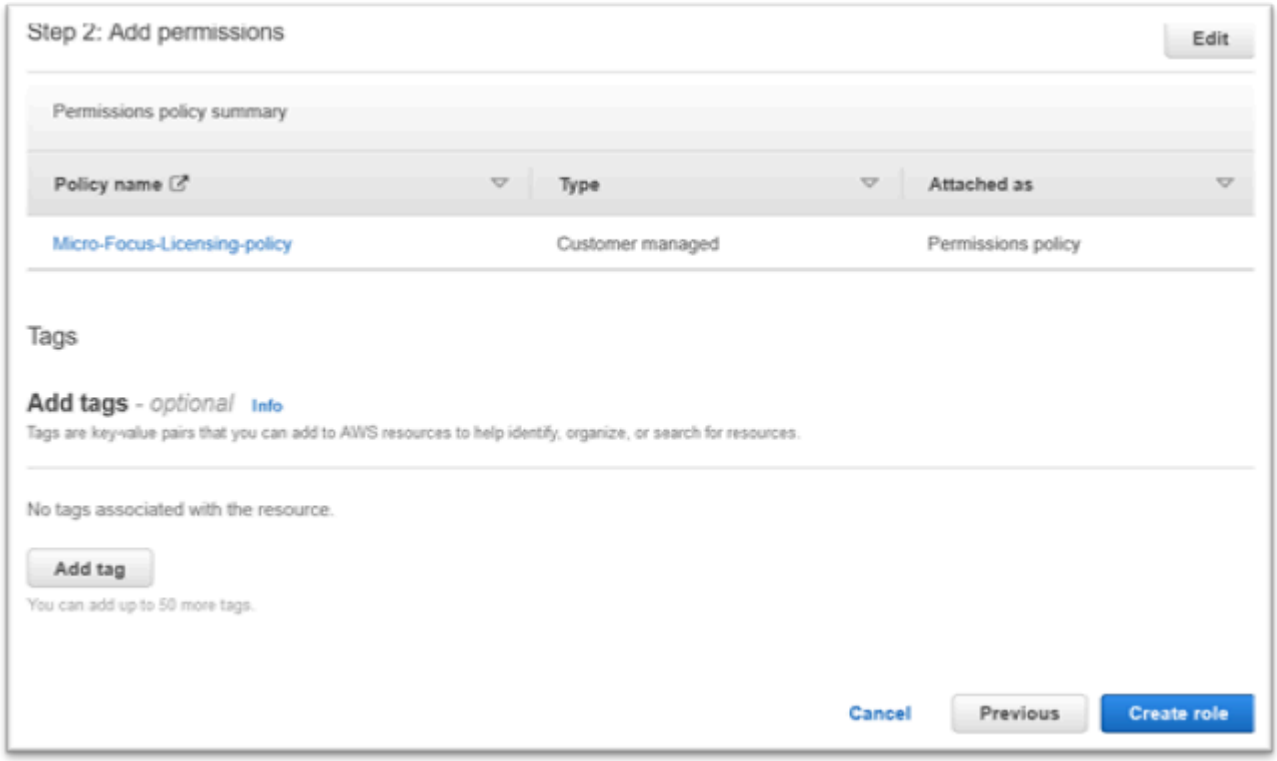


10. Di bawah Langkah 1: Pilih entitas tepercaya, tinjau JSON dan konfirmasikan bahwa JSON memiliki nilai berikut:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Principal": {
                "Service": [
                    "ec2.amazonaws.com"
                ]
            }
        }
    ]
}
```

> **Note**
>
> Urutan Efek, Tindakan, dan Prinsipal tidak signifikan.

11. Konfirmasikan bahwa Langkah 2: Tambahkan izin menunjukkan kebijakan Lisensi Anda.

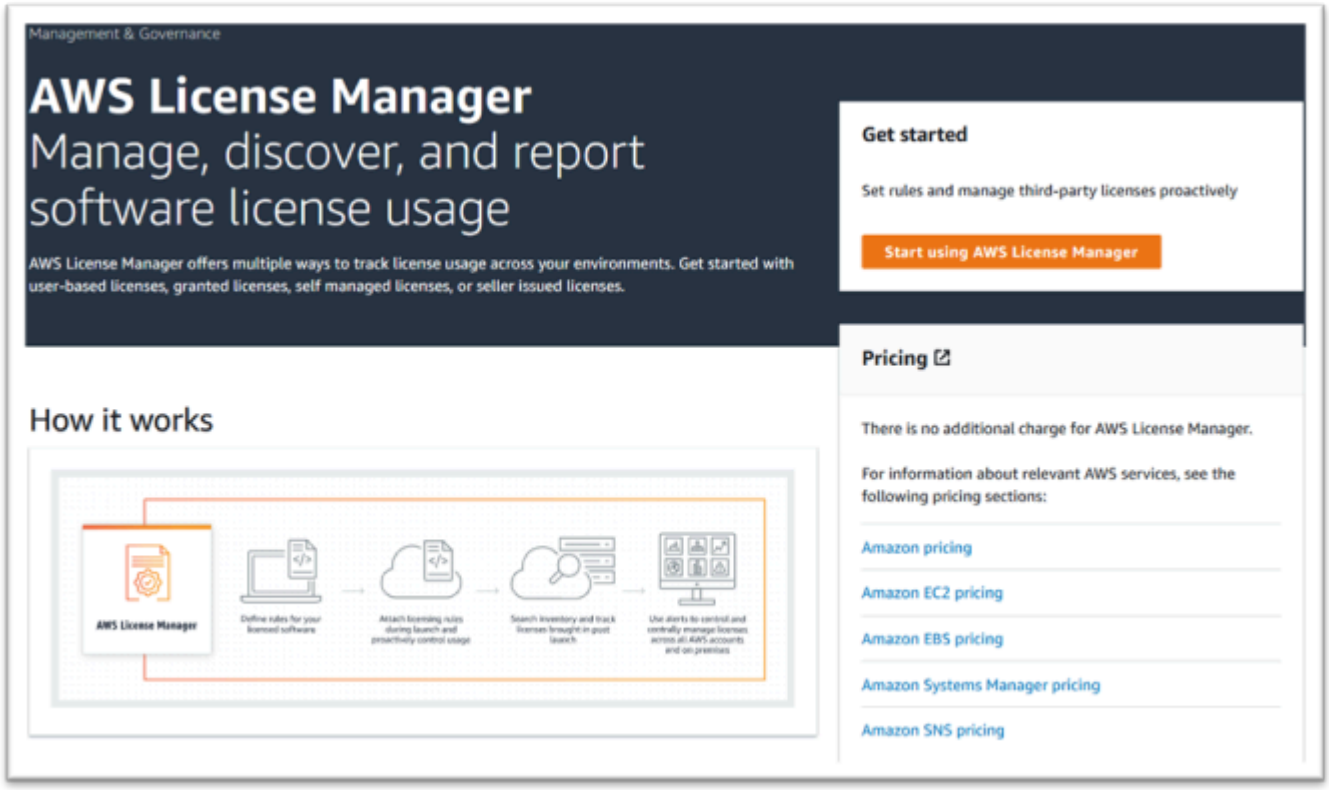


12. Pilih Buat peran.

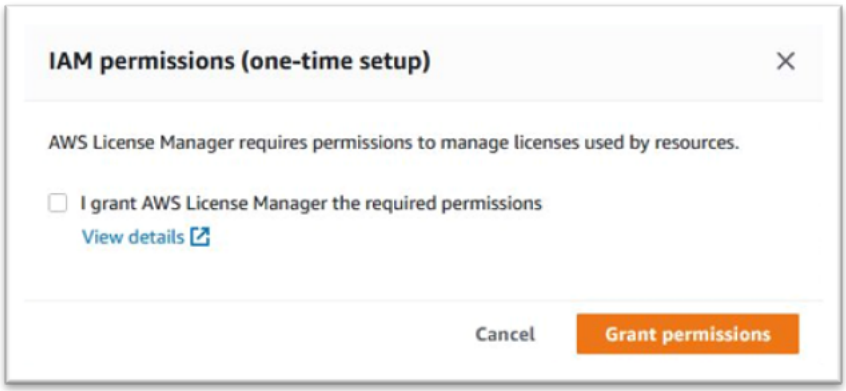
Setelah permintaan allowlist selesai, lanjutkan dengan langkah-langkah berikut.

Berikan License Manager izin yang diperlukan

1. Arahkan ke AWS License Manager dalam AWS Management Console.



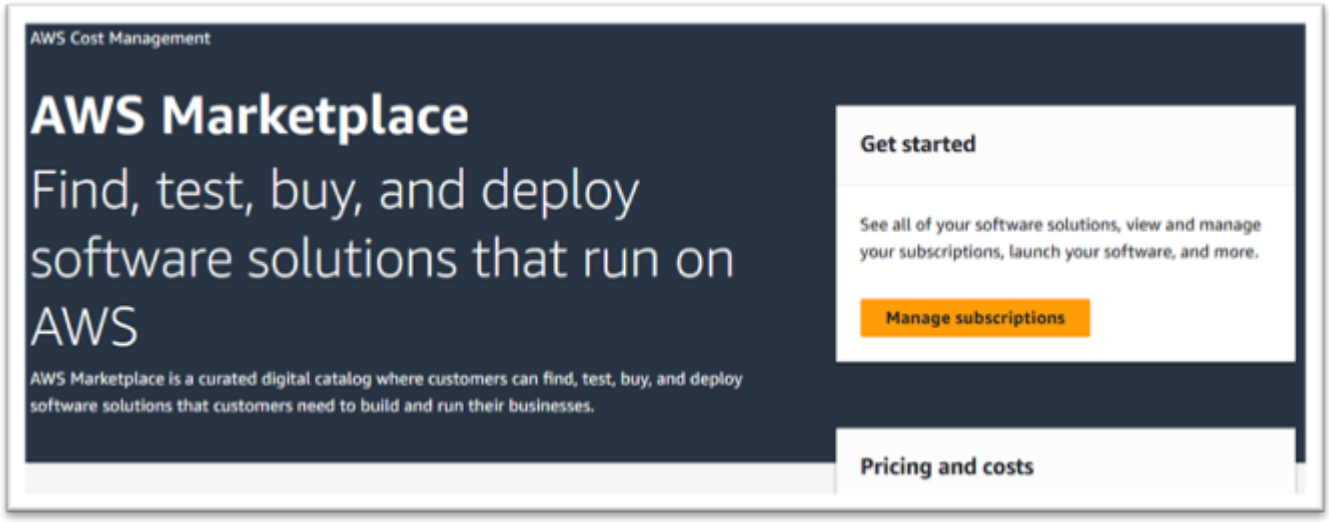
2. Pilih Mulai menggunakan AWS License Manager.
3. Jika Anda melihat pop-up berikut, lihat detailnya, lalu pilih kotak centang dan tekan Hibah Izin.



Berlangganan Gambar Mesin Amazon

Setelah Anda berlangganan AWS Marketplace produk, Anda dapat meluncurkan instance dari AMI produk.

1. Arahkan ke AWS Marketplace Langganan di. AWS Management Console
2. Pilih Kelola langganan.



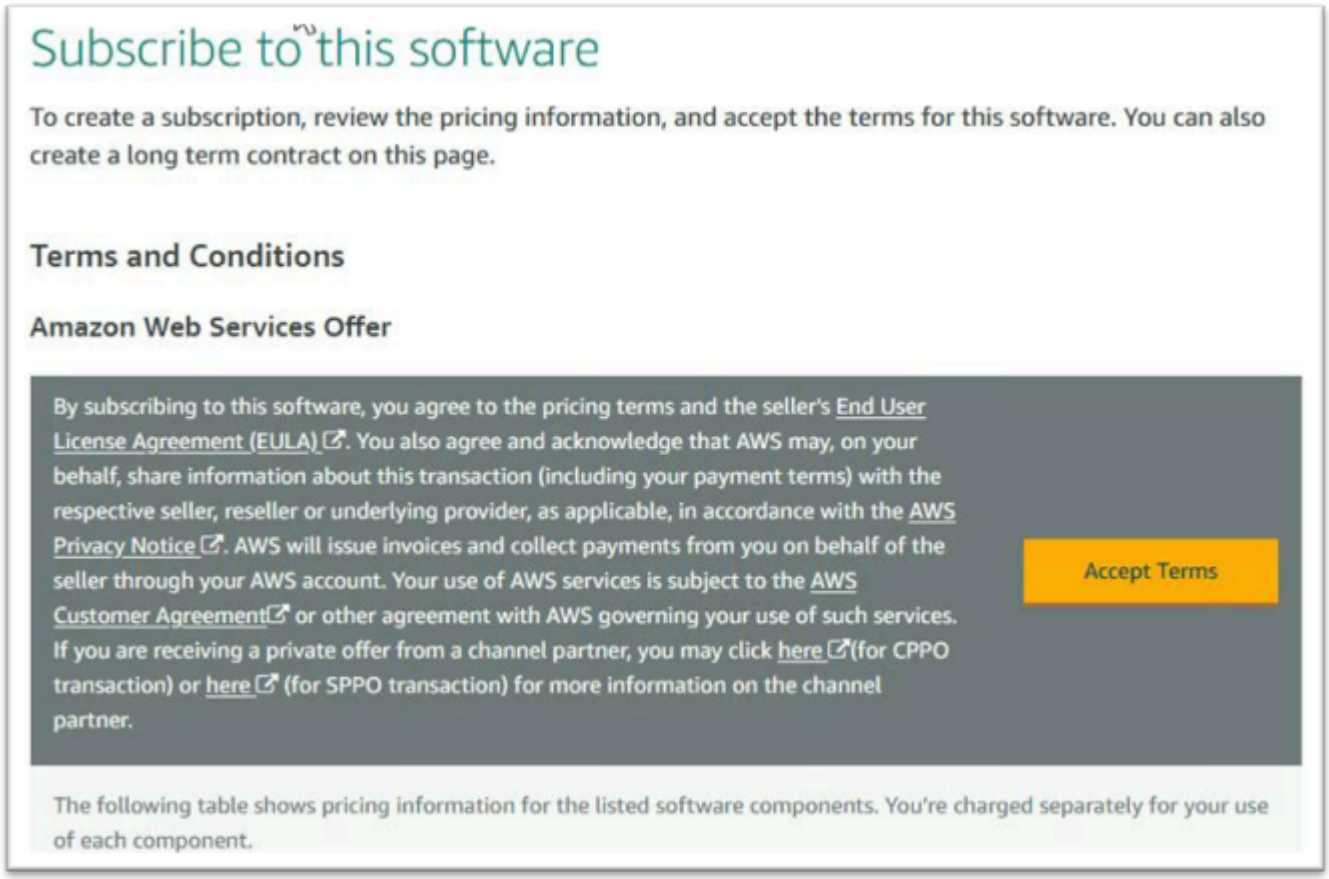
3. Salin dan tempel salah satu tautan berikut ke bilah alamat browser.

> **Note**
>
> Hanya pilih tautan untuk salah satu produk yang telah diizinkan untuk Anda gunakan.

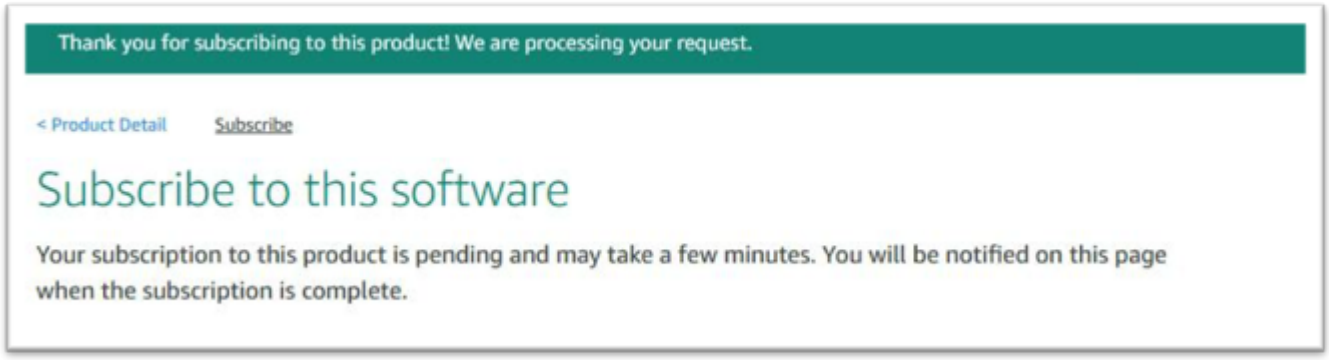
- Server Perusahaan: https://aws.amazon.com/marketplace/pp/prodview-g5emev63l7blc
- Server Perusahaan untuk Windows: https://aws.amazon.com/marketplace/pp/prodview-lwybsiyikbhc2
- Pengembang Perusahaan: https://aws.amazon.com/marketplace/pp/prodview-77qmpr42yzxwk
- Pengembang Perusahaan dengan Visual Studio 2022: https://aws.amazon.com/marketplace/pp/prodview-m4l3lqiszo6cm
- Penganalisis Perusahaan: https://aws.amazon.com/marketplace/pp/prodview-tttheylcmcihm
- Alat Bangun Perusahaan untuk Windows: https://aws.amazon.com/marketplace/pp/prodview-2rw35bbt6uozi
- Prosedur Tersimpan Perusahaan: https://aws.amazon.com/marketplace/pp/prodview-zoeyqnsdsj6ha
- Prosedur Tersimpan Perusahaan dengan SQL Server 2019: https://aws.amazon.com/marketplace/pp/prodview-ynfklquwubnz4

4. Pilih Lanjutkan Berlangganan.

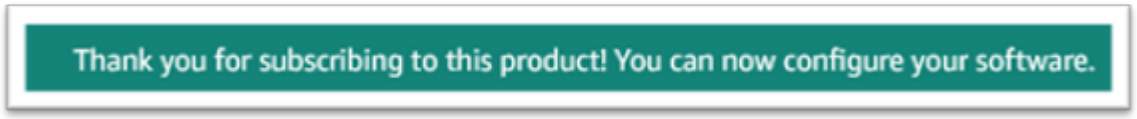
5. Jika Syarat dan Ketentuan dapat diterima, pilih Terima Syarat.



6. Langganan mungkin memakan waktu beberapa menit untuk diproses.

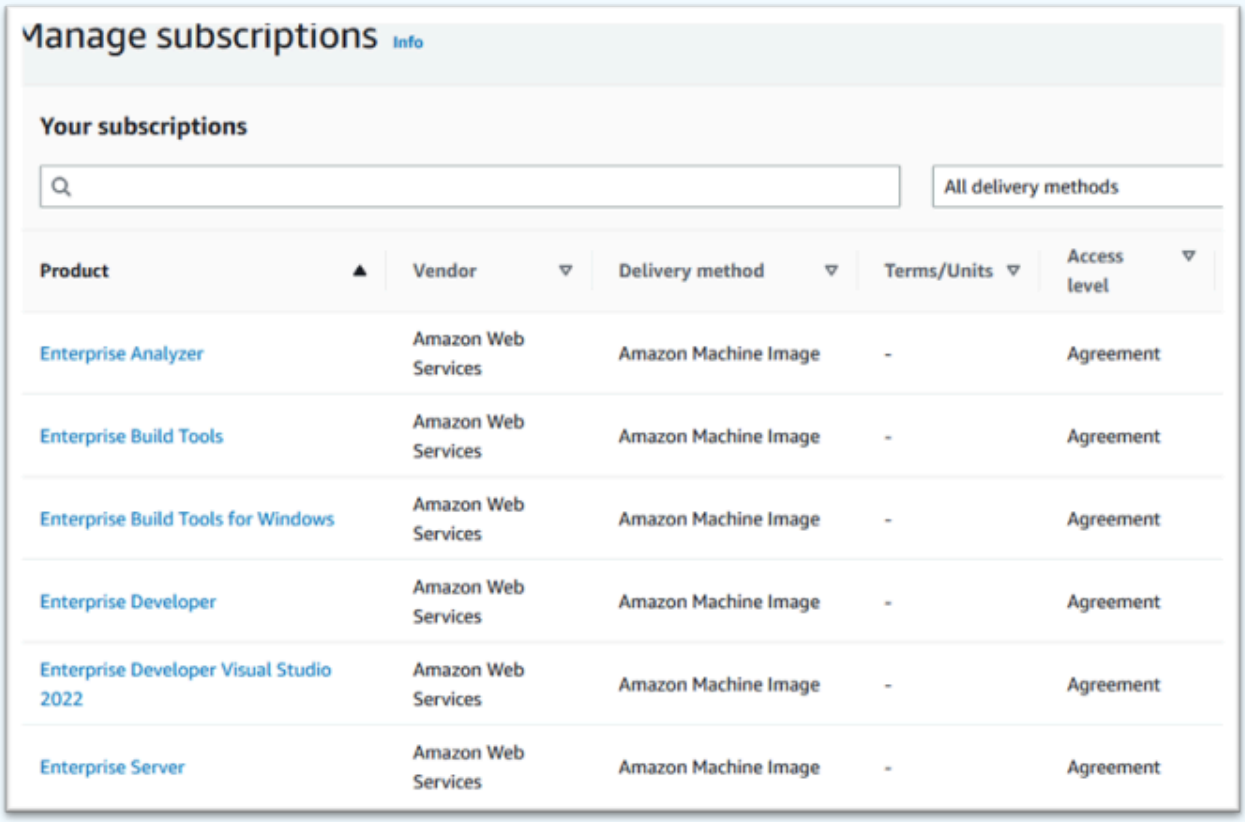


7. Setelah pesan Terima kasih muncul, salin dan tempel tautan berikutnya dari langkah 3 untuk terus menambahkan langganan.



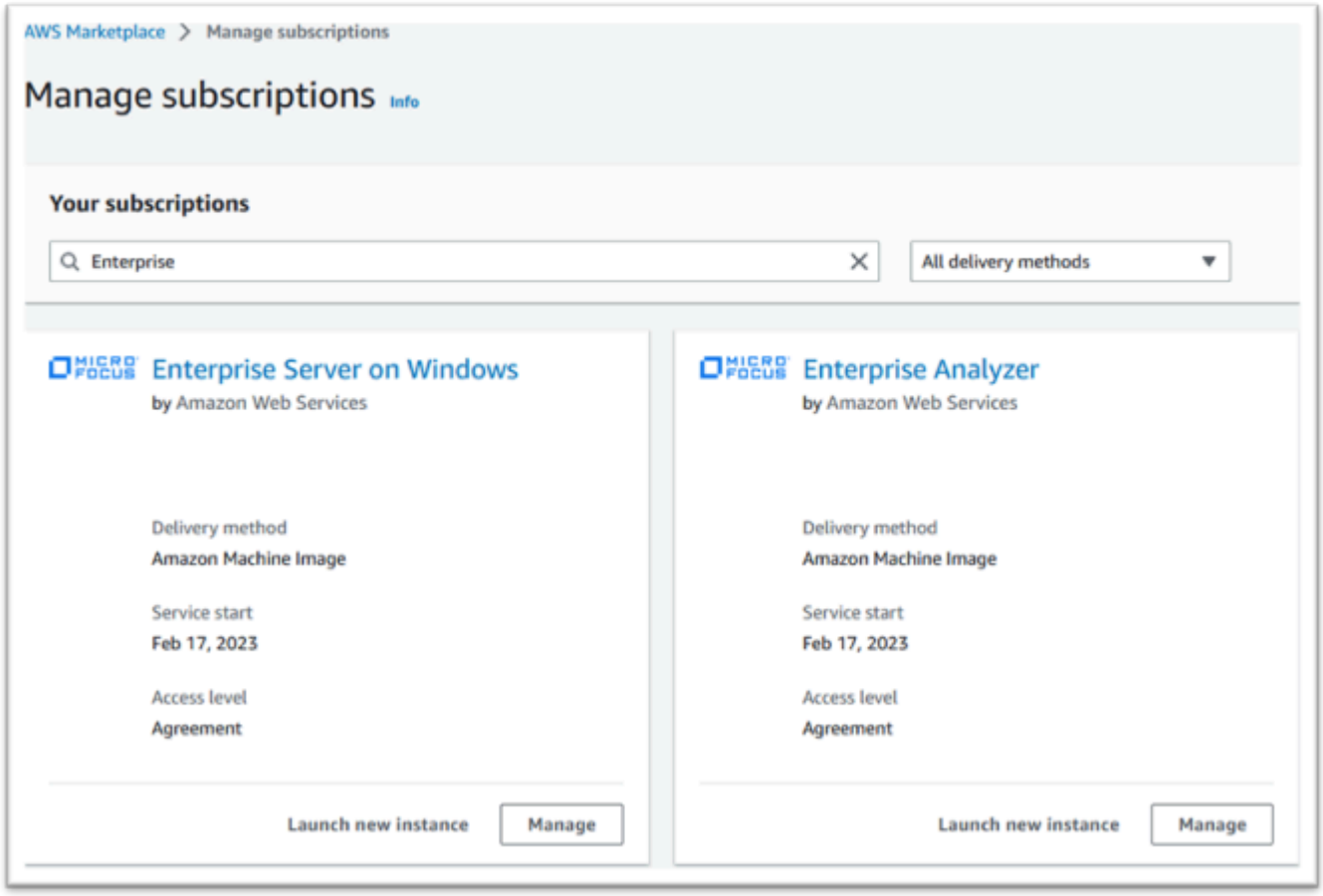
8. Berhenti saat Kelola langganan menampilkan semua AMI berlangganan Anda.

> **Note**
>
> Preferensi panel (ikon roda gigi) diatur untuk menampilkan Tampilan sebagai Tabel.



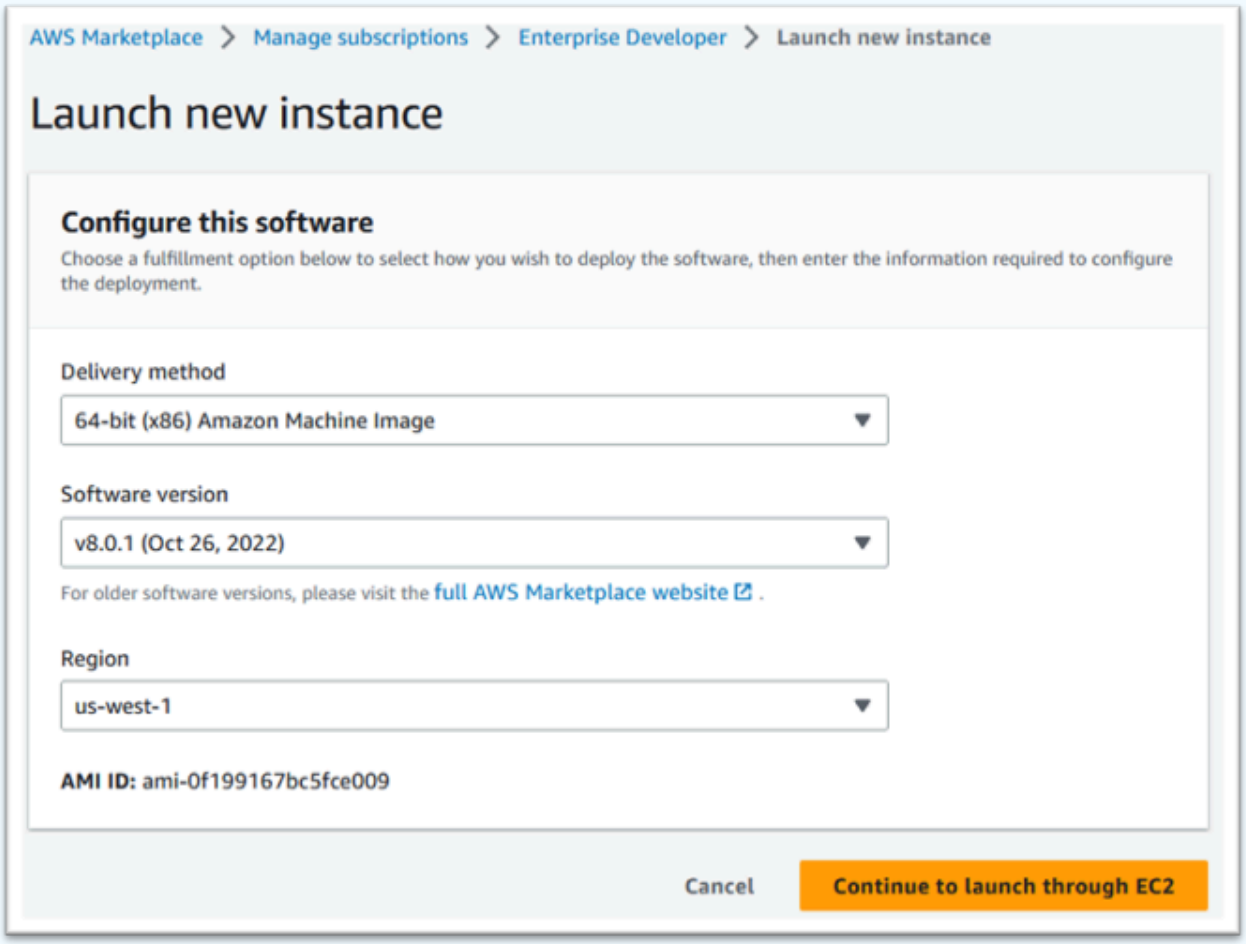
Luncurkan Instans AWS Mikro Fokus Modernisasi Mainframe

1. Arahkan ke AWS Marketplace Langganan di. AWS Management Console
2. Temukan AMI yang akan diluncurkan dan pilih Launch New Instance.



3. Dalam dialog peluncuran instance baru, pastikan wilayah yang diizinkan dipilih.
4. Tekan Lanjutkan untuk meluncurkan melalui EC2.

> **Note**
>
> Contoh berikut menunjukkan peluncuran AMI Pengembang Perusahaan, tetapi prosesnya sama untuk semua AMI Modernisasi AWS Mainframe.

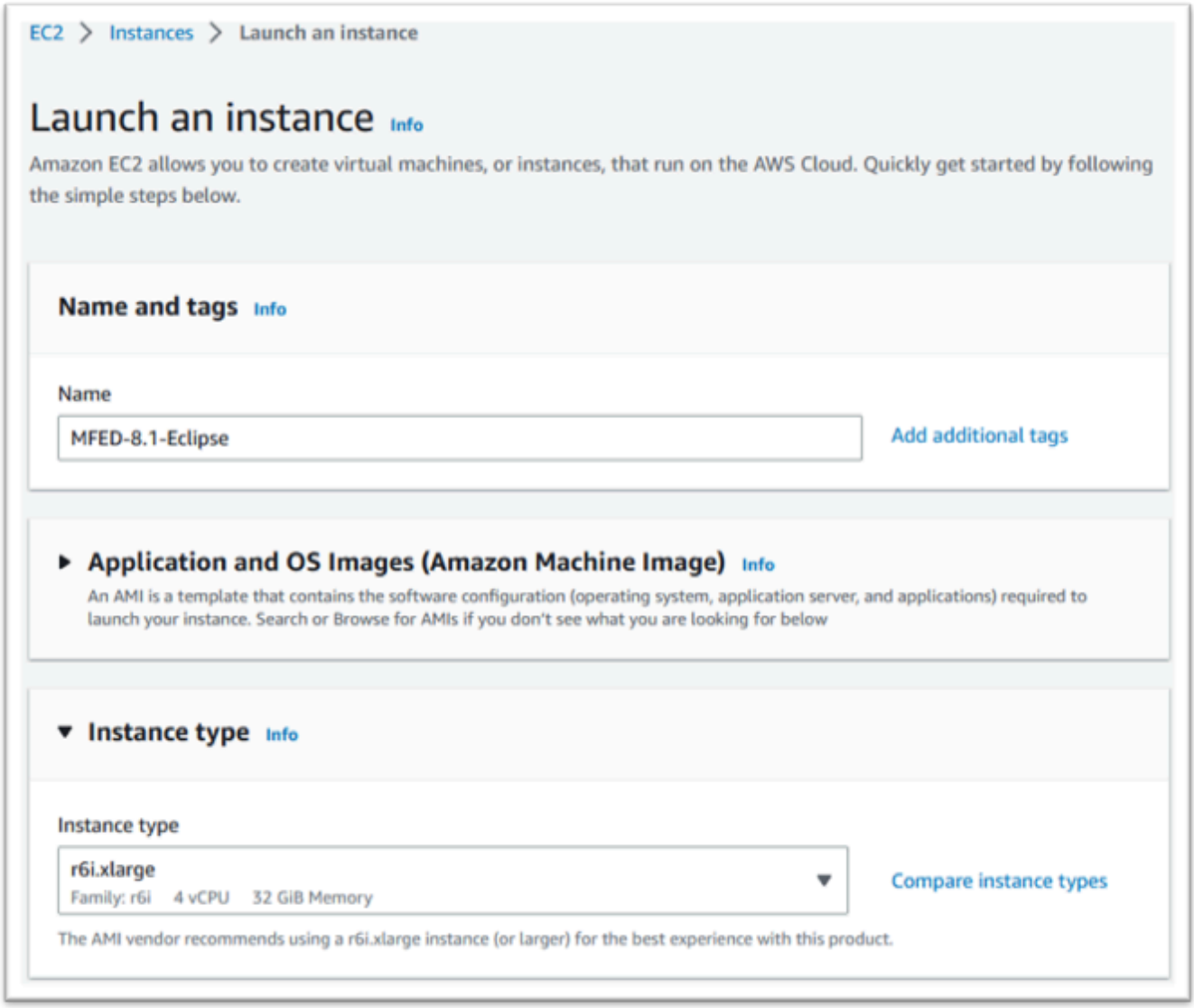


5. Masukkan nama untuk server.
6. Pilih jenis instance.

   Jenis Instance yang dipilih harus ditentukan oleh kinerja proyek dan persyaratan biaya. Berikut ini adalah titik awal yang disarankan:

   - Untuk Enterprise Analyzer, sebuah r6i.xlarge
   - Untuk Pengembang Perusahaan, sebuah r6i.large
   - Untuk instance mandiri Enterprise Server, sebuah r6i.xlarge
   - Untuk Micro Focus Performance Availability Cluster (PAC) dengan scale-out, sebuah r6i.large

   > **Note**
   >
   > Bagian Application and OS Images telah diciutkan untuk screen shot.



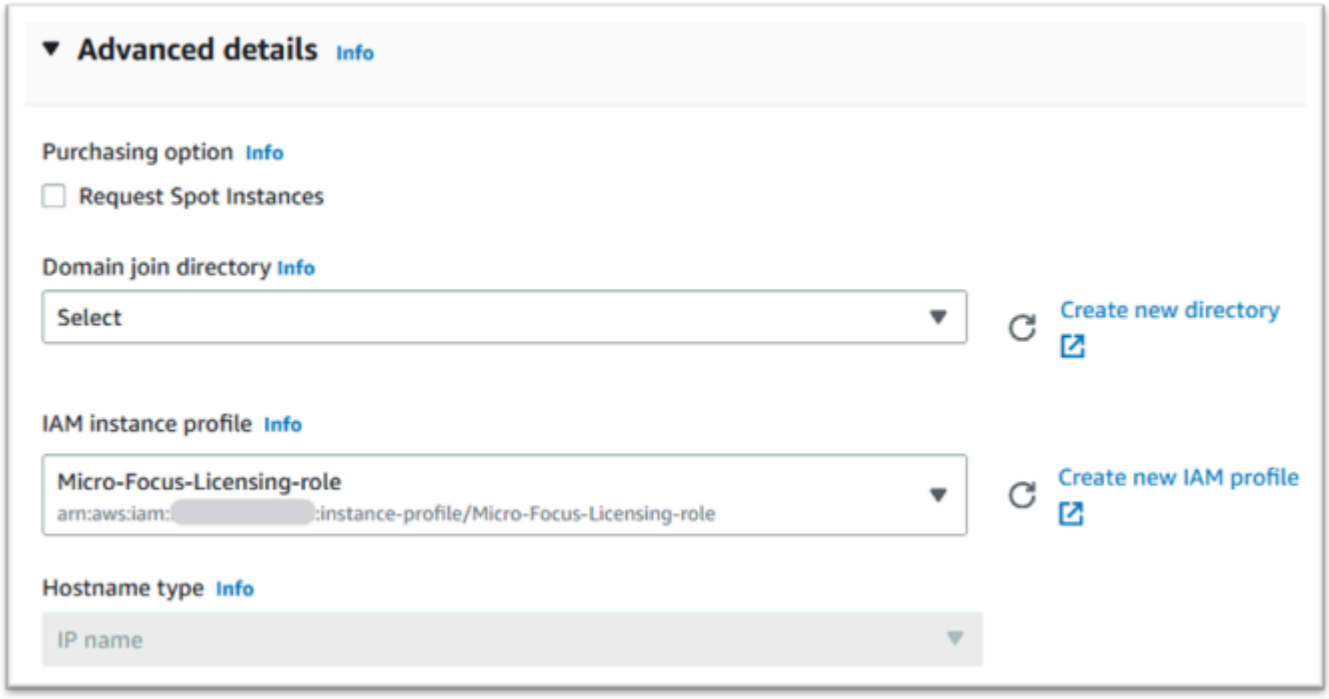
7. Pilih atau buat (dan simpan) pasangan kunci (tidak ditampilkan).

   Untuk informasi selengkapnya tentang pasangan kunci untuk instans Linux, lihat pasangan kunci Amazon EC2 dan instans Linux.

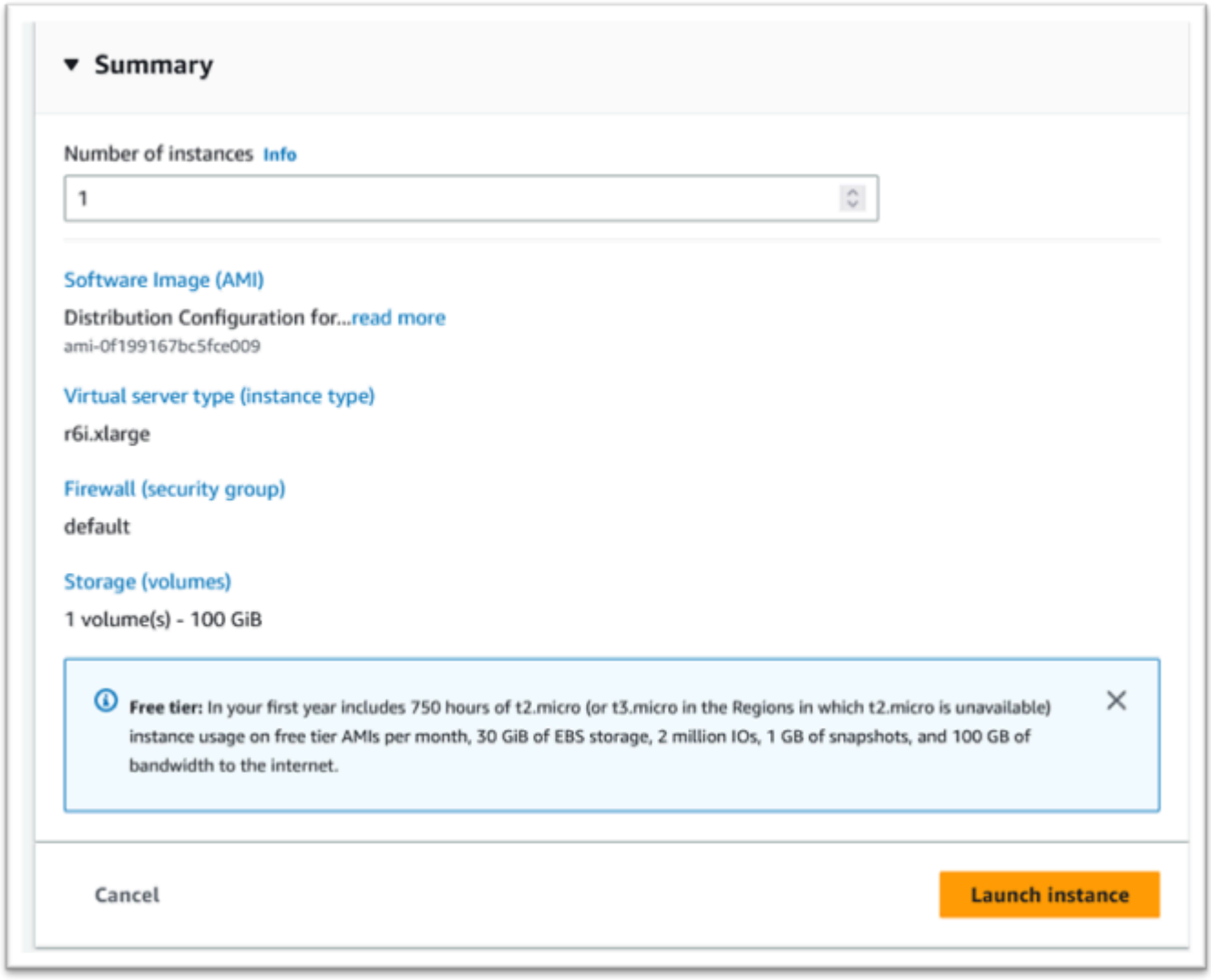
   Untuk informasi selengkapnya tentang pasangan kunci untuk instans Windows, lihat pasangan kunci Amazon EC2 dan instans Windows.
8. Edit pengaturan Jaringan dan pilih VPC yang diizinkan dan Subnet yang sesuai.
9. Pilih atau buat Grup Keamanan. Jika ini adalah instance Enterprise Server EC2, biasanya memungkinkan lalu lintas TCP ke port 86 dan 10086 untuk mengelola konfigurasi Micro Focus.
10. Konfigurasikan penyimpanan untuk instans Amazon EC2 secara opsional.

11. Penting - Perluas Detail lanjutan dan di bawah profil instans IAM pilih peran Lisensi yang dibuat sebelumnya, misalnya “Micro-focus-Licensing-role”.

> **Note**
>
> Jika langkah ini terlewatkan, setelah instance dibuat, Anda dapat memodifikasi peran IAM dari opsi Keamanan menu Tindakan untuk instans EC2.

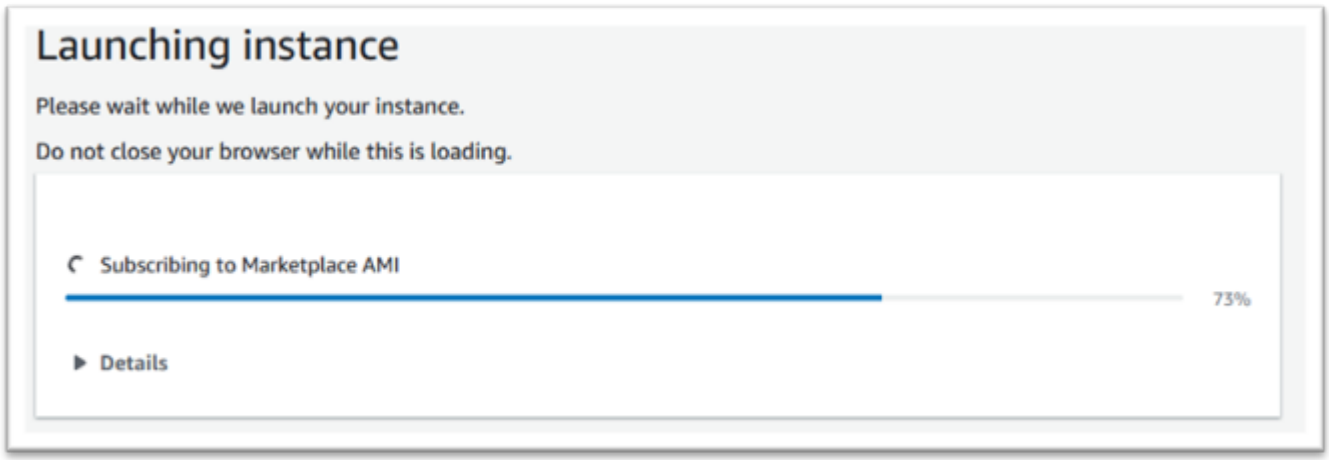


12. Tinjau Ringkasan dan dorong Instans Peluncuran.

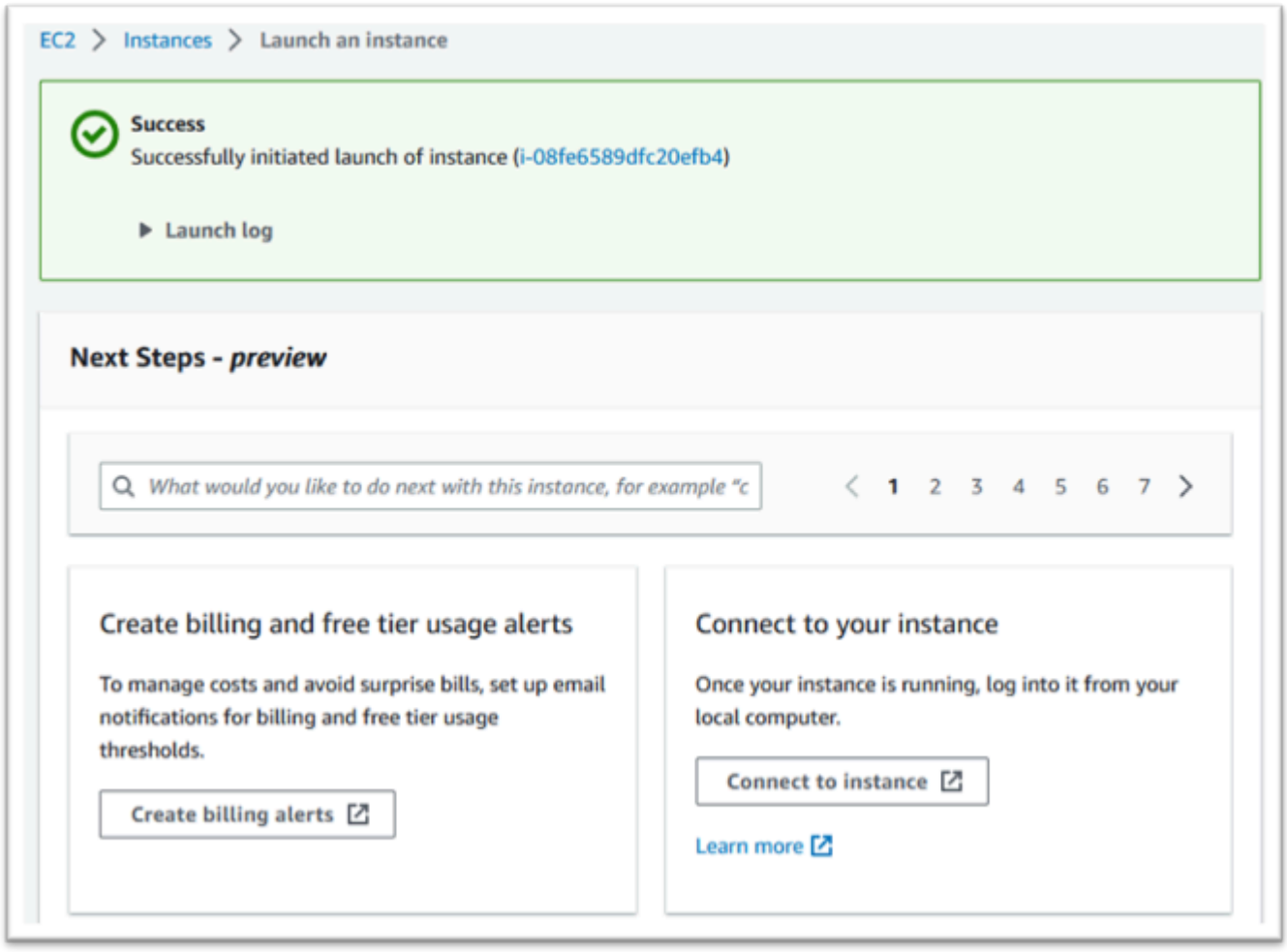


13. Peluncuran instance akan gagal jika jenis server virtual yang tidak valid dipilih.

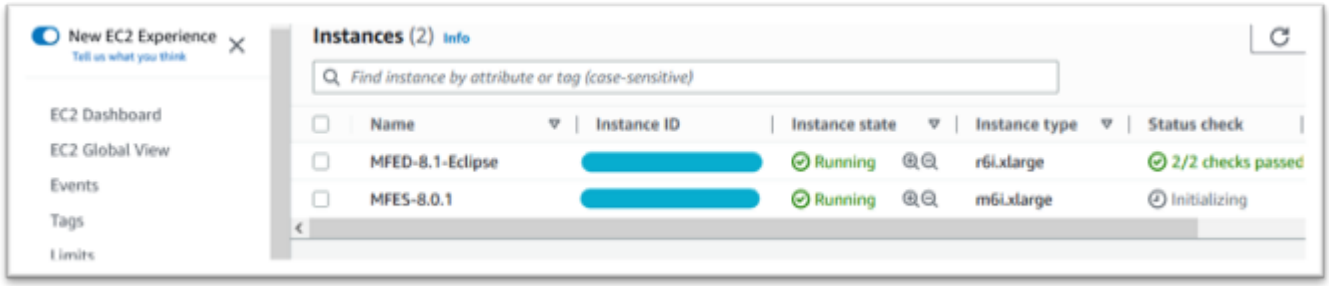
    Jika ini terjadi, pilih Edit konfigurasi instance dan ubah jenis instance.



14. Setelah pesan “Sukses” ditampilkan, pilih Connect to instance untuk mendapatkan detail koneksi.



15. Atau, navigasikan ke EC2 di. AWS Management Console
16. Pilih Instans untuk melihat status instans baru.



Subnet atau VPC tanpa Akses Internet

Lakukan perubahan tambahan ini jika subnet atau VPC tidak memiliki akses Internet keluar.

Manajer lisensi memerlukan akses ke layanan AWS berikut:

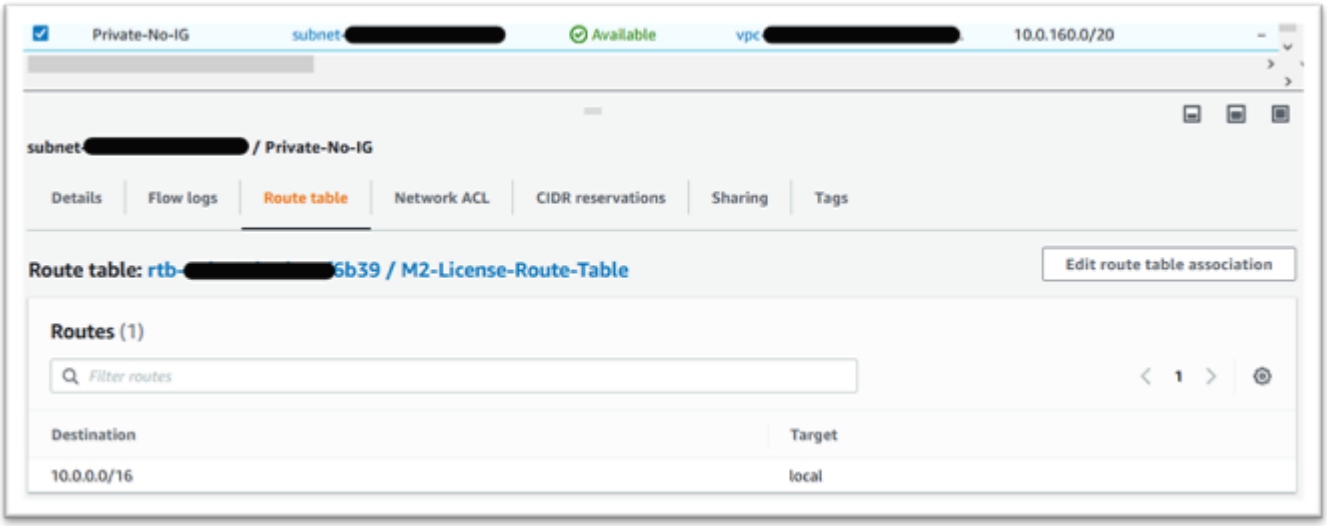
- com.amazonaws.*wilayah*.s3
- com.amazonaws. *wilayah .ec2*
- com.amazonaws.*wilayah*.pengelola lisensi
- com.amazonaws. *wilayah* .sts

Langkah-langkah sebelumnya mendefinisikan com.amazonaws. layanan *region* .s3 sebagai titik akhir gateway. Endpoint ini membutuhkan entri tabel rute untuk setiap subnet tanpa akses Internet.

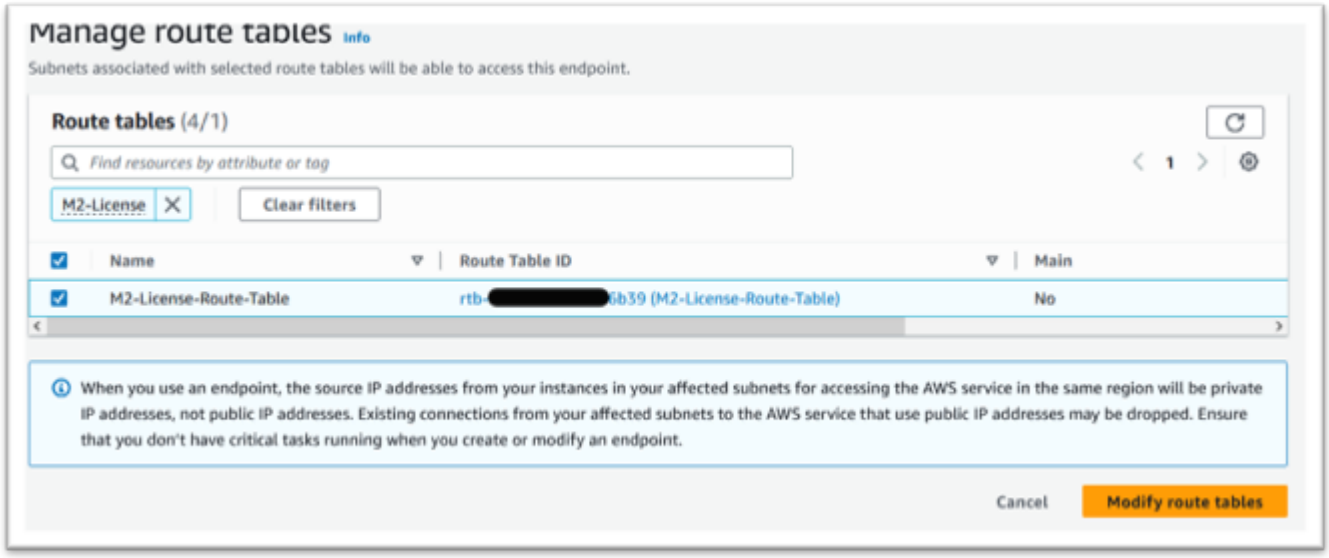
Tiga layanan tambahan akan didefinisikan sebagai titik akhir antarmuka.

Tambahkan entri tabel Route untuk titik akhir Amazon S3

1. Arahkan ke VPC di AWS Management Console dan pilih Subnet.
2. Pilih subnet tempat instans Amazon EC2 akan dibuat dan pilih tab Route Table.
3. Perhatikan beberapa digit tambahan dari id tabel Route. Misalnya, 6b39 pada gambar di bawah ini.



4. Pilih Endpoints dari panel navigasi.
5. Pilih titik akhir yang dibuat sebelumnya dan kemudian Kelola tabel Rute, baik dari tab Tabel Rute untuk titik akhir, atau dari drop-down Tindakan.
6. Pilih tabel Route menggunakan digit yang diidentifikasi sebelumnya dan tekan Ubah tabel rute.

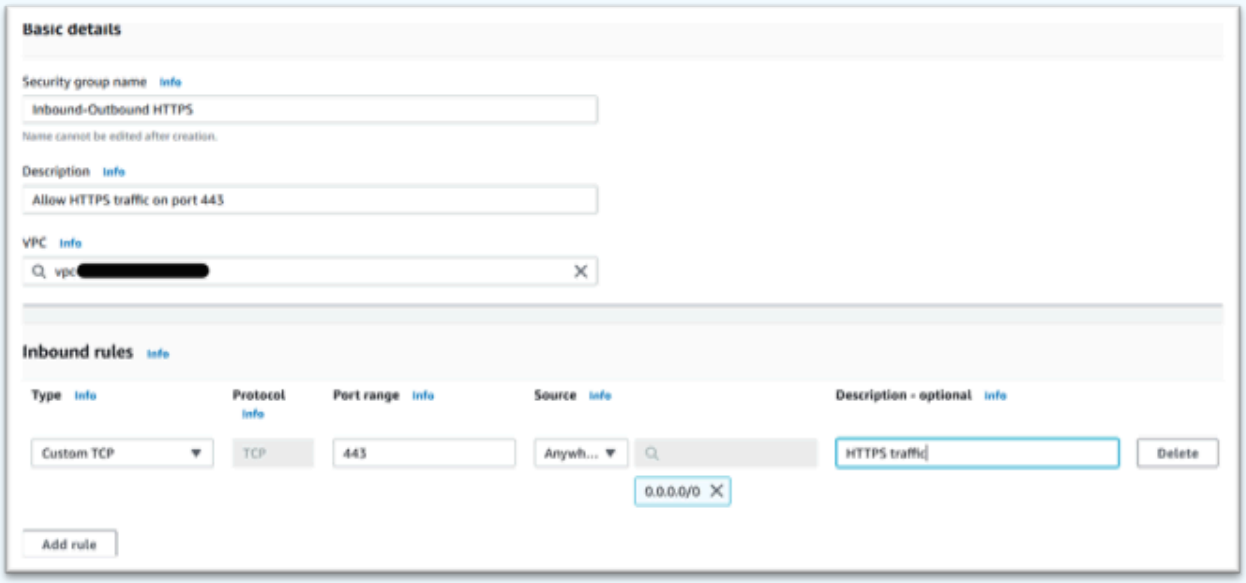


Tentukan grup keamanan yang diperlukan

Layanan Amazon EC2, AWS STS, dan License Manager berkomunikasi melalui HTTPS melalui port 443. Komunikasi ini bersifat bi-directional dan membutuhkan aturan masuk dan keluar untuk memungkinkan instance berkomunikasi dengan layanan.

1. Arahkan ke Amazon VPC di. AWS Management Console
2. Temukan Grup Keamanan di bilah navigasi dan pilih Buat grup keamanan.
3. Masukkan nama dan deskripsi grup Keamanan, misalnya “HTTPS Inbound-Outbound”.
4. Tekan X di area pemilihan VPC untuk menghapus VPC default, dan pilih VPC yang berisi titik akhir S3.
5. Tambahkan Aturan Masuk yang memungkinkan lalu lintas TCP di Port 443 dari mana saja.

> **Note**
>
> Aturan masuk (dan keluar) dapat dibatasi lebih lanjut dengan membatasi Sumber. Untuk informasi selengkapnya, lihat Mengontrol lalu lintas ke AWS sumber daya Anda menggunakan grup keamanan di Panduan Pengguna Amazon VPC.

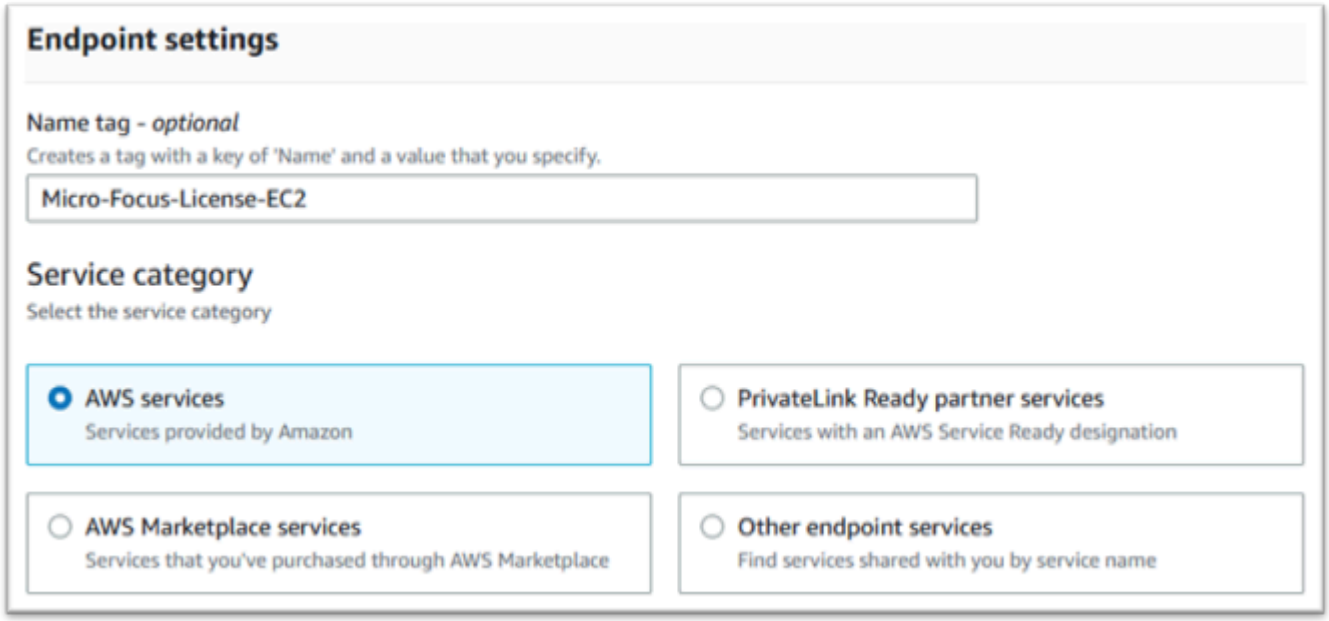


6. Tekan Buat grup keamanan.

Buat titik akhir layanan

Ulangi proses ini tiga kali — sekali untuk setiap layanan.

1. Arahkan ke Amazon VPC di AWS Management Console dan pilih Endpoints.
2. Tekan Buat titik akhir.
3. Masukkan nama, misalnya “Micro-focus-License-EC2", “Micro-focus-License-sts”, atau “Micro-Focus-License-Manager”.
4. Pilih Kategori Layanan AWS Services.



5. Di bawah Layanan, cari layanan Antarmuka yang cocok yang merupakan salah satu dari:

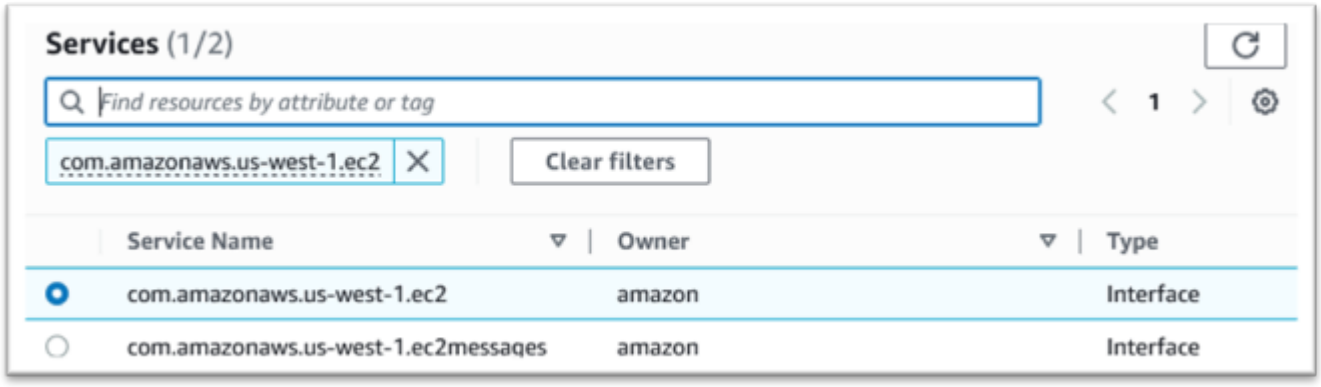
   - “com.amazonaws. *wilayah* .ec2”
   - “com.amazonaws. *wilayah* .sts”
   - “com.amazonaws. *region* .license-manager”

   Sebagai contoh:

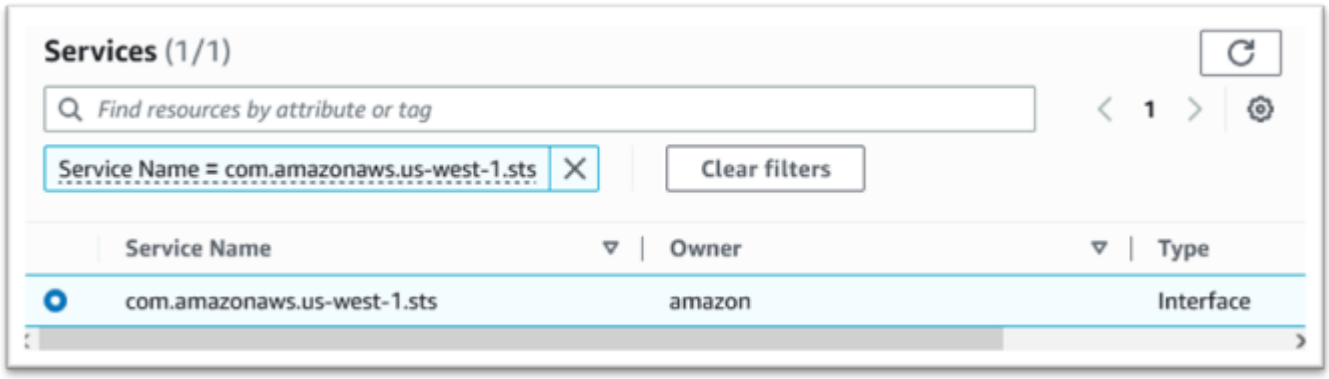
   - “com.amazonaws.us-west-1.ec2"
   - “com.amazonaws.us-west-1.sts”
   - “com.amazonaws.us-west-1.license-manager”

6. Pilih layanan Antarmuka yang cocok.

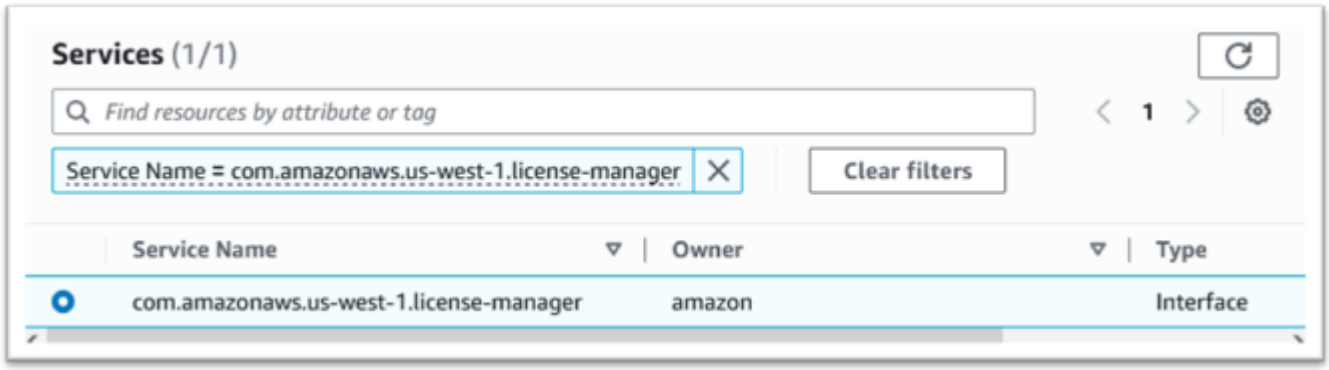
   com.amazonaws. ***wilayah*** .ec2:



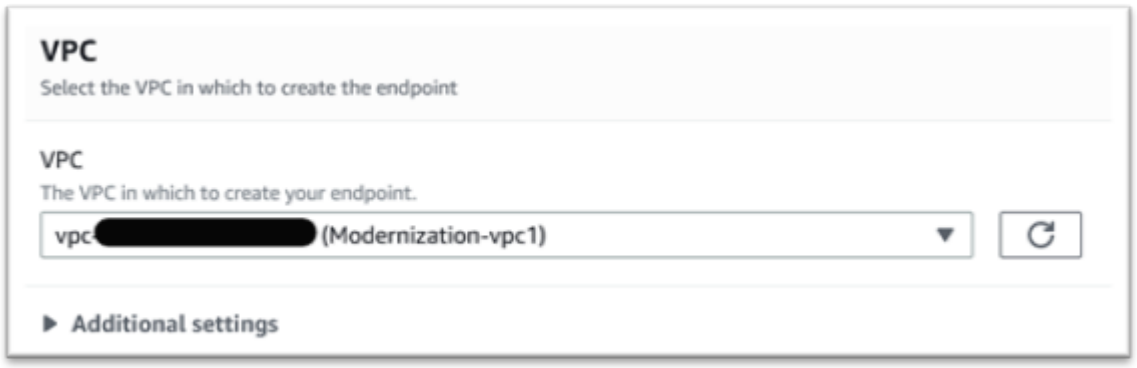
com.amazonaws. ***wilayah*** .sts:



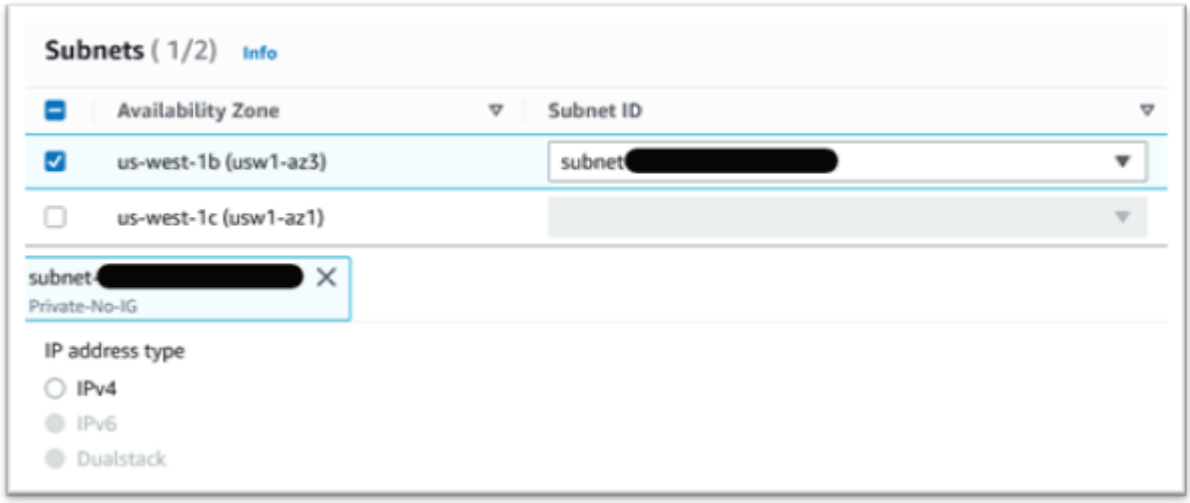
com.amazonaws. ***region*** .license-manager:



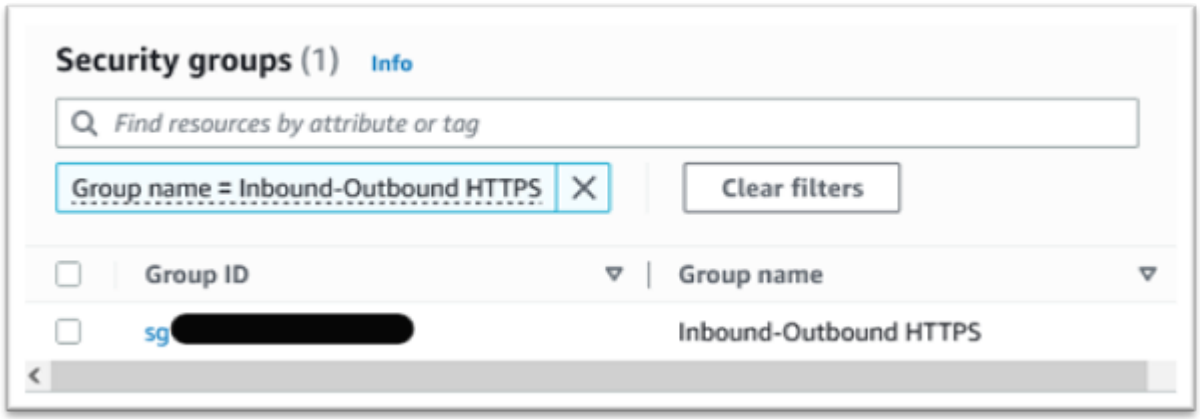
7. Untuk VPC pilih VPC untuk instance.



8. Pilih Availability Zone dan Subnet untuk VPC.



9. Pilih Grup Keamanan yang dibuat sebelumnya.



10. Di bawah Kebijakan pilih Akses Penuh.

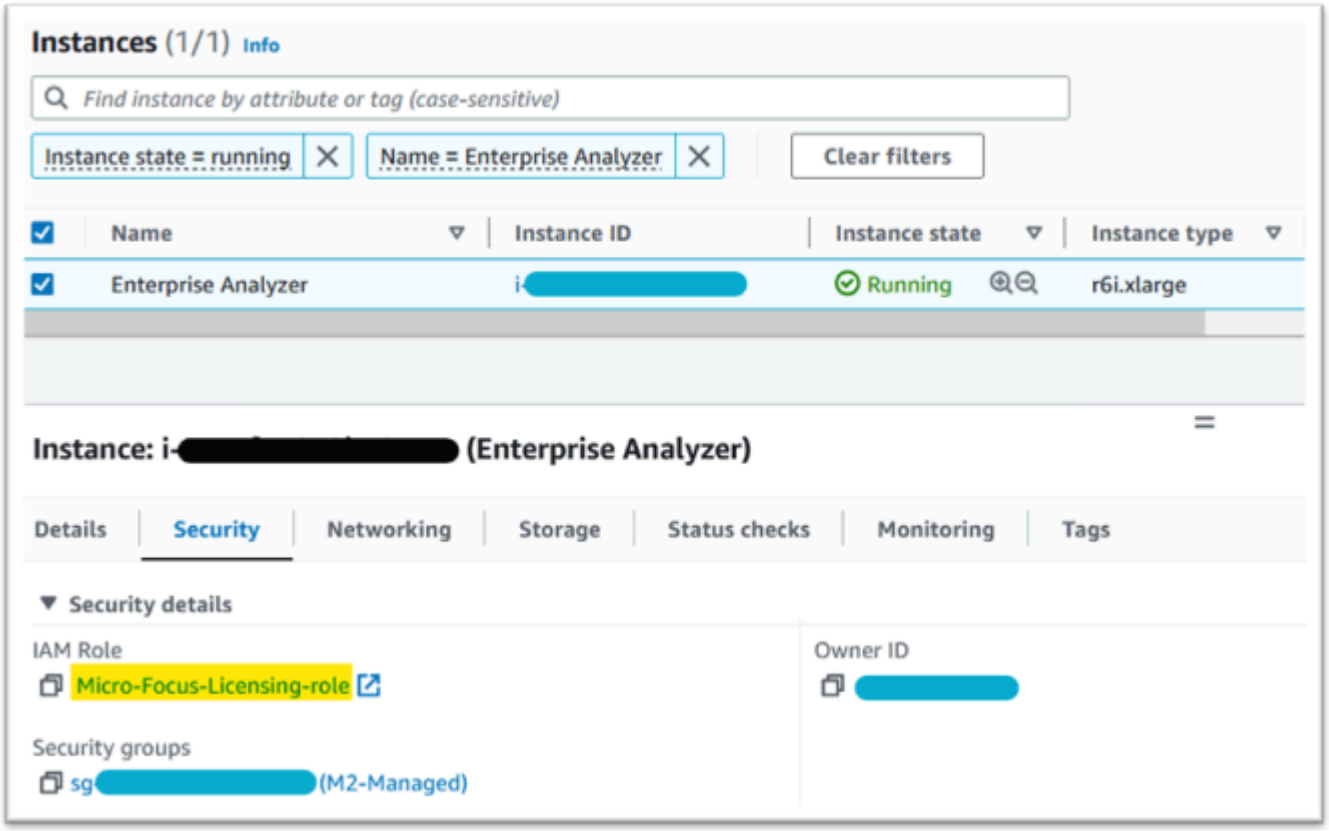
11. Pilih Buat Titik Akhir.
12. Ulangi proses ini untuk antarmuka yang tersisa.

Memecahkan masalah lisensi

Jika Anda mengalami kesulitan mengakses atau menggunakan AMI, informasi berikut dapat membantu Anda.

Verifikasi Instans Amazon EC2 memiliki peran Lisensi IAM

Ini dapat diperiksa pada tab Keamanan pada Detail Instans Amazon EC2. Ini dapat diubah menggunakan Opsi Keamanan dari menu tarik-turun Tindakan.



Gunakan Reachability Analyzer

Temukan Reachability Analyzer di halaman Konsol. AWS Network Manager

Buat dan analisis jalur antara instans Amazon EC2 yang dibuat dari AMI dan Titik Akhir VPC Amazon S3.

Jika Instans Amazon EC2 tidak memiliki akses internet, ulangi analisis jalur ke semua 4 titik akhir.

Untuk informasi selengkapnya tentang Reachability Analyzer, lihat Memulai Reachability Analyzer dalam panduan Reachability Analyzer.

Jalankan lisensi-daemon

Pada Windows Enterprise Developer menggunakan perintah berikut dari Command Prompt:

```
“C:\Program Files (x86)\Micro Focus\Enterprise Developer\AdoptOpenJDK\bin\java” -jar
 "C:\Program Files (x86)\Micro Focus\Licensing\aws-license-daemon.jar"
```

dan memeriksa outputnya. Abaikan pesan SLF4J dan cari pengecualian pertama.

Pada Enterprise Analyzer gunakan perintah berikut dari Command Prompt:

```
"C:\Program Files (x86)\Micro Focus\AdoptOpenJDK\bin\java" -jar "C:\Program Files
 (x86)\Micro Focus\Licensing\aws-license-daemon.jar"
```

dan memeriksa outputnya. Abaikan pesan SLF4J dan cari pengecualian pertama.

Di Linux jalankan:

```
java -jar /var/microfocuslicensing/bin/aws-license-daemon.jar
```

Abaikan pesan SLF4J dan cari pengecualian pertama.

Misalnya, jika sumber daya Amazon S3 tidak tersedia, pengecualiannya adalah sebagai berikut:

```
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.

Exception in thread "main" software.amazon.awssdk.services.s3.model.S3Exception: Access
 Denied (Service: S3, Status Code: 403, Request ID: P6
```

Pesan pengecualian menunjukkan sumber daya mana yang tidak tersedia. Bandingkan nilai konfigurasi dengan yang ditunjukkan dalam topik ini.

Tutorial: Siapkan AppStream 2.0 untuk digunakan dengan Micro Focus Enterprise Analyzer dan Micro Focus Enterprise Developer

AWSModernisasi Mainframe menyediakan beberapa alat melalui Amazon 2.0. AppStream AppStream2.0 adalah layanan streaming aplikasi yang dikelola sepenuhnya dan aman yang memungkinkan Anda melakukan streaming aplikasi desktop ke pengguna tanpa menulis ulang aplikasi. AppStream2.0 memberi pengguna akses instan ke aplikasi yang mereka butuhkan dengan pengalaman pengguna yang responsif dan lancar pada perangkat pilihan mereka. Menggunakan AppStream 2.0 untuk meng-host alat khusus mesin waktu proses memberi tim aplikasi pelanggan kemampuan untuk menggunakan alat langsung dari browser web mereka, berinteraksi dengan file aplikasi yang disimpan di bucket atau repositori Amazon S3. CodeCommit

Untuk informasi tentang dukungan browser di AppStream 2.0, lihat Persyaratan Sistem dan Dukungan Fitur (Browser Web) di Panduan Administrasi Amazon AppStream 2.0. Jika Anda

mengalami masalah saat menggunakan AppStream 2.0, lihat Mengatasi Masalah Pengguna AppStream 2.0 di Panduan Administrasi Amazon AppStream 2.0.

Dokumen ini ditujukan untuk anggota tim operasi pelanggan. Ini menjelaskan cara menyiapkan armada dan tumpukan Amazon AppStream 2.0 untuk menjadi tuan rumah alat Micro Focus Enterprise Analyzer dan Micro Focus Enterprise Developer yang digunakan dengan AWS Modernisasi Mainframe. Micro Focus Enterprise Analyzer biasanya digunakan selama fase Menilai dan Micro Focus Enterprise Developer biasanya digunakan selama fase Migrasi dan Modernisasi pendekatan Modernisasi Mainframe. AWS Jika Anda berencana untuk menggunakan Enterprise Analyzer dan Enterprise Developer, Anda harus membuat armada dan tumpukan terpisah untuk setiap alat. Setiap alat membutuhkan armada dan tumpukannya sendiri karena istilah perizinannya berbeda.

> **Important**
>
> Langkah-langkah dalam tutorial ini didasarkan pada AWS CloudFormation template yang dapat diunduh cfn-m2- .yml.yml. appstream-fleet-ea-ed

Topik

Prasyarat

- Unduh templatnya: cfn-m2- appstream-fleet-ea-ed .yml.id.
- Dapatkan ID VPC default dan grup keamanan Anda. Untuk informasi selengkapnya tentang VPC default, lihat VPC Default di Panduan Pengguna Amazon VPC. Untuk informasi selengkapnya

tentang grup keamanan default, lihat Grup keamanan default dan kustom di Panduan Pengguna Amazon EC2 untuk Instans Linux.

- Pastikan Anda memiliki izin berikut:
  - membuat tumpukan, armada, dan pengguna di AppStream 2.0.
  - membuat tumpukan dalam AWS CloudFormation menggunakan template.
  - buat bucket dan unggah file ke bucket di Amazon S3.
  - unduh kredensi (`access_key_id`dan`secret_access_key`) dari IAM.

Langkah 1: Dapatkan gambar AppStream 2.0

Pada langkah ini, Anda membagikan gambar AppStream 2.0 untuk Enterprise Analyzer dan Enterprise Developer dengan AWS akun Anda.

1. Buka konsol Modernisasi AWS Mainframe di https://console.aws.amazon.com/m2/.
2. Di navigasi kiri, pilih Tools.
3. Dalam Analisis, pengembangan, dan bangun aset, pilih Bagikan aset dengan AWS akun saya.

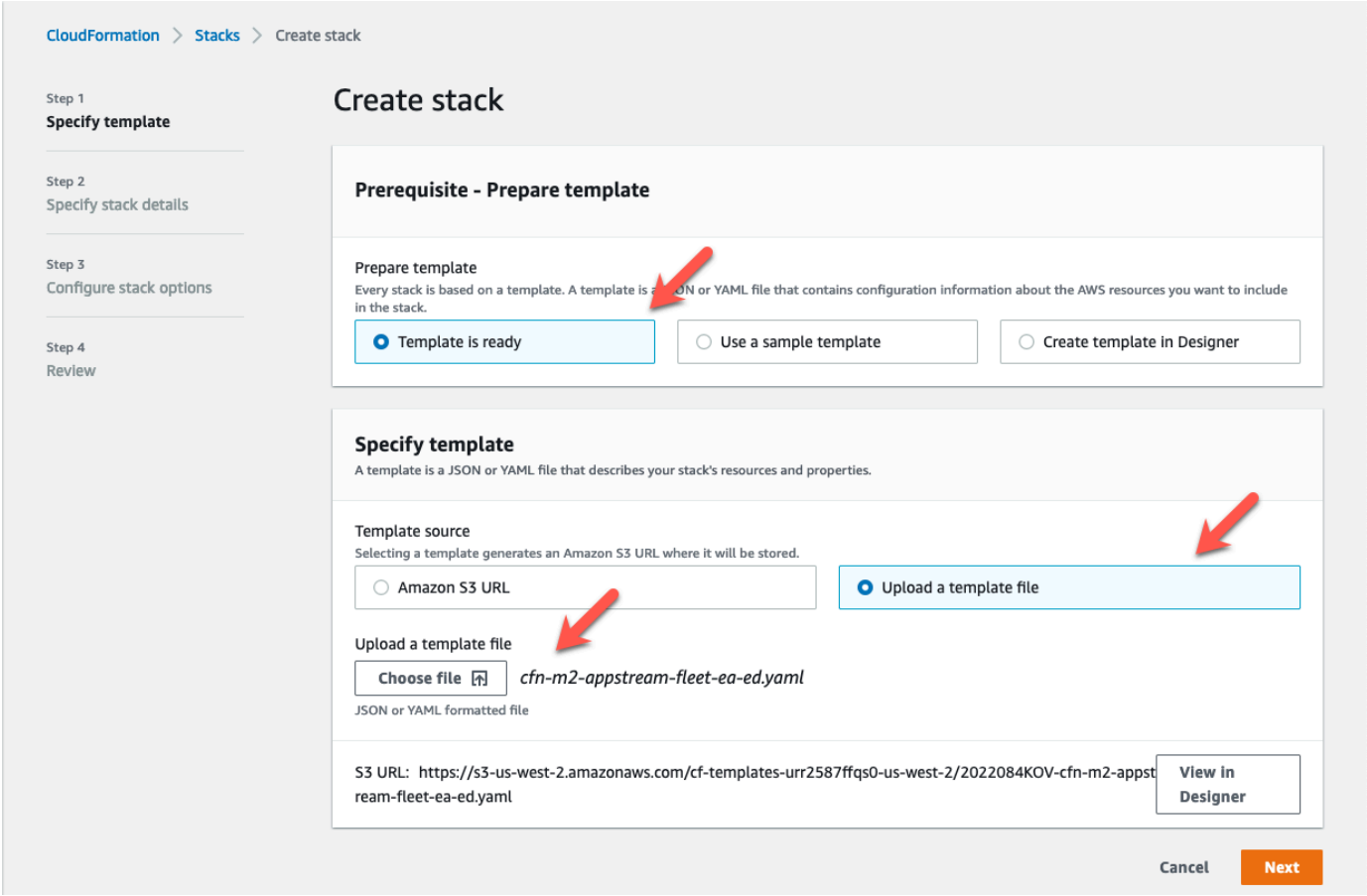
Langkah 2: Buat tumpukan menggunakan AWS CloudFormation template

Pada langkah ini, Anda menggunakan AWS CloudFormation template yang diunduh untuk membuat tumpukan dan armada AppStream 2.0 untuk menjalankan Micro Focus Enterprise Analyzer. Anda dapat mengulangi langkah ini nanti untuk membuat tumpukan dan armada AppStream 2.0 lainnya untuk menjalankan Micro Focus Enterprise Developer, karena setiap alat memerlukan armada sendiri dan menumpuk di AppStream 2.0. Untuk informasi selengkapnya tentang AWS CloudFormation tumpukan, lihat Bekerja dengan tumpukan di AWS CloudFormationPanduan Pengguna.

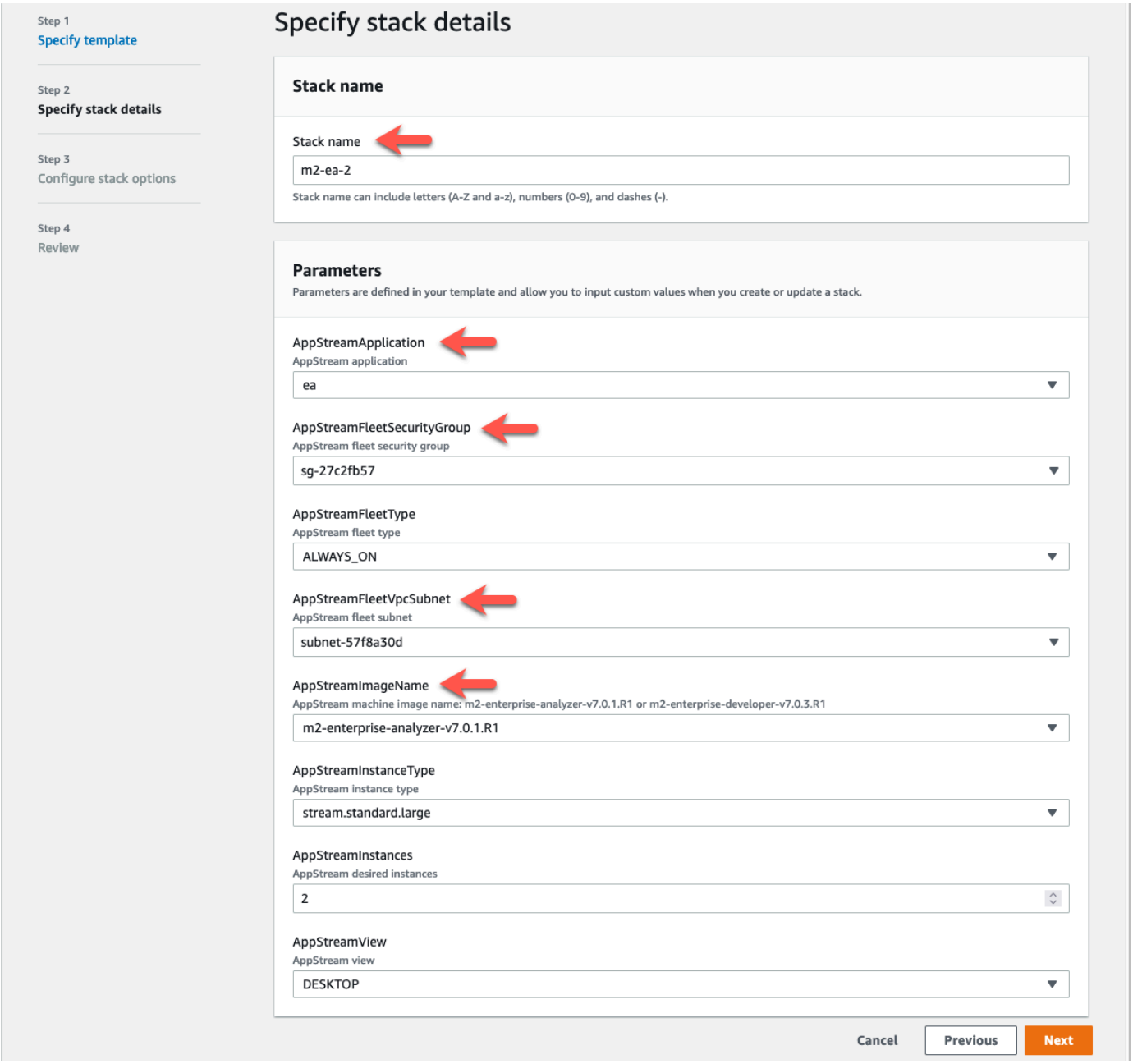
> **Note**
>
> AWSModernisasi Mainframe menambahkan biaya tambahan ke harga standar AppStream 2.0 untuk penggunaan Enterprise Analyzer dan Enterprise Developer. Untuk informasi selengkapnya, lihat Harga Modernisasi AWS Mainframe.

1. Unduh template cfn-m2- appstream-fleet-ea-ed .yml. jika perlu.

2. Buka AWS CloudFormation konsol dan pilih Create Stack dan dengan sumber daya baru (standar).
3. Dalam Prasyarat - Siapkan template, pilih Template sudah siap.
4. Di Tentukan Template, pilih Unggah file template.
5. Di Unggah file template, pilih Pilih file dan unggah templat [cfn-m2- appstream-fleet-ea-ed .yml.]
6. Pilih Selanjutnya.



7. Pada Tentukan detail tumpukan, masukkan informasi berikut:
   - Dalam nama Stack, masukkan nama pilihan Anda. Sebagai contoh, **`m2-ea`**.
   - Di AppStreamApplication, pilih ea.
   - Di AppStreamFleetSecurityGroup, pilih grup keamanan default VPC default Anda.
   - Di AppStreamFleetVpcSubnet, pilih subnet dalam VPC default Anda.
   - Di AppStreamImageName, pilih gambar yang dimulai dengan`m2-enterprise-analyzer`. Gambar ini berisi versi alat Micro Focus Enterprise Analyzer yang saat ini didukung.
   - Terima default untuk bidang lainnya, lalu pilih Berikutnya.



8. Terima semua default, lalu pilih Berikutnya lagi.
9. Pada Review, pastikan semua parameter adalah apa yang Anda inginkan.
10. Gulir ke bawah, pilih Saya mengakui bahwa AWS CloudFormation mungkin membuat sumber daya IAM dengan nama kustom, dan pilih Create Stack.

Dibutuhkan antara 20 dan 30 menit untuk tumpukan dan armada yang akan dibuat. Anda dapat memilih Segarkan untuk melihat AWS CloudFormation peristiwa saat terjadi.

Langkah 3: Buat pengguna di AppStream 2.0

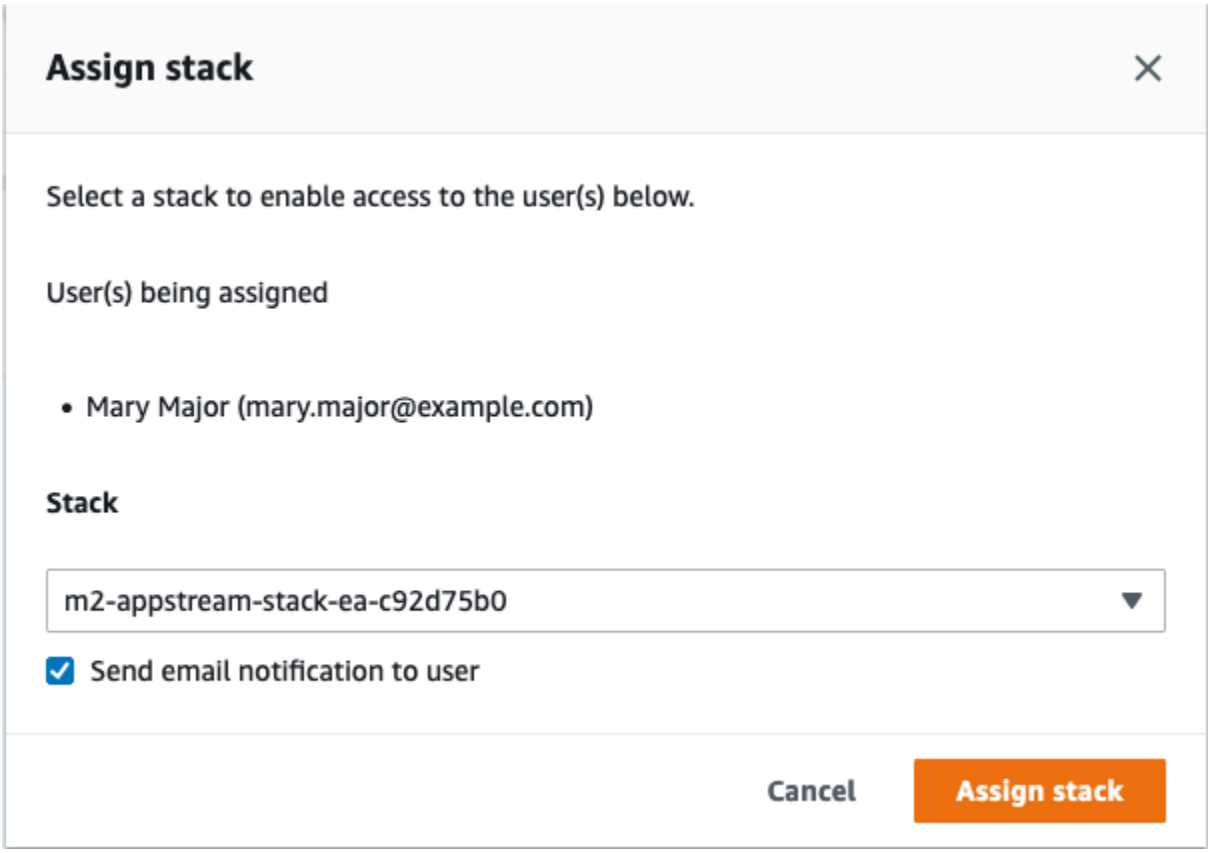
Saat Anda menunggu AWS CloudFormation untuk menyelesaikan pembuatan tumpukan, Anda dapat membuat satu atau lebih pengguna di AppStream 2.0. Pengguna ini adalah mereka yang akan menggunakan Enterprise Analyzer di AppStream 2.0. Anda perlu menentukan alamat email untuk setiap pengguna, dan memastikan bahwa setiap pengguna memiliki izin yang cukup untuk membuat bucket di Amazon S3, mengunggah file ke bucket, dan menautkan ke bucket untuk memetakan isinya.

1. Buka konsol AppStream 2.0.
2. Di navigasi kiri, pilih Pangkalan pengguna.
3. Pilih Create user (Buat pengguna).
4. Berikan alamat email tempat pengguna dapat menerima undangan email untuk menggunakan AppStream 2.0, nama depan dan nama belakang, lalu pilih Buat pengguna.
5. Ulangi jika perlu untuk membuat lebih banyak pengguna. Alamat email untuk setiap pengguna harus unik.

Untuk informasi selengkapnya tentang membuat pengguna AppStream 2.0, lihat AppStream2.0 User Pools di Panduan Administrasi Amazon AppStream 2.0.

Ketika AWS CloudFormation selesai membuat stack, Anda dapat menetapkan pengguna yang Anda buat ke stack, sebagai berikut:

1. Buka konsol AppStream 2.0.
2. Pilih nama pengguna.
3. Pilih Tindakan, lalu Tetapkan tumpukan.
4. Di Assign stack, pilih stack yang dimulai dengan`m2-appstream-stack-ea`.
5. Pilih Tetapkan tumpukan.



Menetapkan pengguna ke tumpukan menyebabkan AppStream 2.0 mengirim email ke pengguna di alamat yang Anda berikan. Email ini berisi tautan ke halaman login AppStream 2.0.

Langkah 4: Masuk ke AppStream 2.0

Pada langkah ini, Anda masuk ke AppStream 2.0 menggunakan tautan di email yang dikirim oleh AppStream 2.0 ke pengguna yang Anda buatLangkah 3: Buat pengguna di AppStream 2.0.

1. Masuk ke AppStream 2.0 menggunakan tautan yang disediakan dalam email yang dikirim oleh AppStream 2.0.
2. Ubah kata sandi Anda, jika diminta. Layar AppStream 2.0 yang Anda lihat mirip dengan yang berikut ini:

3. Pilih Desktop.
4. Di bilah tugas, pilih Cari dan masukkan **`D:`** untuk menavigasi ke Folder Beranda.

> **Note**
>
> Jika Anda melewatkan langkah ini, Anda mungkin mendapatkan `Device not ready` kesalahan saat mencoba mengakses Folder Beranda.

Kapan saja, jika Anda mengalami masalah saat masuk ke AppStream 2.0, Anda dapat memulai ulang armada AppStream 2.0 Anda dan mencoba masuk lagi, menggunakan langkah-langkah berikut.

1. Buka konsol AppStream 2.0.
2. Di navigasi kiri, pilih Armada.
3. Pilih armada yang Anda coba gunakan.
4. Pilih Tindakan, lalu pilih Berhenti.
5. Tunggu armada berhenti.
6. Pilih Tindakan, lalu pilih Mulai.

Proses ini bisa memakan waktu sekitar 10 menit.

Langkah 5: Verifikasi bucket di Amazon S3 (opsional)

Salah satu tugas yang diselesaikan oleh AWS CloudFormation template yang Anda gunakan untuk membuat tumpukan adalah membuat dua bucket di Amazon S3, yang diperlukan untuk menyimpan dan memulihkan data pengguna dan pengaturan aplikasi di seluruh sesi kerja. Ember ini adalah sebagai berikut:

- Nama dimulai dengan`appstream2-`. Bucket ini memetakan data ke Home Folder Anda di AppStream 2.0 (`D:\PhotonUser\My Files\Home Folder`).

  > **Note**
  >
  > Home Folder unik untuk alamat email tertentu dan dibagikan di semua armada dan tumpukan di akun tertentuAWS. Nama Home Folder adalah hash SHA256 dari alamat email pengguna, dan disimpan pada jalur berdasarkan hash itu.

- Nama dimulai dengan`appstream-app-settings-`. Bucket ini berisi informasi sesi pengguna untuk AppStream 2.0, dan mencakup pengaturan seperti favorit browser, profil koneksi IDE dan aplikasi, dan penyesuaian UI. Untuk informasi selengkapnya, lihat Cara Kerja Persistensi Pengaturan Aplikasi di Panduan Administrasi Amazon AppStream 2.0.

Untuk memverifikasi bahwa bucket telah dibuat, ikuti langkah-langkah berikut:

1. Buka konsol Amazon S3.
2. Di navigasi kiri, pilih Bucket.
3. Di Temukan ember berdasarkan nama, masukkan **`appstream`** untuk memfilter daftar.

Jika Anda melihat ember, tidak ada tindakan lebih lanjut diperlukan. Sadarilah bahwa ember itu ada. Jika Anda tidak melihat ember, maka AWS CloudFormation template belum selesai berjalan, atau terjadi kesalahan. Buka AWS CloudFormation konsol dan tinjau pesan pembuatan tumpukan.

Langkah selanjutnya

Sekarang infrastruktur AppStream 2.0 sudah diatur, Anda dapat mengatur dan mulai menggunakan Enterprise Analyzer. Untuk informasi selengkapnya, lihat Tutorial: Mengatur Enterprise Analyzer pada 2.0 AppStream . Anda juga dapat mengatur Enterprise Developer. Untuk informasi selengkapnya, lihat Tutorial: Mengatur Pengembang Perusahaan Fokus Mikro di AppStream 2.0.

Pembersihan sumber daya

Prosedur untuk membersihkan tumpukan dan armada yang dibuat dijelaskan dalam Create an AppStream 2.0 Fleet and Stack.

Ketika objek AppStream 2.0 telah dihapus, administrator akun juga dapat, jika sesuai, membersihkan bucket Amazon S3 untuk Pengaturan Aplikasi dan Folder Rumah.

> **Note**
>
> Folder beranda untuk pengguna tertentu unik di semua armada, jadi Anda mungkin perlu mempertahankannya jika tumpukan AppStream 2.0 lainnya aktif di akun yang sama.

Akhirnya, AppStream 2.0 saat ini tidak memungkinkan Anda untuk menghapus pengguna menggunakan konsol. Sebagai gantinya, Anda harus menggunakan API layanan dengan CLI. Untuk informasi selengkapnya, lihat Administrasi Pangkalan Pengguna di Panduan Administrasi Amazon AppStream 2.0.

Tutorial: Mengatur Enterprise Analyzer pada 2.0 AppStream

Tutorial ini menjelaskan cara mengatur Micro Focus Enterprise Analyzer untuk menganalisis satu atau lebih aplikasi mainframe. Alat Enterprise Analyzer menyediakan beberapa laporan berdasarkan analisisnya terhadap kode sumber aplikasi dan definisi sistem.

Pengaturan ini dirancang untuk mendorong kolaborasi tim. Instalasi menggunakan bucket Amazon S3 untuk berbagi kode sumber dengan disk virtual. Melakukan hal ini menggunakan Rclone) pada mesin Windows. Dengan instans Amazon RDS umum yang menjalankan PostgreSQL, setiap anggota tim dapat mengakses semua laporan yang diminta.

Anggota tim juga dapat memasang disk virtual Amazon S3 yang didukung di mesin pribadi mereka. dan memperbarui bucket sumber dari workstation mereka. Mereka berpotensi menggunakan skrip atau bentuk otomatisasi lainnya pada mesin mereka jika mereka terhubung ke sistem internal lokal lainnya.

Pengaturan didasarkan pada gambar Windows AppStream 2.0 yang dibagikan Modernisasi AWS Mainframe dengan pelanggan. Pengaturan juga didasarkan pada pembuatan armada AppStream 2.0 dan tumpukan seperti yang dijelaskan dalam. Tutorial: Siapkan AppStream 2.0 untuk digunakan dengan Micro Focus Enterprise Analyzer dan Micro Focus Enterprise Developer

> **Important**
>
> Langkah-langkah dalam tutorial ini mengasumsikan bahwa Anda mengatur AppStream 2.0 dengan AWS CloudFormation template yang dapat diunduh cfn-m2- .yml. appstream-fleet-ea-ed Untuk informasi selengkapnya, lihat Tutorial: Siapkan AppStream 2.0 untuk digunakan dengan Micro Focus Enterprise Analyzer dan Micro Focus Enterprise Developer.
> Untuk melakukan langkah-langkah dalam tutorial ini, Anda harus mengatur armada dan tumpukan Enterprise Analyzer Anda dan mereka harus berjalan.

Untuk penjelasan lengkap tentang fitur dan kiriman Enterprise Analyzer, lihat Dokumentasi Penganalisis Perusahaan di situs web Micro Focus.

Isi gambar

Selain aplikasi Enterprise Analyzer itu sendiri, gambar berisi alat dan pustaka berikut.

Alat pihak ketiga

- Python
- Rklon
- pGADmin
- git-scm
- Pengemudi PostgreSQL ODBC

Perpustakaan di `C:\Users\Public`

- BankDemo kode sumber dan definisi proyek untuk Pengembang Perusahaan:`m2-bankdemo-template.zip`.
- Paket instalasi MFA untuk mainframe:. `mfa.zip` Untuk informasi selengkapnya, lihat Ikhtisar Akses Mainframe di dokumentasi Pengembang Perusahaan Fokus Mikro.
- File perintah dan konfigurasi untuk Rclone (petunjuk penggunaannya dalam tutorial): dan. `m2-rclone.cmd` `m2-rclone.conf`

Topik

- Prasyarat

Prasyarat

- Unggah kode sumber dan definisi sistem untuk aplikasi pelanggan yang ingin Anda analisis ke bucket S3. Definisi sistem termasuk CICS CSD, definisi objek DB2, dan sebagainya. Anda dapat membuat struktur folder di dalam bucket yang masuk akal untuk bagaimana Anda ingin mengatur artefak aplikasi. Misalnya, ketika Anda membuka zip BankDemo sampel, ia memiliki struktur berikut:

```
demo
     |--> jcl
     |--> RDEF
     |--> transaction
     |--> xa
```

- Buat dan mulai instans Amazon RDS yang menjalankan PostgreSQL. Contoh ini akan menyimpan data dan hasil yang dihasilkan oleh Enterprise Analyzer. Anda dapat membagikan contoh ini dengan semua anggota tim aplikasi. Selain itu, buat skema kosong yang disebut `m2_ea` (atau nama lain yang sesuai) dalam database. Tentukan kredensil untuk pengguna resmi yang memungkinkan mereka membuat, menyisipkan, memperbarui, dan menghapus item dalam skema ini. Anda dapat memperoleh nama database, URL endpoint servernya, dan port TCP dari konsol Amazon RDS atau dari administrator akun.
- Pastikan Anda telah mengatur akses terprogram ke AndaAkun AWS. Untuk informasi selengkapnya, lihat Akses terprogram di. Referensi Umum Amazon Web

Langkah 1: Pengaturan

1. Mulai sesi dengan AppStream 2.0 dengan URL yang Anda terima dalam pesan email selamat datang dari AppStream 2.0.
2. Gunakan email Anda sebagai ID pengguna Anda, dan tentukan kata sandi permanen Anda.

3. Pilih tumpukan Enterprise Analyzer.
4. Pada halaman menu AppStream 2.0, pilih Desktop untuk mencapai desktop Windows yang sedang streaming armada.

Langkah 2: Buat folder virtual berbasis Amazon S3 di Windows

> **Note**
>
> Jika Anda sudah menggunakan Rclone selama pratinjau Modernisasi AWS Mainframe, Anda harus memperbarui `m2-rclone.cmd` ke versi yang lebih baru yang terletak di. `C:\Users\Public`

1. Salin `m2-rclone.cmd` file `m2-rclone.conf` dan file yang disediakan `C:\Users\Public` ke folder rumah Anda `C:\Users\PhotonUser\My Files\Home Folder` menggunakan File Explorer.
2. Perbarui parameter `m2-rclone.conf` konfigurasi dengan kunci AWS akses Anda dan rahasia yang sesuai, serta AndaWilayah AWS.

```
[m2-s3]
type = s3
provider = AWS
access_key_id = YOUR-ACCESS-KEY
secret_access_key = YOUR-SECRET-KEY
region = YOUR-REGION
acl = private
server_side_encryption = AES256
```

3. Di `m2-rclone.cmd`, lakukan perubahan berikut:
   - Ubah `your-s3-bucket` ke nama bucket Amazon S3 Anda. Sebagai contoh, `m2-s3-mybucket`.
   - Ubah `your-s3-folder-key` ke kunci bucket Amazon S3 Anda. Sebagai contoh, `myProject`.
   - Ubah `your-local-folder-path` ke jalur direktori tempat Anda ingin file aplikasi disinkronkan dari bucket Amazon S3 yang berisi file tersebut. Sebagai contoh, `D:\PhotonUser\My Files\Home Folder\m2-new`. Direktori yang disinkronkan ini harus

merupakan subdirektori dari Folder Rumah agar AppStream 2.0 dapat mencadangkan dan mengembalikannya dengan benar pada awal dan akhir sesi.

```
:loop
timeout /T 10
"C:\Program Files\rclone\rclone.exe" sync m2-s3:your-s3-bucket/your-s3-folder-
key "D:\PhotonUser\My Files\Home Folder\your-local-folder-path" --config "D:
\PhotonUser\My Files\Home Folder\m2-rclone.conf"
goto :loop
```

4. Buka prompt perintah Windows, cd ke `C:\Users\PhotonUser\My Files\Home Folder` jika diperlukan dan jalankan`m2-rclone.cmd`. Skrip perintah ini menjalankan loop kontinu, menyinkronkan bucket Amazon S3 Anda dan kunci ke folder lokal setiap 10 detik. Anda dapat menyesuaikan waktu habis sesuai kebutuhan. Anda akan melihat kode sumber aplikasi yang terletak di bucket Amazon S3 di Windows File Explorer.

Untuk menambahkan file baru ke set yang sedang Anda kerjakan atau untuk memperbarui file yang sudah ada, unggah file ke bucket Amazon S3 dan file tersebut akan disinkronkan ke direktori Anda pada iterasi berikutnya yang ditentukan. `m2-rclone.cmd` Demikian pula, jika Anda ingin menghapus beberapa file, hapus dari bucket Amazon S3. Operasi sinkronisasi berikutnya akan menghapusnya dari direktori lokal Anda.

Langkah 3: Buat sumber ODBC untuk instans Amazon RDS

1. Untuk memulai alat EA_admin, navigasikan ke menu pemilih aplikasi di sudut kiri atas jendela browser dan pilih MF EA_admin.
2. Dari menu Administer, pilih ODBC Data Sources, dan pilih Add dari tab User DSN.
3. Dalam kotak dialog Create New Data Source, pilih driver PostgreSQL Unicode, lalu pilih Finish.
4. Di kotak dialog Penyiapan PostgreSQL Unicode ODBC Driver (psqlodBC), tentukan dan catat nama sumber data yang Anda inginkan. Lengkapi parameter berikut dengan nilai dari instance RDS yang sebelumnya Anda buat:

   Deskripsi

   Deskripsi opsional untuk membantu Anda mengidentifikasi koneksi database ini dengan cepat.

Basis Data

Database Amazon RDS yang Anda buat sebelumnya.

Server

Titik akhir Amazon RDS.

Port

Port Amazon RDS.

Nama Pengguna

Seperti yang didefinisikan dalam instance Amazon RDS.

Kata sandi

Seperti yang didefinisikan dalam instance Amazon RDS.

5. Pilih Uji untuk memvalidasi bahwa koneksi ke Amazon RDS berhasil, lalu pilih Simpan untuk menyimpan DSN Pengguna baru Anda.
6. Tunggu hingga Anda melihat pesan yang mengonfirmasi pembuatan ruang kerja yang tepat, lalu pilih OK untuk menyelesaikan dengan Sumber Data ODBC dan tutup alat EA_admin.
7. Arahkan lagi ke menu pemilih aplikasi, dan pilih Enterprise Analyzer untuk memulai alat. Pilih Buat Baru.
8. Di jendela konfigurasi Workspace, masukkan nama ruang kerja Anda dan tentukan lokasinya. Ruang kerja dapat berupa disk berbasis Amazon S3 jika Anda bekerja di bawah konfigurasi ini, atau folder rumah Anda jika Anda mau.
9. Pilih Pilih Database Lain untuk terhubung ke instans Amazon RDS Anda.
10. Pilih ikon Postgre dari opsi, lalu pilih OK.
11. Untuk pengaturan Windows di bawah Opsi — Tentukan Parameter Koneksi, masukkan nama sumber data yang Anda buat. Masukkan juga nama database, nama skema, nama pengguna, dan kata sandi. Pilih OKE.
12. Tunggu Enterprise Analyzer untuk membuat semua tabel, indeks, dan sebagainya sehingga perlu menyimpan hasil. Proses ini mungkin memakan waktu beberapa menit. Enterprise Analyzer mengonfirmasi kapan database dan ruang kerja siap digunakan.
13. Arahkan lagi ke menu pemilih aplikasi dan pilih Enterprise Analyzer untuk memulai alat.
14. Jendela startup Enterprise Analyzer muncul di lokasi ruang kerja baru yang dipilih. Pilih Ok.

15. Arahkan ke repositori Anda di panel kiri, pilih nama repositori, dan pilih Tambahkan file/folder ke ruang kerja Anda. Pilih folder tempat kode aplikasi Anda disimpan untuk menambahkannya ke ruang kerja. Anda dapat menggunakan kode BankDemo contoh sebelumnya jika Anda mau. Saat Enterprise Analyzer meminta Anda untuk memverifikasi file-file tersebut, pilih Verifikasi untuk memulai laporan verifikasi Enterprise Analyzer awal. Mungkin perlu beberapa menit untuk menyelesaikannya, tergantung pada ukuran aplikasi Anda.
16. Perluas ruang kerja Anda untuk melihat file dan folder yang telah Anda tambahkan ke ruang kerja. Jenis objek dan laporan kompleksitas siklomatik juga terlihat di kuadran atas panel Penampil Bagan.

Anda sekarang dapat menggunakan Enterprise Analyzer untuk semua tugas yang diperlukan.

Sesi selanjutnya

1. Mulai sesi dengan AppStream 2.0 dengan URL yang Anda terima dalam pesan email selamat datang dari AppStream 2.0.
2. Masuk dengan email dan kata sandi permanen Anda.
3. Pilih tumpukan Enterprise Analyzer.
4. Luncurkan `Rclone` untuk terhubung ke disk yang didukung Amazon S3 jika Anda menggunakan opsi ini untuk berbagi file ruang kerja.
5. Luncurkan Enterprise Analyzer untuk melakukan tugas Anda.

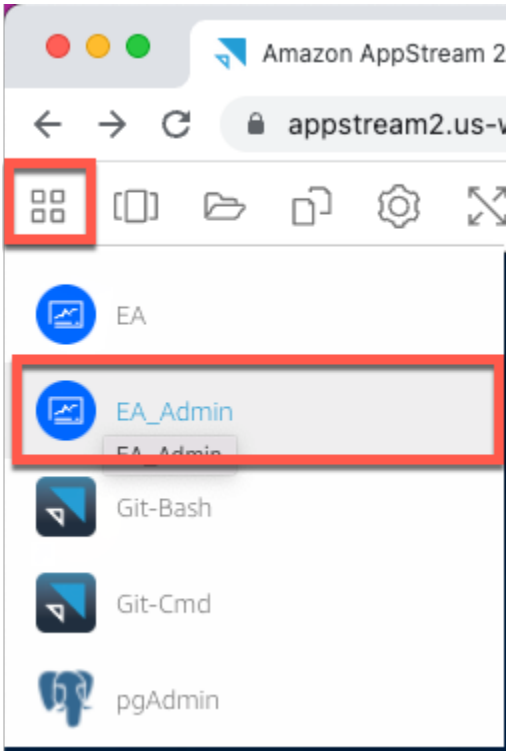
Memecahkan masalah koneksi ruang kerja

Saat Anda mencoba menyambung kembali ke ruang kerja Enterprise Analyzer, Anda mungkin melihat kesalahan seperti ini:

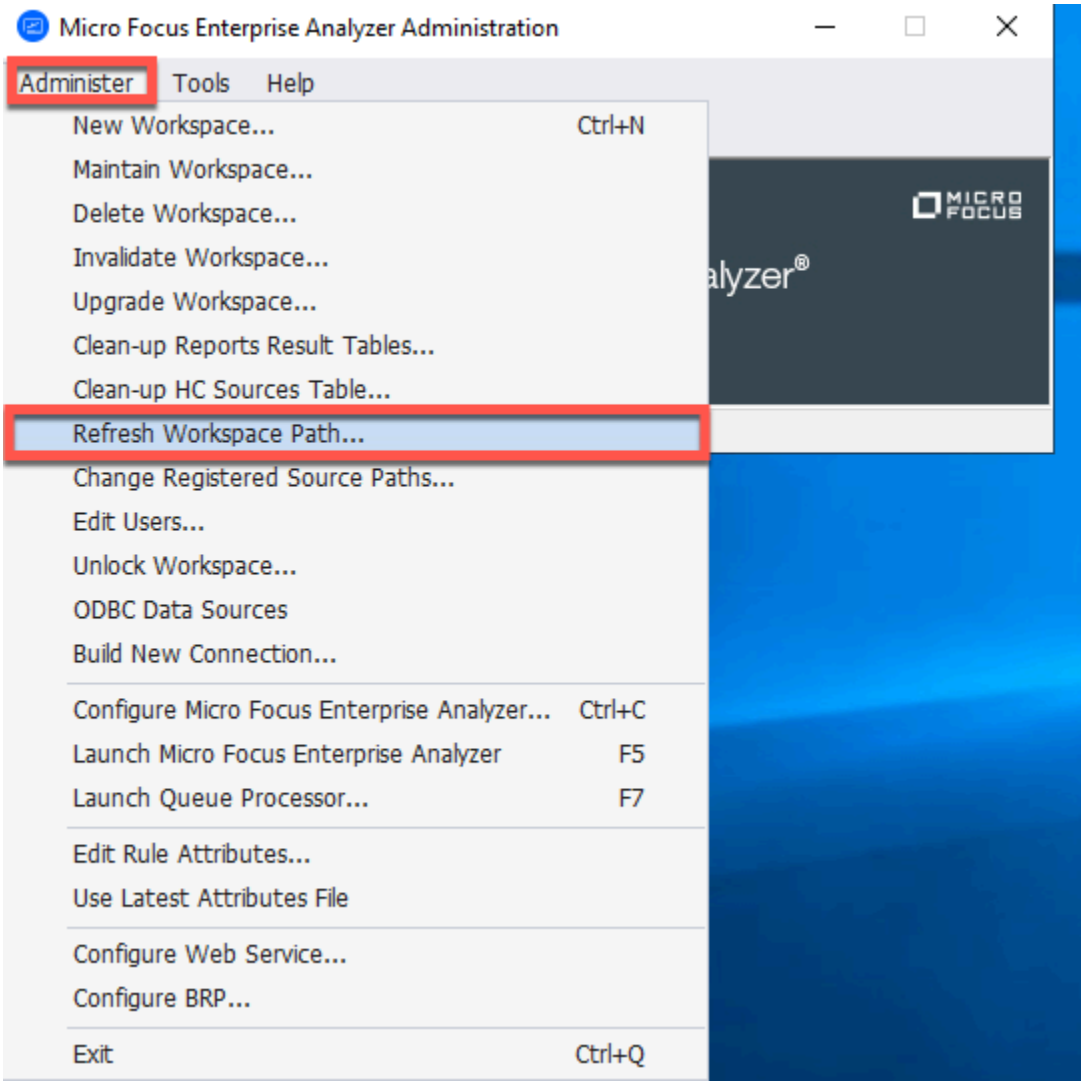
```
Cannot access the workspace directory D:\PhotonUser\My Files\Home Folder\EA_BankDemo.
 The workspace has been created on a non-shared disk of the EC2AMAZ-E6LC33H computer.
 Would you like to correct the workspace directory location?
```

Untuk mengatasi masalah ini, pilih OK untuk menghapus pesan, lalu selesaikan langkah-langkah berikut.

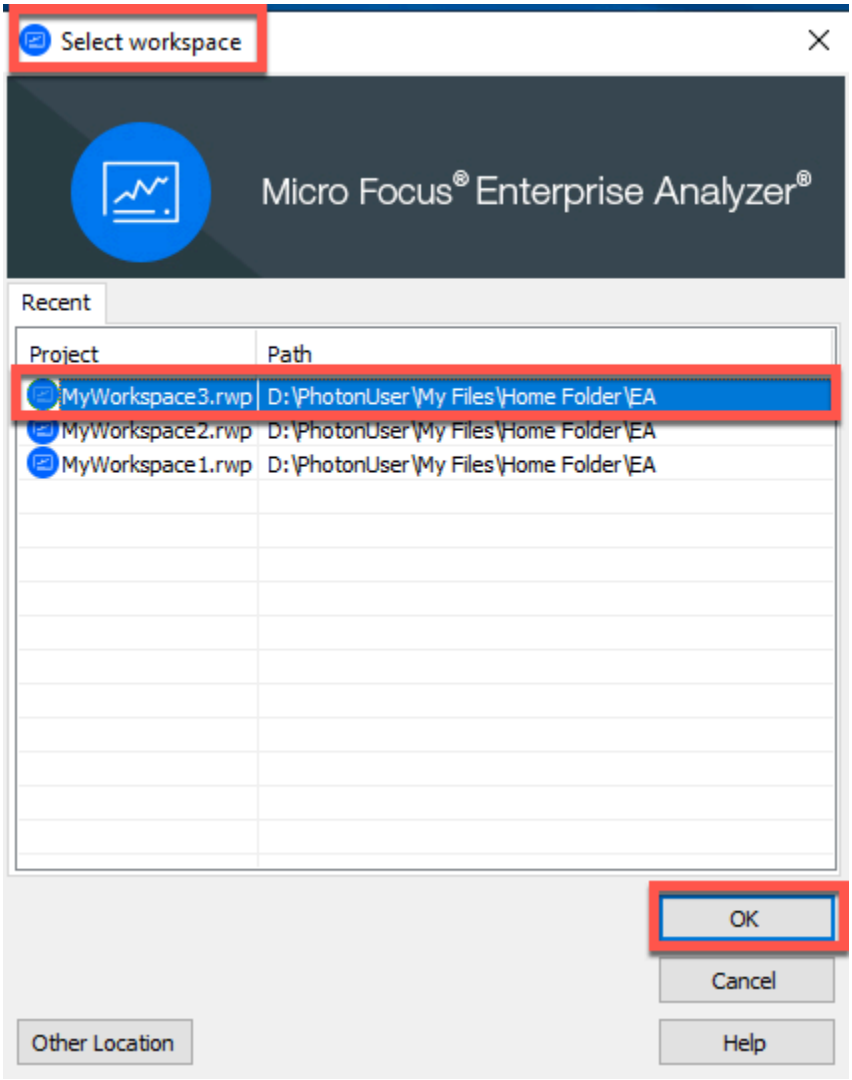
1. Di AppStream 2.0, pilih ikon Luncurkan Aplikasi pada bilah alat, lalu pilih EA_admin untuk memulai alat Administrasi Penganalisis Perusahaan Fokus Mikro.



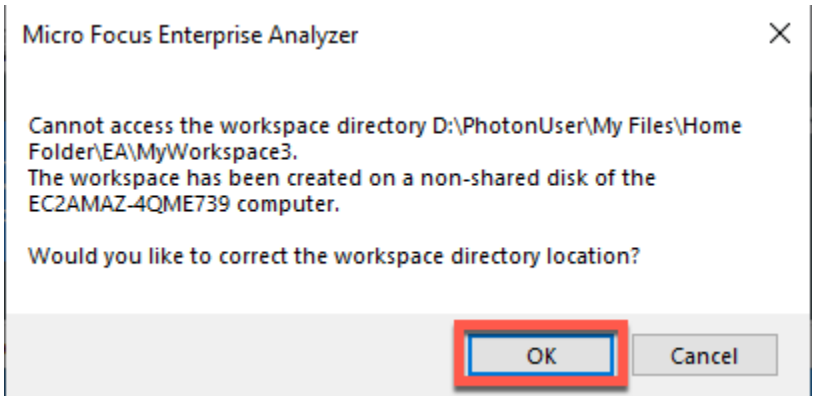
2. Dari menu Administer, pilih Refresh Workspace Path... .



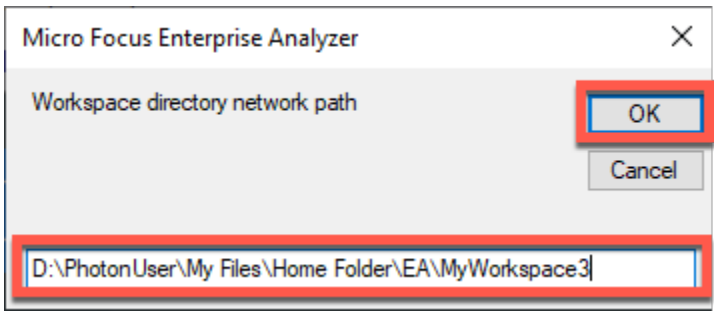
3. Di bawah Pilih ruang kerja, pilih ruang kerja yang Anda inginkan, lalu pilih OK.



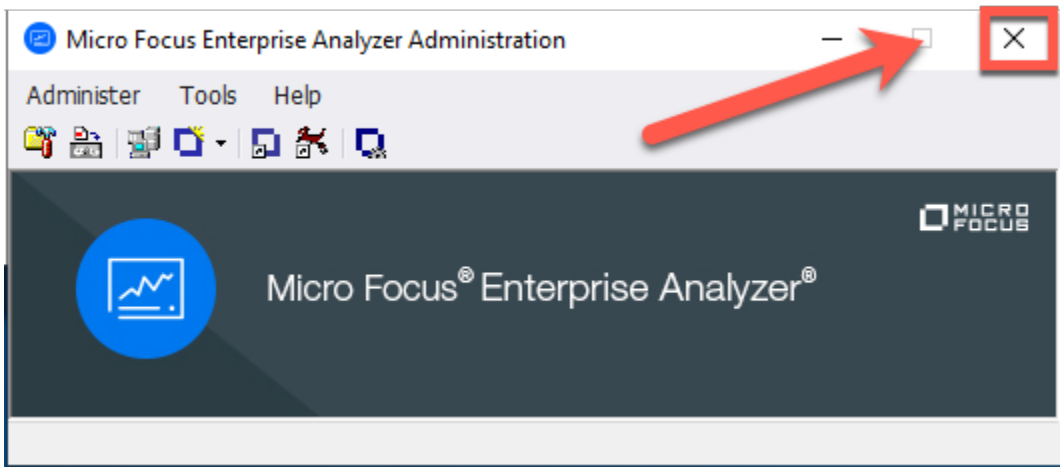
4. Pilih OK untuk mengonfirmasi pesan kesalahan.



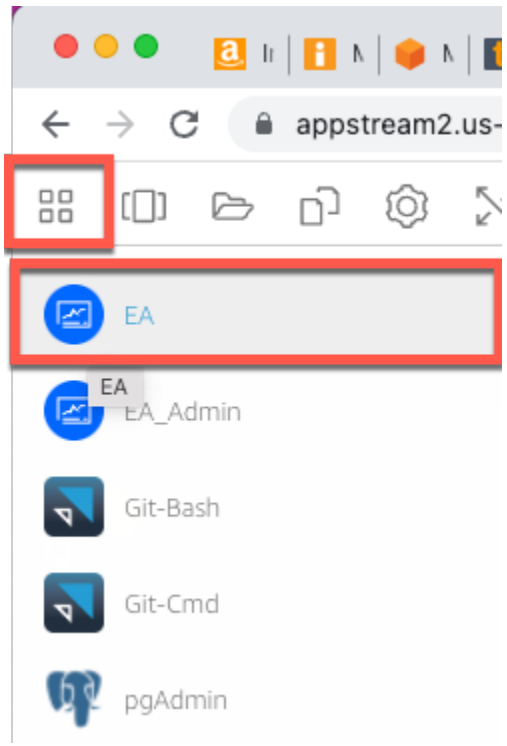
5. Di bawah jalur jaringan direktori Workspace, masukkan jalur yang benar ke ruang kerja Anda, misalnya,. `D:\PhotonUser\My Files\Home Folder\EA\MyWorkspace3`



6. Tutup alat Administrasi Penganalisis Perusahaan Fokus Mikro.



7. Di AppStream 2.0, pilih ikon Launch Application pada toolbar, dan kemudian pilih EA untuk memulai Micro Focus Enterprise Analyzer.



8. Ulangi langkah 3 - 5.

Micro Focus Enterprise Analyzer sekarang harus terbuka dengan ruang kerja yang ada.

Pembersihan sumber daya

Jika Anda tidak lagi membutuhkan sumber daya yang Anda buat untuk tutorial ini, hapus sehingga Anda tidak dikenakan biaya lebih lanjut. Selesaikan langkah-langkah berikut:

- Gunakan alat EA_admin untuk menghapus ruang kerja.
- Hapus bucket S3 yang Anda buat untuk tutorial ini. Untuk informasi selengkapnya, lihat Menghapus bucket di Panduan Pengguna Amazon S3.
- Hapus database yang Anda buat untuk tutorial ini. Untuk informasi lebih lanjut, lihat Menghapus instans DB.

Tutorial: Mengatur Pengembang Perusahaan Fokus Mikro di AppStream 2.0

Tutorial ini menjelaskan cara mengatur Micro Focus Enterprise Developer untuk satu atau lebih aplikasi mainframe untuk memelihara, mengkompilasi, dan mengujinya menggunakan fitur Enterprise Developer. Pengaturan didasarkan pada gambar Windows AppStream 2.0 yang dibagikan Modernisasi AWS Mainframe dengan pelanggan dan pada pembuatan armada AppStream 2.0 dan tumpukan seperti yang dijelaskan dalam. Tutorial: Siapkan AppStream 2.0 untuk digunakan dengan Micro Focus Enterprise Analyzer dan Micro Focus Enterprise Developer

> **Important**
>
> Langkah-langkah dalam tutorial ini mengasumsikan bahwa Anda mengatur AppStream 2.0 menggunakan AWS CloudFormation template yang dapat diunduh cfn-m2- .yaml. appstream-fleet-ea-ed Untuk informasi selengkapnya, lihat Tutorial: Siapkan AppStream 2.0 untuk digunakan dengan Micro Focus Enterprise Analyzer dan Micro Focus Enterprise Developer. Anda harus melakukan langkah-langkah pengaturan ini ketika armada dan tumpukan Pengembang Perusahaan aktif dan berjalan.

Untuk deskripsi lengkap tentang fitur dan kiriman Enterprise Developer v7, lihat dokumentasi up-to-date online-nya (v7.0) di situs Micro Focus.

Isi gambar

Selain Enterprise Developer itu sendiri, gambar berisi gambar berisi Rumba (emulator TN3270). Ini juga berisi alat dan perpustakaan berikut.

Alat pihak ketiga

- Python
- Rklon
- pGADmin
- git-scm
- Pengemudi PostgreSQL ODBC

Perpustakaan di `C:\Users\Public`

- BankDemo kode sumber dan definisi proyek untuk Pengembang Perusahaan:`m2-bankdemo-template.zip`.
- Paket instalasi MFA untuk mainframe:. `mfa.zip` Untuk informasi selengkapnya, lihat Ikhtisar Akses Mainframe di dokumentasi Pengembang Perusahaan Fokus Mikro.
- File perintah dan konfigurasi untuk Rclone (petunjuk penggunaannya dalam tutorial): dan. `m2-rclone.cmd` `m2-rclone.conf`

Jika Anda perlu mengakses kode sumber yang belum dimuat ke dalam CodeCommit repositori, tetapi tersedia di bucket Amazon S3, misalnya untuk melakukan pemuatan awal kode sumber ke git, ikuti prosedur untuk membuat disk Windows virtual seperti yang dijelaskan dalam. Tutorial: Mengatur Enterprise Analyzer pada 2.0 AppStream

Topik

- Prasyarat
- Langkah 1: Pengaturan oleh masing-masing pengguna Enterprise Developer
- Langkah 2: Buat folder virtual berbasis Amazon S3 di Windows (opsional)
- Langkah 3: Kloning repositori

- Sesi selanjutnya
- Pembersihan sumber daya

Prasyarat

- Satu atau lebih CodeCommit repositori dimuat dengan kode sumber aplikasi yang akan dipertahankan. Pengaturan repositori harus sesuai dengan persyaratan pipa CI/CD di atas untuk membuat sinergi dengan kombinasi kedua alat.
- Setiap pengguna harus memiliki kredensi ke repositori atau CodeCommit repositori yang ditentukan oleh administrator akun sesuai dengan informasi dalam Otentikasi dan kontrol akses untuk AWS. CodeCommit Struktur kredensil tersebut ditinjau dalam Otentikasi dan kontrol akses untuk AWS CodeCommit dan referensi lengkap untuk otorisasi IAM CodeCommit ada dalam referensi CodeCommit izin: administrator dapat menentukan kebijakan IAM yang berbeda untuk peran berbeda yang memiliki kredensil khusus untuk peran untuk setiap repositori dan membatasi otorisasi pengguna ke serangkaian tugas tertentu yang harus dia selesaikan pada repositori tertentu. Jadi, untuk setiap pengelola CodeCommit repositori, administrator akun akan menghasilkan pengguna utama dan memberikan izin pengguna ini untuk mengakses repositori atau repositori yang diperlukan melalui memilih kebijakan atau kebijakan IAM yang tepat untuk akses. CodeCommit

Langkah 1: Pengaturan oleh masing-masing pengguna Enterprise Developer

1. Dapatkan kredensi IAM Anda:
    1. Connect ke AWS konsol di https://console.aws.amazon.com/iam/.
    2. Ikuti prosedur yang dijelaskan pada langkah 3 Penyiapan untuk pengguna HTTPS yang menggunakan kredensi Git di AWS CodeCommitPanduan Pengguna.
    3. Salin kredensial-masuk CodeCommit khusus yang dibuat IAM untuk Anda, baik dengan menampilkan, menyalin, dan kemudian menempelkan informasi ini ke file aman di komputer lokal Anda, atau dengan memilih Unduh kredensional untuk mengunduh informasi ini sebagai file.CSV. Anda memerlukan informasi ini untuk terhubung CodeCommit.
2. Mulai sesi dengan AppStream 2.0 berdasarkan url yang diterima di email selamat datang. Gunakan email Anda sebagai nama pengguna dan buat kata sandi Anda.
3. Pilih tumpukan Enterprise Developer Anda.

4. Pada halaman menu, pilih Desktop untuk mencapai desktop Windows yang dialirkan oleh armada.

Langkah 2: Buat folder virtual berbasis Amazon S3 di Windows (opsional)

Jika ada kebutuhan untuk Rclone (lihat di atas), buat folder virtual berbasis Amazon S3 di Windows: (opsional jika semua artefak aplikasi secara eksklusif berasal dari akses). CodeCommit

> **Note**
>
> Jika Anda sudah menggunakan Rclone selama pratinjau Modernisasi AWS Mainframe, Anda harus memperbarui `m2-rclone.cmd` ke versi yang lebih baru yang terletak di. `C:\Users\Public`

1. Salin `m2-rclone.cmd` file `m2-rclone.conf` dan file yang disediakan `C:\Users\Public` ke folder rumah Anda `C:\Users\PhotonUser\My Files\Home Folder` menggunakan File Explorer.
2. Perbarui parameter `m2-rclone.conf` konfigurasi dengan kunci AWS akses Anda dan rahasia yang sesuai, serta AndaWilayah AWS.

```
[m2-s3]
type = s3
provider = AWS
access_key_id = YOUR-ACCESS-KEY
secret_access_key = YOUR-SECRET-KEY
region = YOUR-REGION
acl = private
server_side_encryption = AES256
```

3. Di `m2-rclone.cmd`, lakukan perubahan berikut:

   - Ubah `your-s3-bucket` ke nama bucket Amazon S3 Anda. Sebagai contoh, `m2-s3-mybucket`.
   - Ubah `your-s3-folder-key` ke kunci bucket Amazon S3 Anda. Sebagai contoh, `myProject`.
   - Ubah `your-local-folder-path` ke jalur direktori tempat Anda ingin file aplikasi disinkronkan dari bucket Amazon S3 yang berisi file tersebut. Sebagai contoh, `D:`

`\PhotonUser\My Files\Home Folder\m2-new`. Direktori yang disinkronkan ini harus merupakan subdirektori dari Folder Rumah agar AppStream 2.0 dapat mencadangkan dan mengembalikannya dengan benar pada awal dan akhir sesi.

```
:loop
timeout /T 10
"C:\Program Files\rclone\rclone.exe" sync m2-s3:your-s3-bucket/your-s3-folder-
key "D:\PhotonUser\My Files\Home Folder\your-local-folder-path" --config "D:
\PhotonUser\My Files\Home Folder\m2-rclone.conf"
goto :loop
```

4. Buka prompt perintah Windows, cd ke `C:\Users\PhotonUser\My Files\Home Folder` jika diperlukan dan jalankan`m2-rclone.cmd`. Skrip perintah ini menjalankan loop kontinu, menyinkronkan bucket Amazon S3 Anda dan kunci ke folder lokal setiap 10 detik. Anda dapat menyesuaikan waktu habis sesuai kebutuhan. Anda akan melihat kode sumber aplikasi yang terletak di bucket Amazon S3 di Windows File Explorer.

Untuk menambahkan file baru ke set yang sedang Anda kerjakan atau memperbarui yang sudah ada, unggah file ke bucket Amazon S3 dan file tersebut akan disinkronkan ke direktori Anda pada iterasi berikutnya yang ditentukan. `m2-rclone.cmd` Demikian pula, jika Anda ingin menghapus beberapa file, hapus dari bucket Amazon S3. Operasi sinkronisasi berikutnya akan menghapusnya dari direktori lokal Anda.

Langkah 3: Kloning repositori

1. Arahkan ke menu pemilih aplikasi di sudut kiri atas jendela browser dan pilih Enterprise Developer.
2. Selesaikan pembuatan ruang kerja yang dibutuhkan oleh Enterprise Developer di folder Home Anda dengan memilih `C:\Users\PhotonUser\My Files\Home Folder` (alias`D:\PhotonUser\My Files\Home Folder`) sebagai lokasi untuk ruang kerja.
3. Di Enterprise Developer, kloning CodeCommit repositori Anda dengan pergi ke Project Explorer, klik kanan dan pilih Import, Import..., Git, Projects from Git Clone URI. Kemudian, masukkan kredensi masuk CodeCommit -spesifik Anda dan selesaikan dialog Eclipse untuk mengimpor kode.

Repositori CodeCommit git sekarang dikloning di ruang kerja lokal Anda.

Ruang kerja Enterprise Developer Anda sekarang siap untuk memulai pekerjaan pemeliharaan pada aplikasi Anda. Secara khusus, Anda dapat menggunakan instance lokal Microfocus Enterprise Server (ES) yang terintegrasi dengan Enterprise Developer untuk secara interaktif men-debug dan menjalankan aplikasi Anda untuk memvalidasi perubahan Anda secara lokal.

> **Note**
>
> Lingkungan Enterprise Developer lokal, termasuk instance Enterprise Server lokal, berjalan di bawah Windows sementara Modernisasi AWS Mainframe berjalan di Linux. Kami menyarankan Anda menjalankan pengujian komplementer di lingkungan Linux yang disediakan oleh Modernisasi AWS Mainframe setelah Anda melakukan aplikasi baru CodeCommit dan membangunnya kembali untuk target ini dan sebelum Anda meluncurkan aplikasi baru ke produksi.

Sesi selanjutnya

Saat Anda memilih folder yang berada di bawah manajemen AppStream 2.0 seperti folder rumah untuk kloning CodeCommit repositori Anda, folder tersebut akan disimpan dan dipulihkan secara transparan di seluruh sesi. Selesaikan langkah-langkah berikut saat berikutnya Anda perlu bekerja dengan aplikasi:

1. Mulai sesi dengan AppStream 2.0 berdasarkan url yang diterima di email selamat datang.
2. Login dengan email dan kata sandi permanen Anda.
3. Pilih tumpukan Enterprise Developer.
4. Luncurkan `Rclone` untuk menghubungkan (lihat di atas) ke disk yang didukung Amazon S3 saat opsi ini digunakan untuk berbagi file ruang kerja.
5. Luncurkan Enterprise Developer untuk melakukan pekerjaan Anda.

Pembersihan sumber daya

Jika Anda tidak lagi membutuhkan sumber daya yang Anda buat untuk tutorial ini, hapuslah sehingga Anda tidak akan terus dikenakan biaya untuk itu. Selesaikan langkah-langkah berikut:

- Hapus CodeCommit reponsitory yang Anda buat untuk tutorial ini. Untuk informasi selengkapnya, lihat Menghapus CodeCommit repositori di AWS CodeCommitPanduan Pengguna.

- Hapus database yang Anda buat untuk tutorial ini. Untuk informasi lebih lanjut, lihat Menghapus instans DB.

Mengatur Otomasi untuk Micro Focus Enterprise Analyzer dan Micro Focus Enterprise Developer Streaming Sesi

Anda dapat secara otomatis menjalankan skrip di awal sesi dan akhir untuk memungkinkan otomatisasi yang spesifik untuk konteks pelanggan Anda. Untuk informasi selengkapnya tentang fitur AppStream 2.0 ini, lihat Menggunakan Skrip Sesi untuk Mengelola Pengalaman Streaming Pengguna AppStream 2.0 Anda di Panduan Administrasi Amazon AppStream 2.0.

Fitur ini mengharuskan Anda memiliki setidaknya versi gambar Enterprise Analyzer dan Enterprise Developer berikut:

- `m2-enterprise-analyzer-v8.0.4.R1`
- `m2-enterprise-developer-v8.0.4.R1`

Topik

- Siapkan otomatisasi saat sesi dimulai
- Mengatur otomatisasi di akhir sesi

Siapkan otomatisasi saat sesi dimulai

Jika Anda ingin menjalankan skrip otomatisasi saat pengguna terhubung ke AppStream 2.0, buat skrip Anda dan beri nama`m2-user-setup.cmd`. Simpan skrip di folder AppStream 2.0 Home untuk pengguna. Gambar AppStream 2.0 yang disediakan oleh ModernisasiAWS Mainframe mencari skrip dengan nama itu di lokasi itu, dan menjalankannya jika ada.

> **Note**
>
> Durasi skrip tidak dapat melebihi batas yang ditetapkan oleh AppStream 2.0, yang saat ini 60 detik. Untuk informasi selengkapnya, lihat Menjalankan Skrip Sebelum Sesi Streaming Dimulai di Panduan Administrasi Amazon AppStream 2.0.

Mengatur otomatisasi di akhir sesi

Jika Anda ingin menjalankan skrip otomatisasi saat pengguna memutuskan sambungan dari AppStream 2.0, buat skrip Anda dan beri namam2-user-teardown.cmd. Simpan skrip di folder AppStream 2.0 Home untuk pengguna. Gambar AppStream 2.0 yang disediakan oleh ModernisasiAWS Mainframe mencari skrip dengan nama itu di lokasi itu, dan menjalankannya jika ada.

> **Note**
>
> Durasi skrip tidak dapat melebihi batas yang ditetapkan oleh AppStream 2.0, yang saat ini 60 detik. Untuk informasi selengkapnya, lihat Menjalankan Skrip Setelah Sesi Streaming Berakhir di Panduan Administrasi Amazon AppStream 2.0.

Lihat Kumpulan Data sebagai Tabel dan Kolom di Pengembang Perusahaan

Anda dapat mengakses kumpulan data mainframe yang digunakan dalam Modernisasi AWS Mainframe menggunakan runtime Micro Focus. Anda dapat melihat kumpulan data yang dimigrasi sebagai tabel dan kolom dari instance Pengembang Perusahaan Fokus Mikro. Melihat kumpulan data dengan cara ini memungkinkan Anda untuk:

- Lakukan SQL SELECT operasi pada file data yang dimigrasi.
- Paparkan data di luar aplikasi mainframe yang dimigrasi tanpa mengubah aplikasi.
- Mudah memfilter data dan menyimpan sebagai CSV atau format file lainnya.

> **Note**
>
> Langkah 1 dan 2 adalah kegiatan satu kali. Ulangi langkah 3 dan 4 untuk setiap kumpulan data untuk membuat tampilan database.

Topik

- Prasyarat

Prasyarat

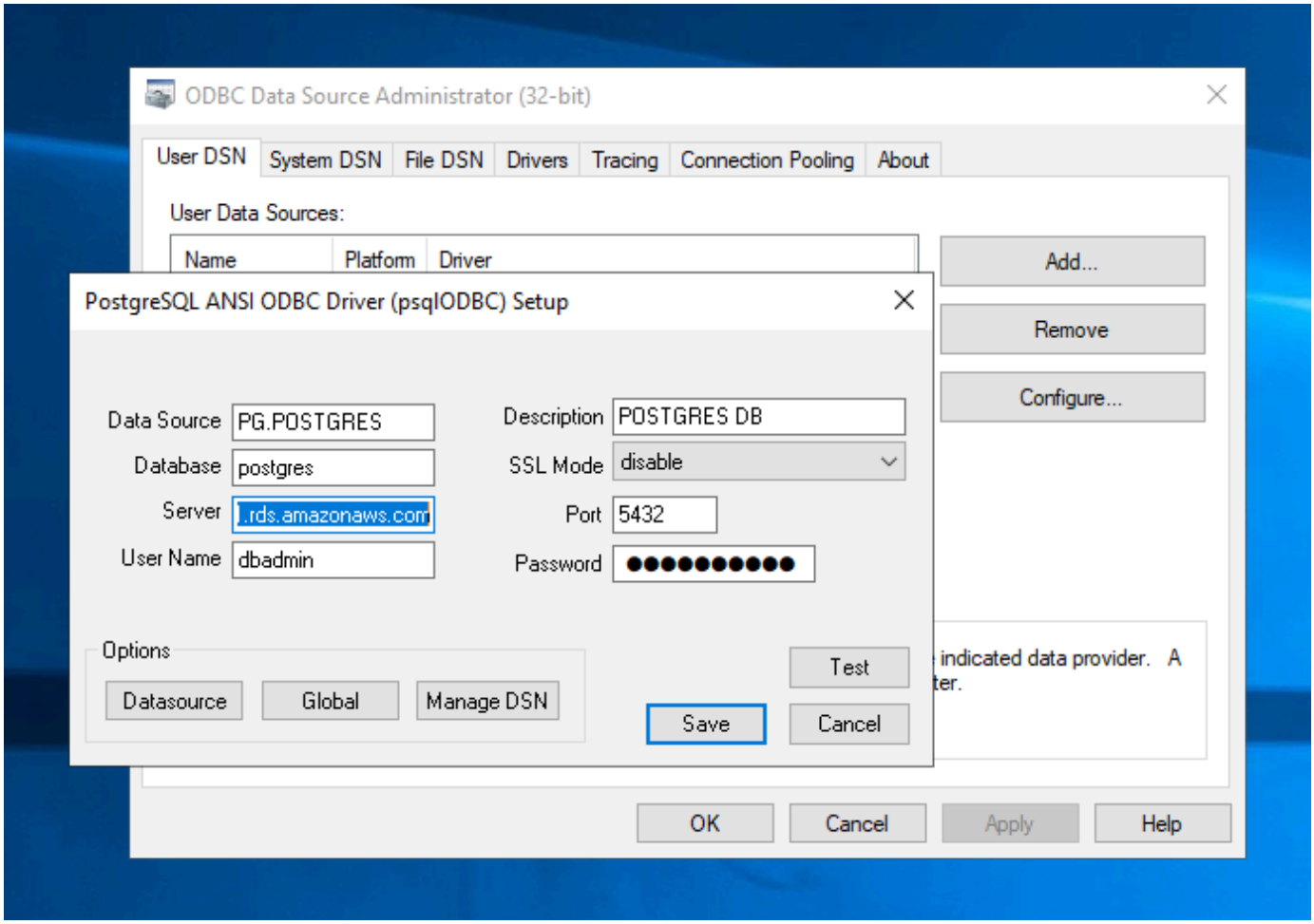
- Anda harus memiliki akses ke Micro Focus Enterprise Developer Desktop melalui AppStream 2.0.
- Anda harus memiliki aplikasi yang digunakan dan berjalan di bawah Modernisasi AWS Mainframe menggunakan mesin runtime Micro Focus.
- Anda menyimpan data aplikasi Anda di Aurora PostgreSQL Edisi yang kompatibel.

Langkah 1: Siapkan Koneksi ODBC ke Datastore Micro Focus (database Amazon RDS)

Pada langkah ini, Anda mengatur koneksi ODBC ke database yang berisi data yang ingin Anda lihat sebagai tabel dan kolom. Ini adalah langkah satu kali saja.

1. Masuk ke Micro Focus Enterprise Developer Desktop menggunakan URL streaming AppStream 2.0.
2. Buka Administrator Sumber Data ODBC, pilih DSN Pengguna dan kemudian pilih Tambah.
3. Di Create New Data Source, pilih PostgreSQL ANSI dan kemudian pilih Finish.
4. Buat sumber data `PG.POSTGRES` dengan menyediakan informasi database yang diperlukan, sebagai berikut:

```
Data Source : PG.POSTGRES
Database    : postgres
Server      : rds_endpoint.rds.amazonaws.com
Port        : 5432
User Name   : user_name
Password    : user_password
```



5. Pilih Uji untuk memastikan koneksi berfungsi. Anda akan melihat pesan `Connection successful` jika tes berhasil.

   Jika tes tidak berhasil, tinjau informasi berikut.

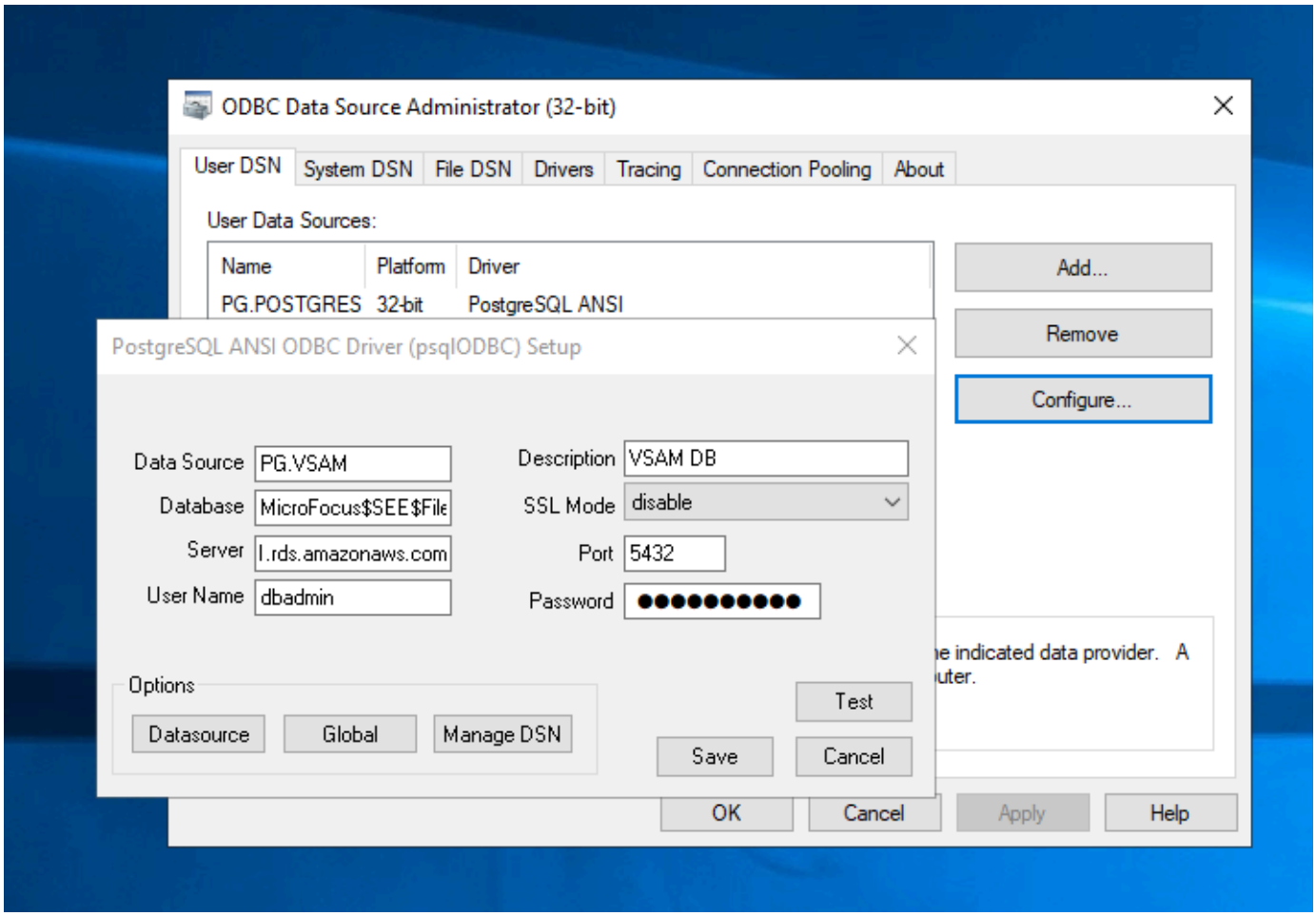
   - Pemecahan masalah untuk Amazon RDS
   - Bagaimana cara mengatasi masalah saat menghubungkan ke instans Amazon RDS DB saya?

6. Simpan sumber data.

7. Buat sumber data untuk`PG.VSAM`, uji koneksi, dan simpan sumber data. Berikan informasi database berikut:

```
Data Source : PG.VSAM
Database    : MicroFocus$SEE$Files$VSAM
Server      : rds_endpoint.rds.amazonaws.com
Port        : 5432
User Name   : user_name
```

```
Password    : user_password
```



Langkah 2: Buat file MFDBFH.cfg

Pada langkah ini, Anda membuat file konfigurasi yang menjelaskan penyimpanan data Micro Focus. Ini adalah langkah konfigurasi satu kali saja.

1. Di Folder Beranda Anda, misalnya`D:\PhotonUser\My Files\Home Folder\MFED\cfg\MFDBFH.cfg`, di, buat file MFDBFH.cfg dengan konten berikut.

```
<datastores>
       <server name="ESPACDatabase" type="postgresql" access="odbc">
        <dsn name="PG.POSTGRES" type="database" dbname="postgres"/>
        <dsn name="PG.VSAM" type="datastore" dsname="VSAM"/>
       </server>
      </datastores>
```

2. Verifikasi konfigurasi MFDBFH dengan menjalankan perintah berikut untuk menanyakan datastore Micro Focus:

```
*###*
*## Test the connection by running the following commands*
*###*

set MFDBFH_CONFIG="D:\PhotonUser\My Files\Home Folder\MFED\cfg\MFDBFH.cfg"

dbfhdeploy list sql://ESPACDatabase/VSAM?folder=/DATA
```

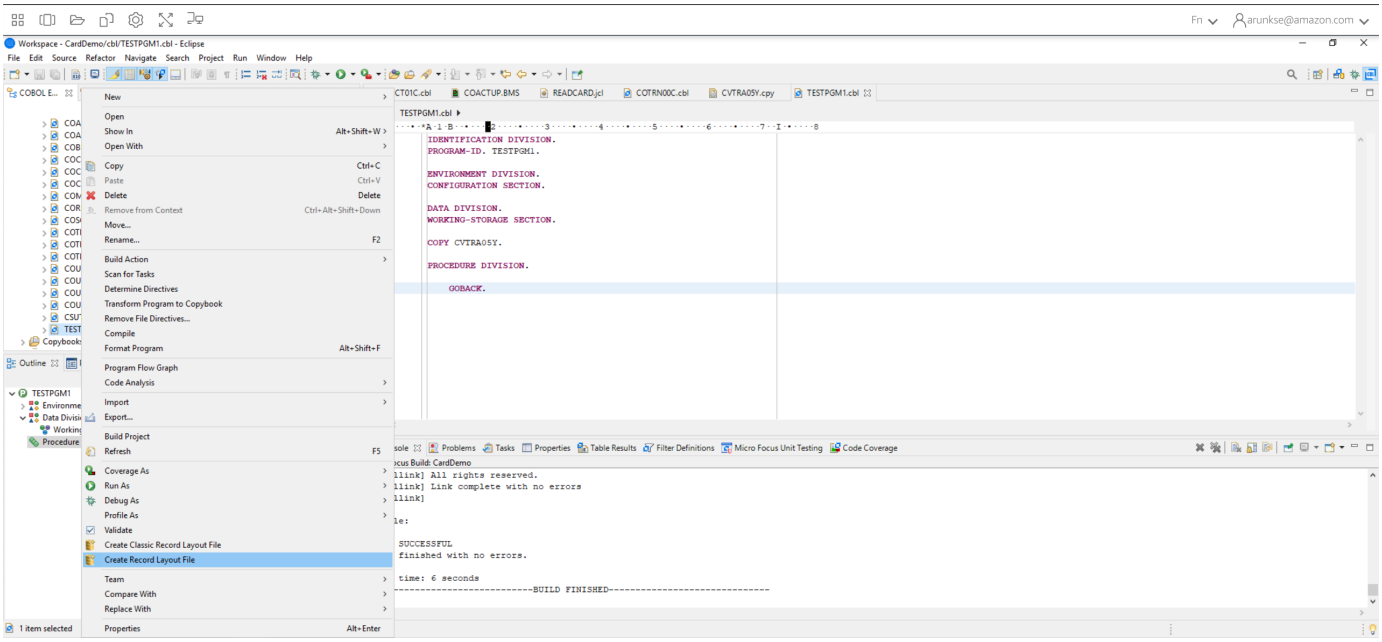
Langkah 3: Buat file struktur (STR) untuk tata letak copybook Anda

Pada langkah ini, Anda membuat file struktur untuk tata letak copybook Anda sehingga Anda dapat menggunakannya nanti untuk membuat tampilan database dari kumpulan data.

1. Kompilasi program yang terkait dengan copybook Anda. Jika tidak ada program yang menggunakan copybook, buat dan kompilasi program sederhana seperti berikut ini dengan pernyataan COPY untuk copybook Anda.

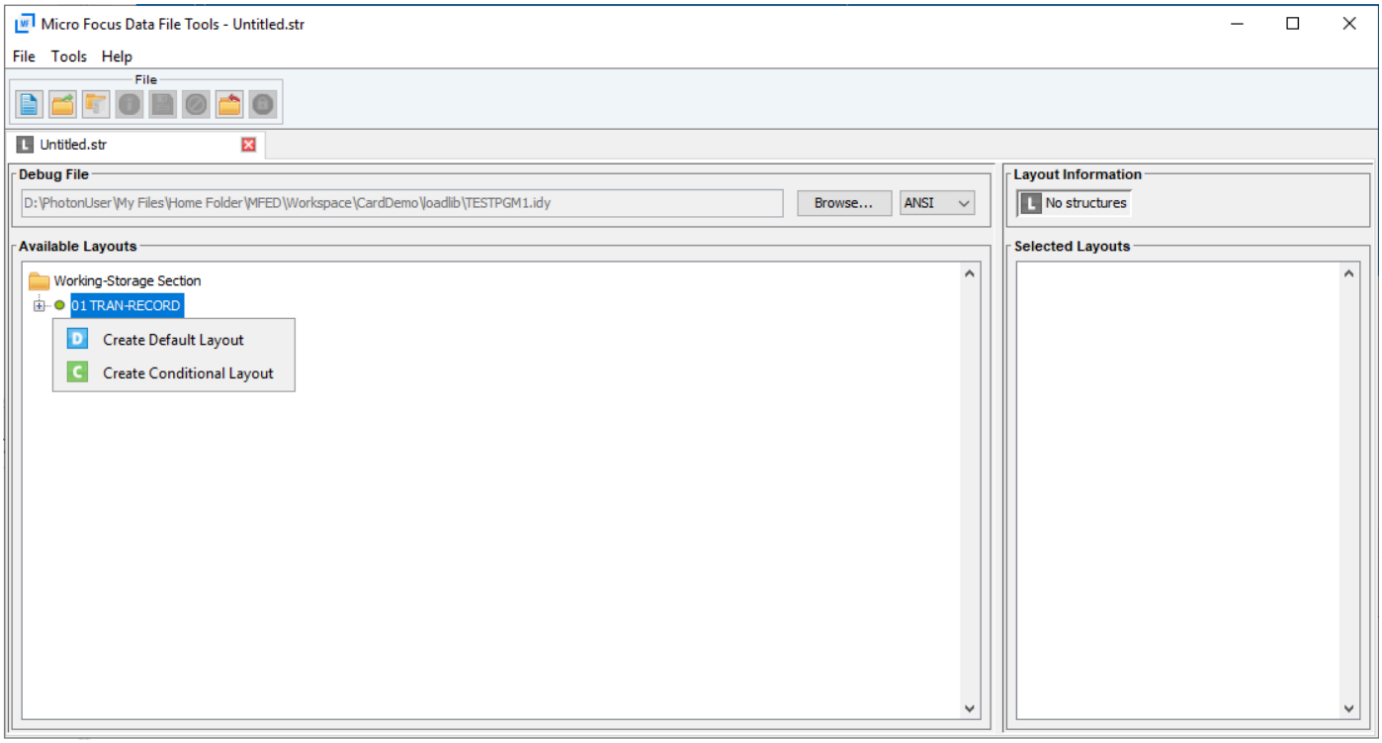
```
IDENTIFICATION DIVISION.
      PROGRAM-ID. TESTPGM1.

      ENVIRONMENT DIVISION.
      CONFIGURATION SECTION.

      DATA DIVISION.
      WORKING-STORAGE SECTION.

      COPY CVTRA05Y.

      PROCEDURE DIVISION.

      GOBACK.
```

2. Setelah kompilasi berhasil, klik kanan pada program dan pilih Create Record Layout File. Ini akan membuka Alat File Data Fokus Mikro menggunakan file.idy yang dihasilkan selama kompilasi.



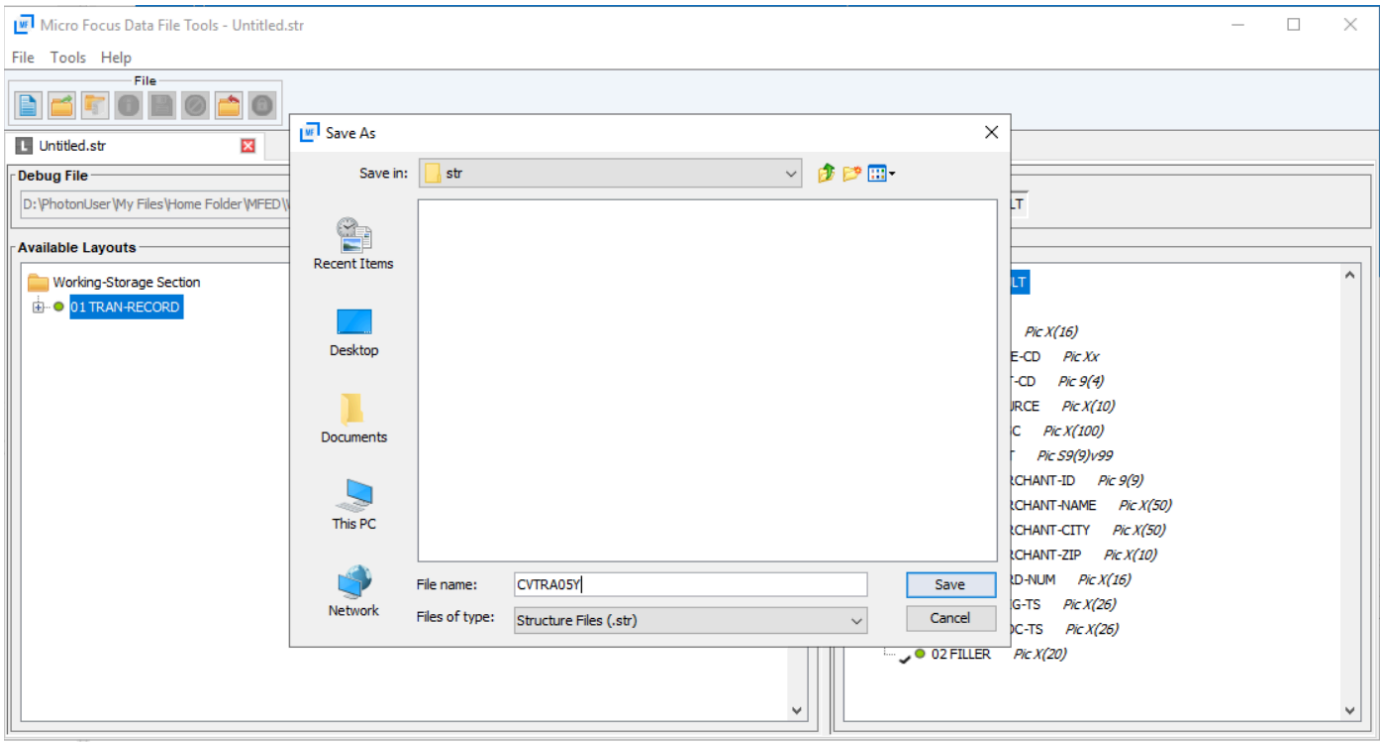
3. Klik kanan pada struktur Rekam dan pilih Buat Tata Letak Default (struktur tunggal) atau Buat Tata Letak Bersyarat (multi struktur) tergantung pada tata letaknya.

   Untuk informasi selengkapnya, lihat Membuat File Struktur dan Tata Letak dalam dokumentasi Fokus Mikro.



4. Setelah membuat tata letak, pilih File dari menu dan kemudian pilih Save As. Jelajahi dan simpan file di bawah Folder Beranda Anda dengan nama file yang sama dengan buku salinan

Anda. Anda dapat memilih untuk membuat folder bernama `str` dan menyimpan semua file struktur Anda di sana.



Langkah 4: Buat tampilan database menggunakan file struktur (STR)

Pada langkah ini, Anda menggunakan file struktur yang dibuat sebelumnya untuk membuat tampilan database untuk kumpulan data.

- Gunakan `dbfhview` perintah untuk membuat tampilan database untuk kumpulan data yang sudah ada di Datastore Micro Focus seperti yang ditunjukkan pada contoh berikut.

```
##
      ## The below command creates database view for VSAM file
 AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS
      ## using the STR file CVTRA05Y.str
      ##

      dbfhview -create -struct:"D:\PhotonUser\My Files\Home Folder\MFED\str
\CVTRA05Y.str" -name:V_AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS.DAT -file:sql://
ESPACDatabase/VSAM/AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS.DAT?folder=/DATA

      ##
      ## Output:
```

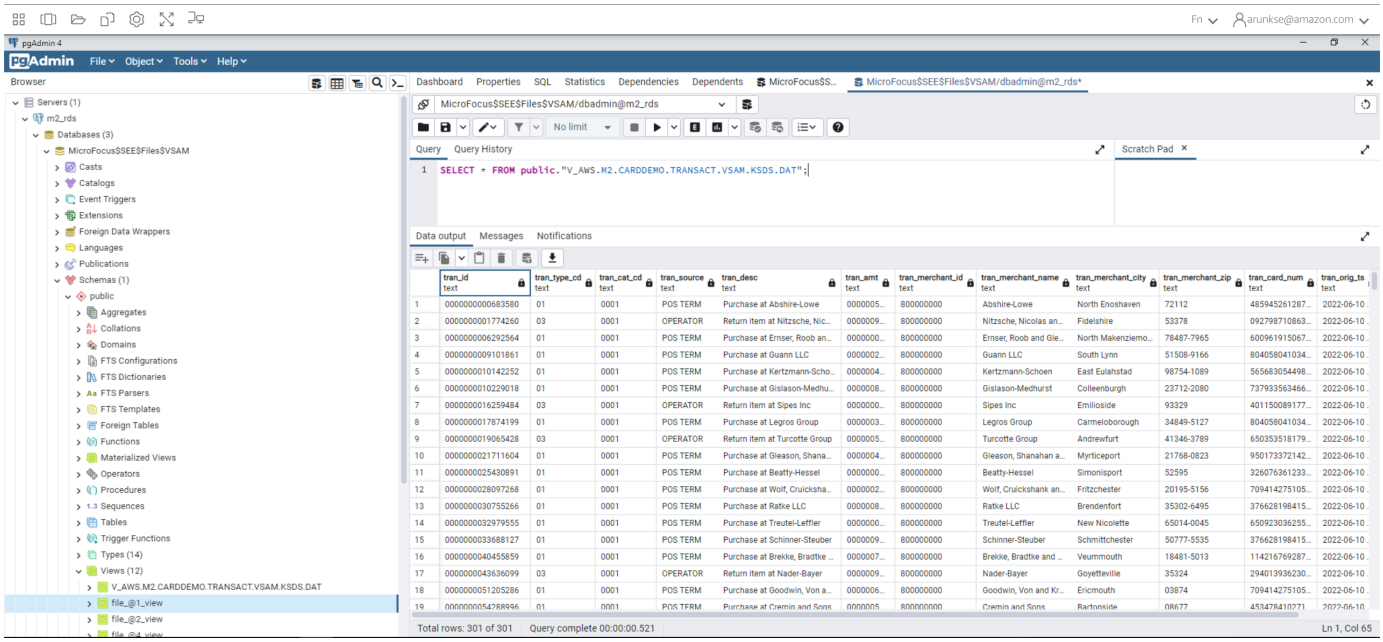
```
      ##

      Micro Focus Database File Handler - View Generation Tool Version 8.0.00
      Copyright (C) 1984-2022 Micro Focus. All rights reserved.

      VGN0017I Using structure definition 'TRAN-RECORD-DEFAULT'
      VGN0022I View 'V_AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS.DAT' installed in
datastore 'sql://espacdatabase/VSAM'
      VGN0002I The operation completed successfully
```

Langkah 5: Lihat kumpulan data Fokus Mikro sebagai tabel dan kolom

Pada langkah ini, sambungkan ke database menggunakan pgAdmin sehingga Anda dapat menjalankan kueri untuk melihat kumpulan data seperti tabel dan kolom.

- Connect ke database `MicroFocus$SEE$Files$VSAM` menggunakan pgAdmin dan kueri tampilan database yang Anda buat di langkah 4.

```
SELECT * FROM public."V_AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS.DAT";
```



Tutorial: Gunakan template dengan Micro Focus Enterprise Developer

Tutorial ini menjelaskan bagaimana menggunakan template dan proyek yang telah ditetapkan dengan Micro Focus Enterprise Developer. Ini mencakup tiga kasus penggunaan. Semua kasus penggunaan menggunakan kode contoh yang disediakan di BankDemo Sampel. Untuk mengunduh sampel, pilih`bankdemo.zip`.

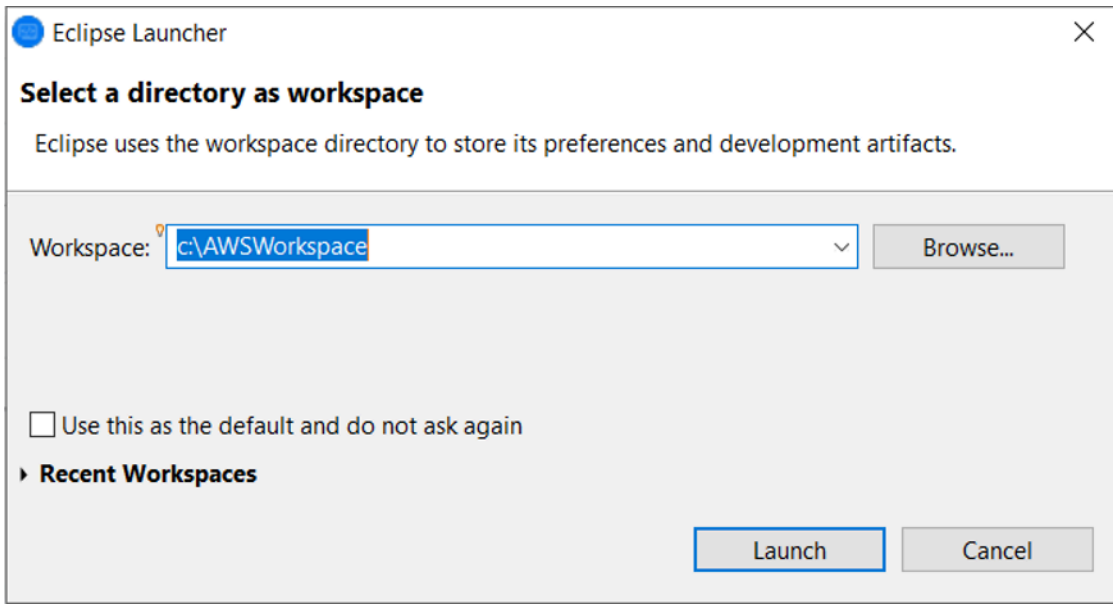
> **Important**
>
> Jika Anda menggunakan versi Enterprise Developer untuk Windows, biner yang dihasilkan oleh compiler dapat berjalan hanya pada Enterprise Server disediakan dengan Enterprise Developer. Anda tidak dapat menjalankannya di bawahAWSMainframe Modernisasi runtime, yang didasarkan pada Linux.

Topik

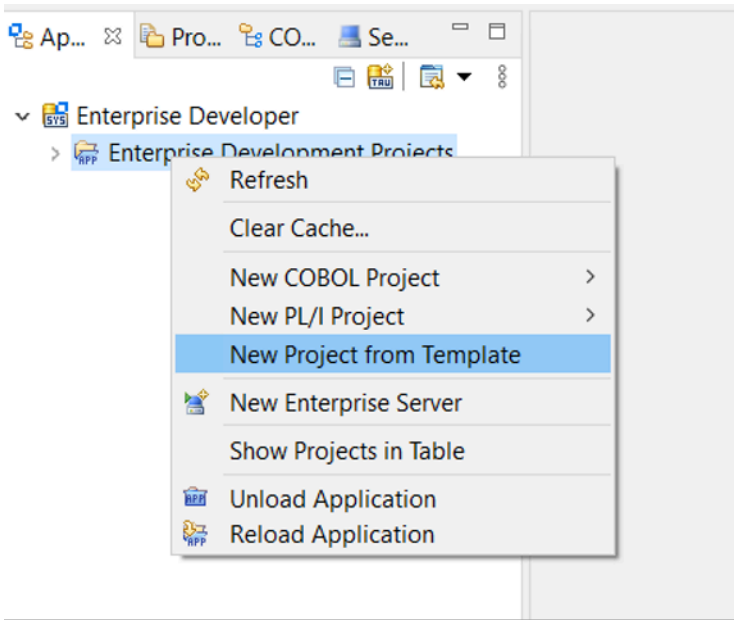
Use Case 1 - Menggunakan Template Proyek COBOL yang berisi komponen sumber

Kasus penggunaan ini mengharuskan Anda untuk menyalin komponen sumber ke dalam struktur direktori Template sebagai bagian dari langkah pengaturan pra demo. Di`bankdemo.zip`ini telah diubah dari aslinya`AWSTemplates.zip`pengiriman untuk menghindari dua salinan sumbernya.

1. Mulai Enterprise Developer dan tentukan ruang kerja yang dipilih.

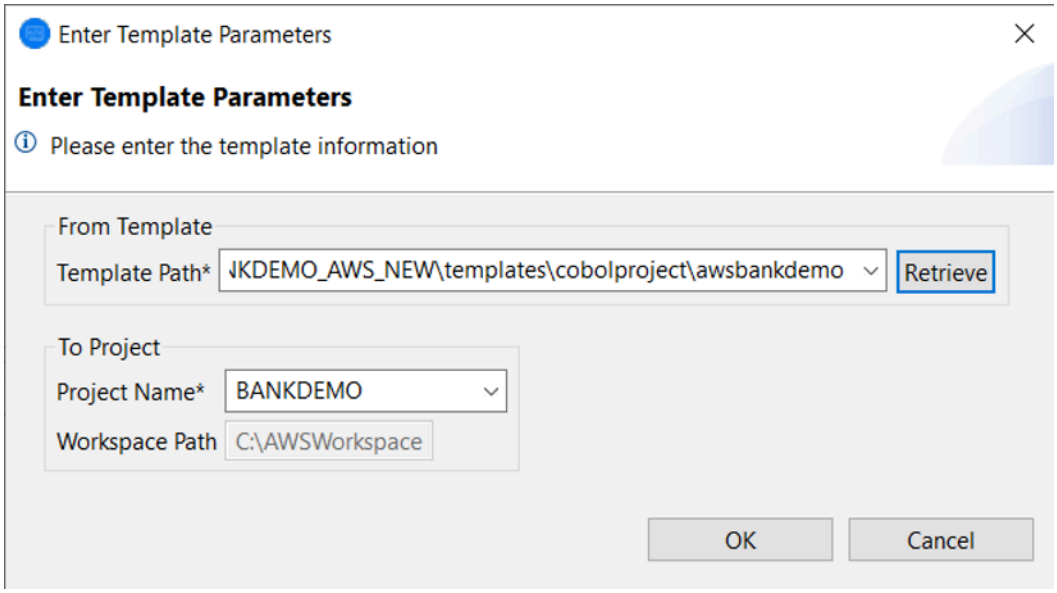


2. DALAMPenjelajah Aplikasiview, dariProyek Pengembangan Perusahaanitem tampilan pohon, pilihProyek Baru dari Templatedari menu konteks.

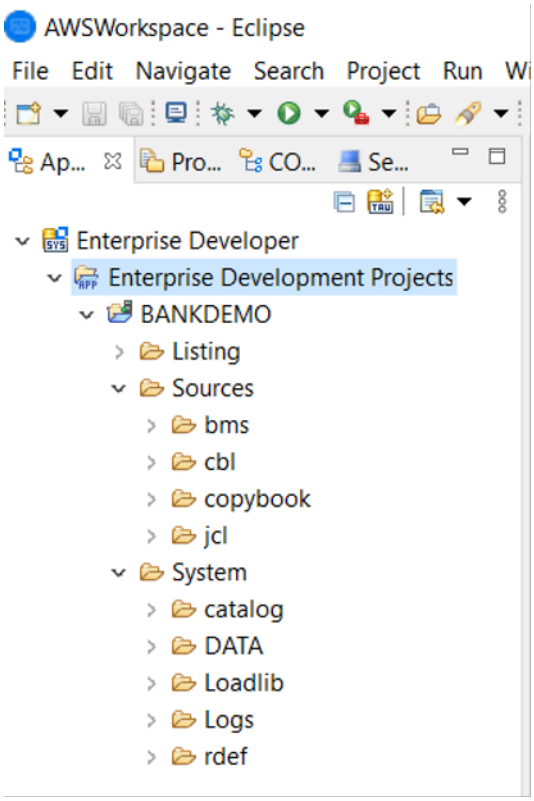


3. Masukkan parameter template seperti yang ditunjukkan.

> **Note**
>
> Jalur Template akan merujuk ke tempat ZIP diekstraksi.



4. Memilih OK akan membuat proyek Eclipse pengembangan lokal berdasarkan template yang disediakan, dengan sumber lengkap dan struktur lingkungan eksekusi.



Yang`System`struktur berisi file definisi sumber daya lengkap dengan entri yang diperlukan untuk BANKDEMO, katalog yang diperlukan dengan entri ditambahkan dan file data ASCII yang sesuai.

Karena struktur template sumber berisi semua item sumber, file-file ini disalin ke proyek lokal dan oleh karena itu secara otomatis dibangun di Enterprise Developer.

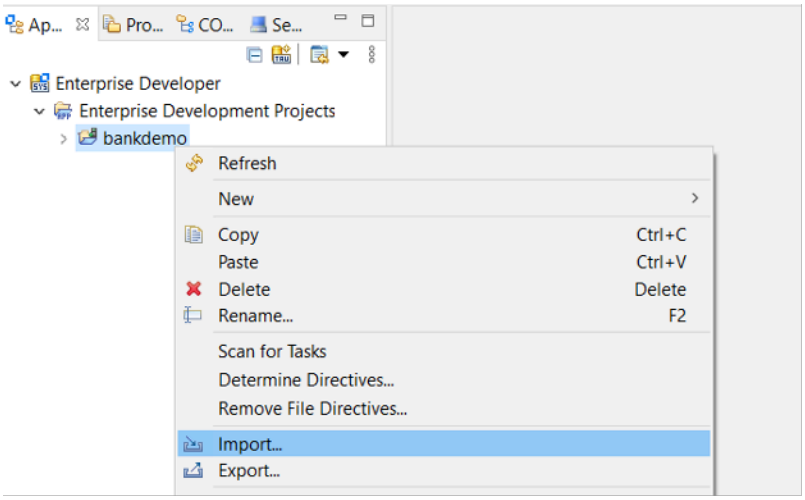
Use Case 2 - Menggunakan Template Proyek COBOL tanpa komponen sumber

Langkah 1 sampai 3 identik denganUse Case 1 - Menggunakan Template Proyek COBOL yang berisi komponen sumber.

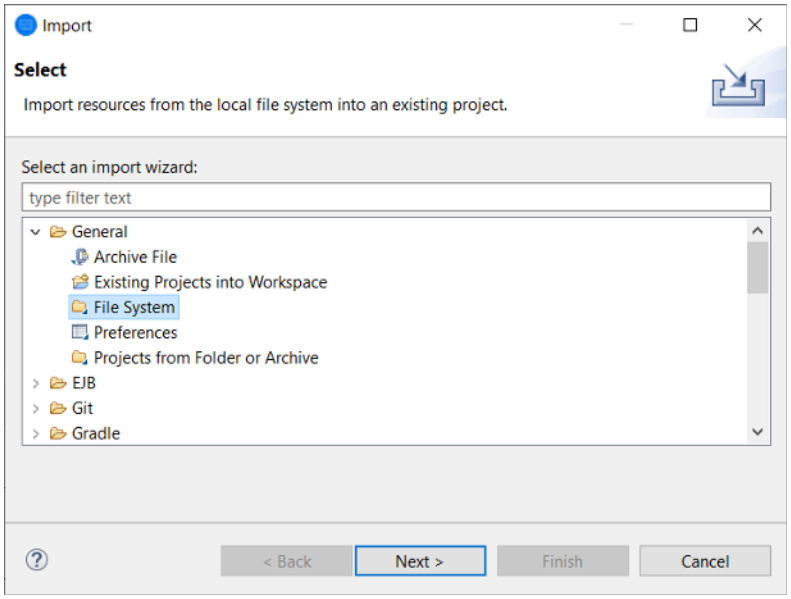
Yang`Systems`struktur dalam kasus penggunaan ini juga berisi file definisi sumber daya lengkap dengan entri yang diperlukan untuk BankDemo, katalog yang diperlukan dengan entri ditambahkan, dan file data ASCII yang sesuai.

Namun, struktur sumber template tidak mengandung komponen apa pun. Anda harus mengimpor ini ke dalam proyek dari repositori sumber apa pun yang Anda gunakan.

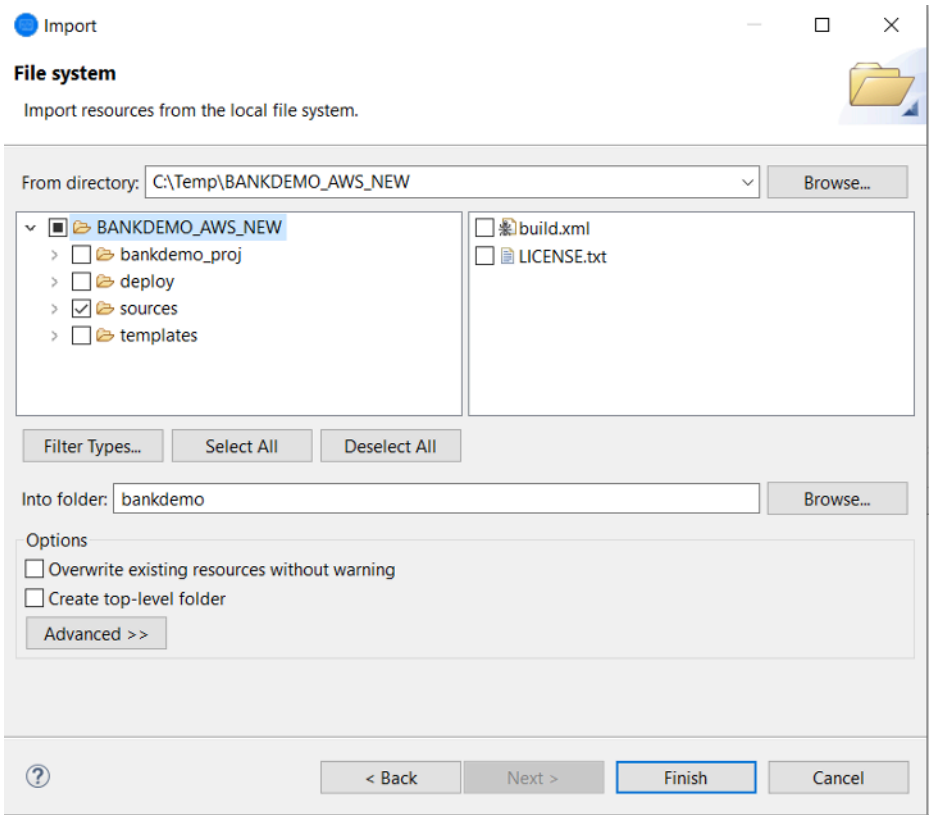
1. Pilih nama proyek. Dari menu konteks terkait, pilihImpor.



2. Dari dialog yang dihasilkan, di bawahUmumbagian, pilihSistem filedan kemudian pilih Berikutnya.



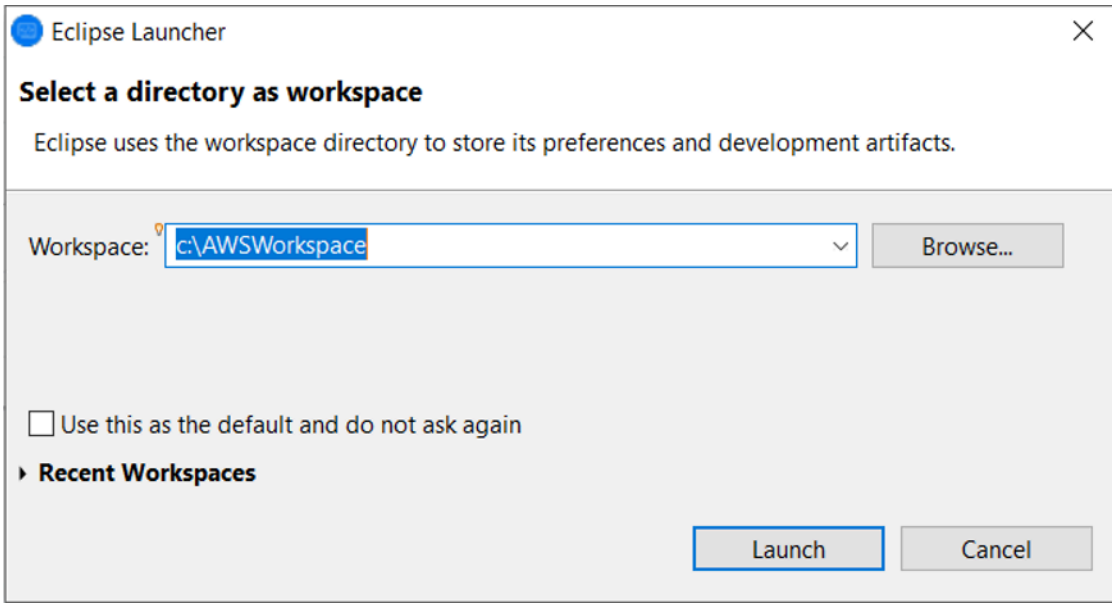
3. MengisiDari direktoribidang dengan browsing sistem file untuk menunjuk ke folder repositori. Pilih semua folder yang ingin Anda impor, seperti`sources`. Yang`Into folder`lapangan akan dihuni sebelumnya. Pilih Selesai.



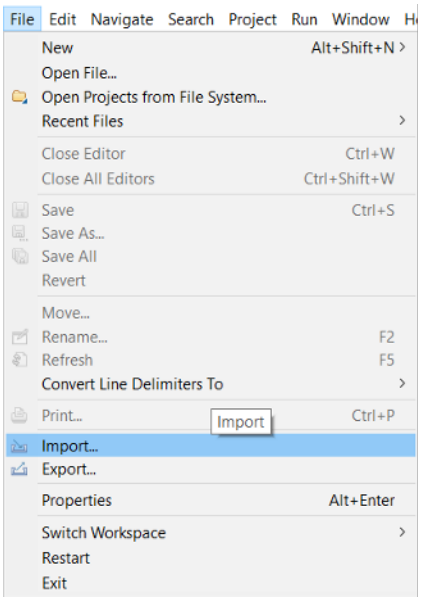
Setelah struktur template sumber berisi semua item sumber, mereka dibangun secara otomatis di Enterprise Developer.

Use Case 3 - Menggunakan proyek COBOL yang telah ditetapkan menghubungkan ke folder sumber

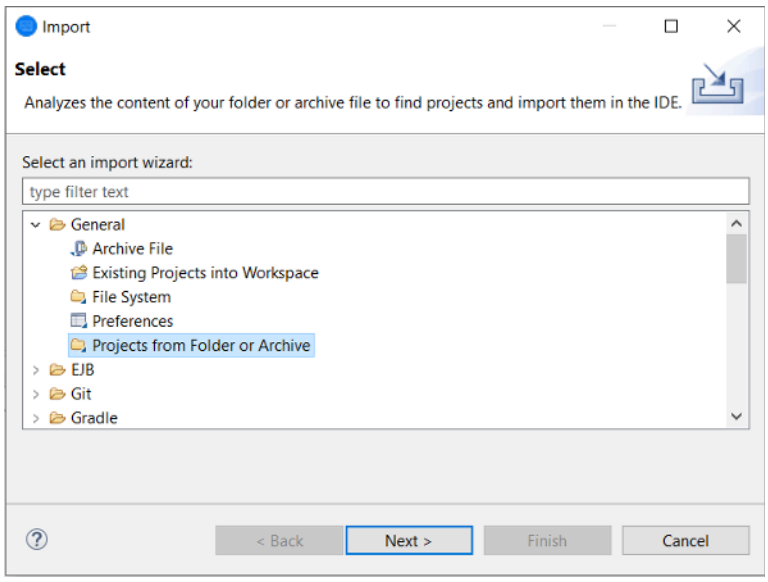
1. Mulai Enterprise Developer dan tentukan ruang kerja yang dipilih.



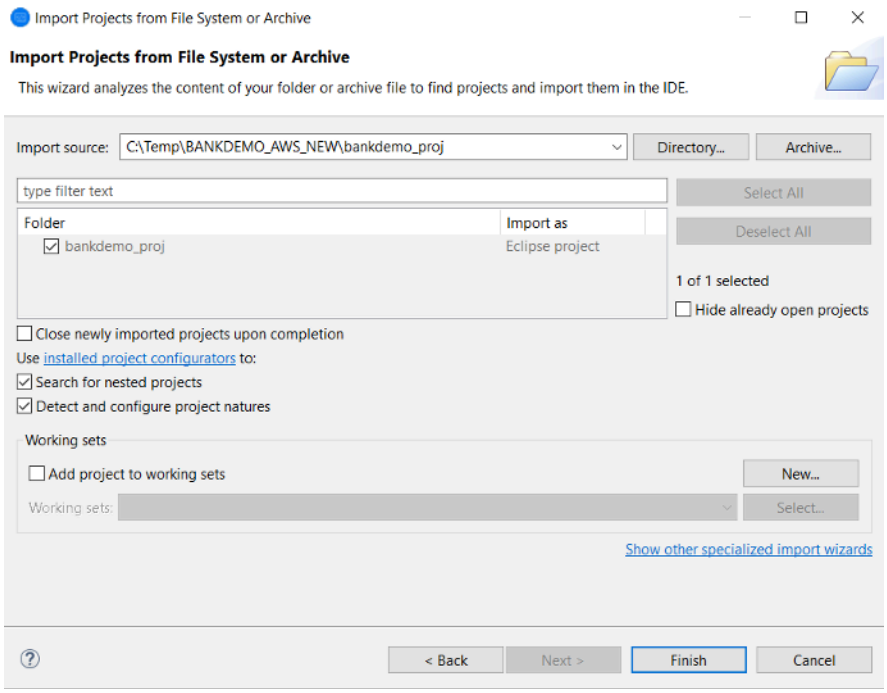
2. DariBerkasmenu, pilihImpor.



3. Dari dialog yang dihasilkan, di bawahUmum, pilihProyek dari Folder atau Arsipdan pilihlahSelanjutnya.



4. MengisiSumber impor, PilihDirektoridan telusuri sistem file untuk memilih folder proyek yang telah ditentukan sebelumnya. Proyek yang terkandung dalam memiliki link ke folder sumber dalam repositori yang sama.

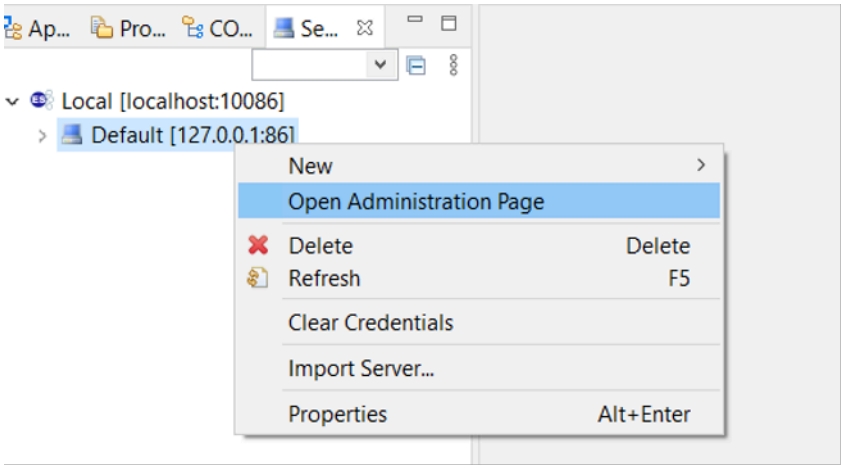


Pilih Selesai.

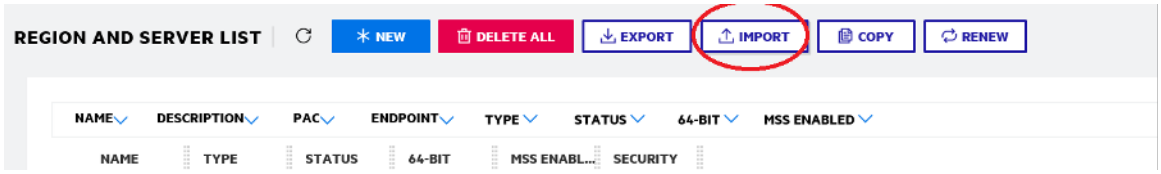
Karena proyek ini diisi oleh link ke folder sumber, kode secara otomatis dibangun.

Menggunakan Templat JSON Definisi Wilayah

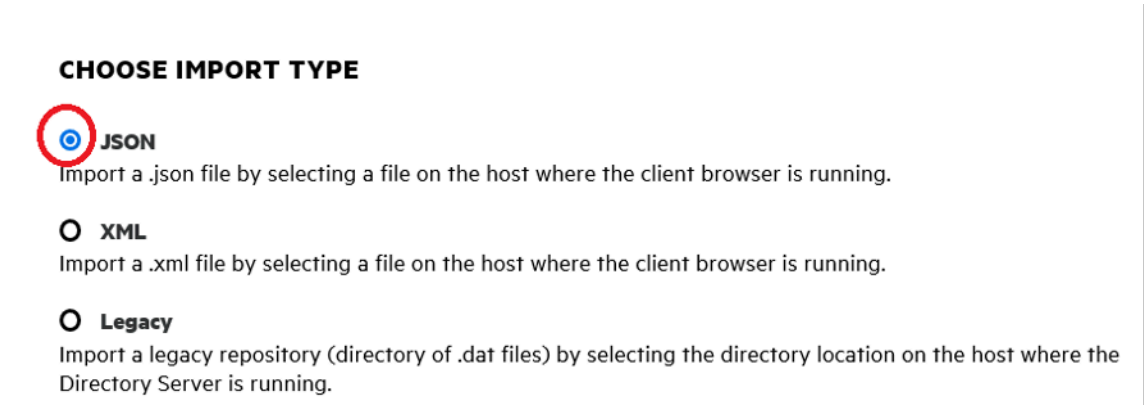
1. Beralih ke tampilan Server Explorer. Dari menu konteks terkait, pilihBuka Halaman Administrasi, yang memulai browser default.



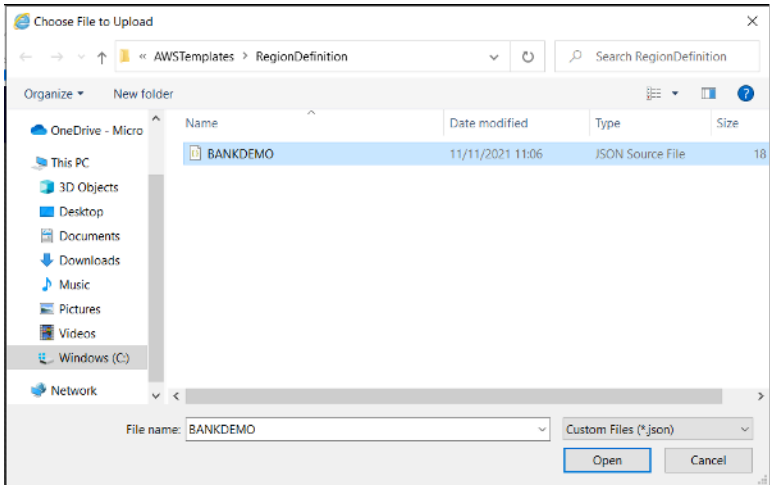
2. Dari layar Enterprise Server Common Web Administration (ESCWA) yang dihasilkan, pilihImpor.



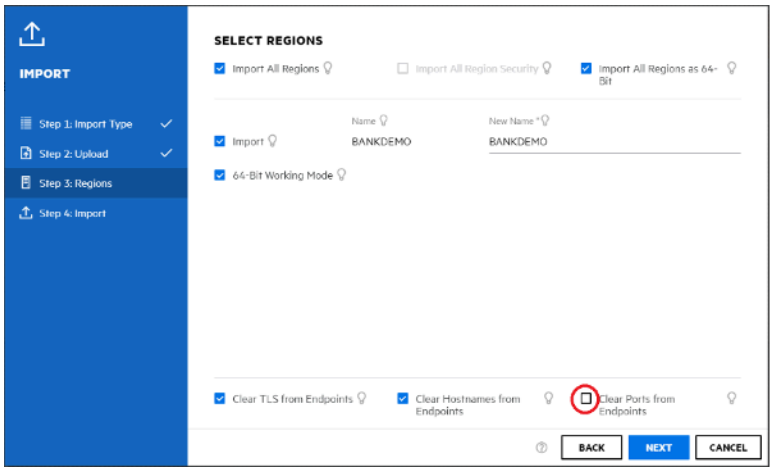
3. PilihJSONjenis impor dan pilihSelanjutnya.



4. Unggah yang disediakan`BANKDEMO.JSON`berkas.

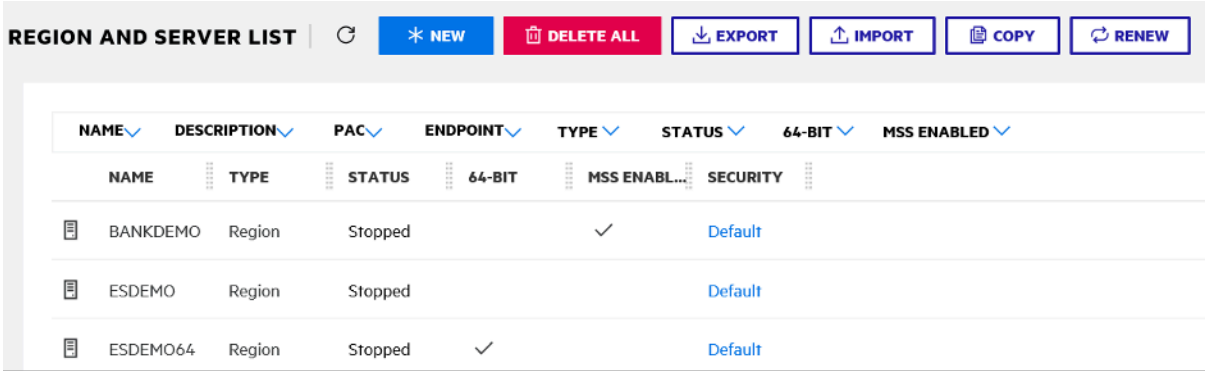


Setelah dipilih, pilihSelanjutnya.

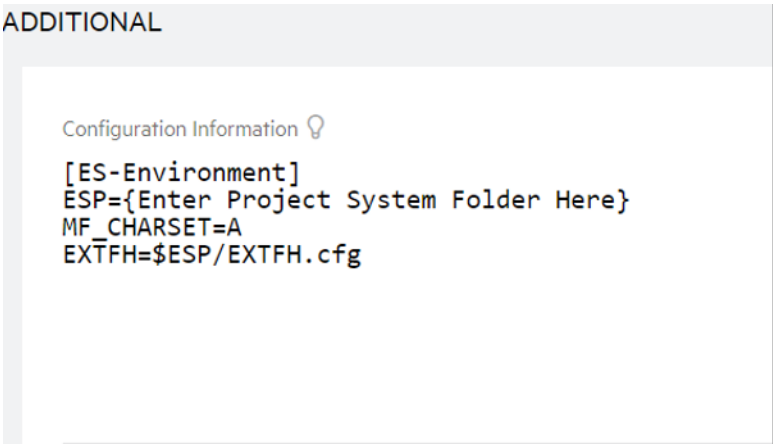


PadaMemiiilih Wilayahpanel, memastikan bahwaHapus Port dari Endpointpilihan tidak dipilih, dan kemudian terus memilihSelanjutnyamelalui panel sampaiMelakukan Imporpanel ditampilkan. Kemudian pilih Impor.

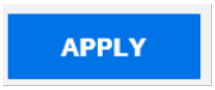
Akhirnya KlikSelesai. Wilayah BANKDEMO kemudian akan ditambahkan ke daftar server.



5. Arahkan keProperti Umumuntuk wilayah BANKDEMO.
6. Gulir keKonfigurasiBagian.
7. Variabel lingkungan ESP perlu diatur ke`System`folder yang relevan dengan Proyek Eclipse dibuat pada langkah sebelumnya. Ini seharusnya`workspacefolder/projectname/System`.

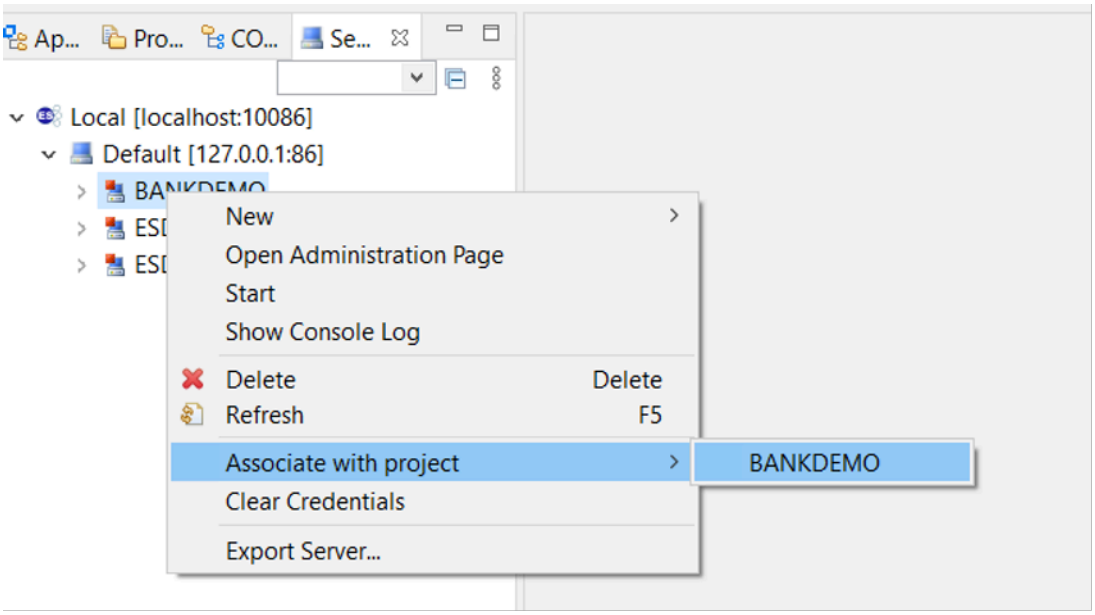


8. Klikapply.



Wilayah ini sekarang sepenuhnya dikonfigurasi untuk berjalan dalam hubungannya dengan proyek Eclipse COBOL.

9. Akhirnya, kembali Enterprise Developer, kaitkan wilayah impor dengan proyek.



Lingkungan Enterprise Developer sekarang siap digunakan, dengan versi kerja lengkap BankDemo. Anda dapat mengedit, mengkompilasi, dan men-debug kode terhadap wilayah tersebut.

> **Important**
>
> Jika Anda menggunakan versi Enterprise Developer untuk Windows, biner yang dihasilkan oleh compiler dapat berjalan hanya pada Enterprise Server disediakan dengan Enterprise Developer. Anda tidak dapat menjalankannya di bawah runtime Modernisasi Mainframe, yang didasarkan pada Linux.

Tutorial: Menyiapkan build Micro Focus untuk aplikasi BankDemo sampel

AWSModernisasi Mainframe memberi Anda kemampuan untuk menyiapkan build dan pipeline integrasi/pengiriman berkelanjutan (CI/CD) untuk aplikasi yang dimigrasi. Build dan pipeline ini menggunakanAWS CodeBuild,AWS CodeCommit, dan AWS CodePipeline untuk menyediakan kemampuan ini. CodeBuild adalah layanan build terkelola penuh yang mengkompilasi kode sumber Anda, menjalankan pengujian unit, dan menghasilkan artefak yang siap digunakan. CodeCommit adalah layanan kontrol versi yang memungkinkan Anda menyimpan dan mengelola reposit ori Git secara pribadi di Cloud. AWS CodePipeline adalah layanan pengiriman berkelanjutan yang

memungkinkan Anda untuk memodelkan, memvisualisasikan, dan mengotomatiskan langkah-langkah yang diperlukan untuk merilis perangkat lunak Anda.

Tutorial ini menunjukkan cara menggunakan AWS CodeBuild untuk mengkompilasi BankDemo contoh kode sumber aplikasi dari Amazon S3 dan kemudian mengekspor kode yang dikompilasi kembali ke Amazon S3.

AWS CodeBuildadalah layanan integrasi berkelanjutan yang dikelola sepenuhnya yang mengkompilasi kode sumber, menjalankan pengujian, dan menghasilkan paket perangkat lunak yang siap digunakan. Dengan CodeBuild, Anda dapat menggunakan lingkungan build yang dikemas sebelumnya, atau Anda dapat membuat lingkungan build khusus yang menggunakan alat build Anda sendiri. Skenario demo ini menggunakan opsi kedua. Ini terdiri dari lingkungan CodeBuild build yang menggunakan image Docker pra-paket.

> **Important**
>
> Sebelum Anda memulai proyek modernisasi mainframe Anda, kami sarankan Anda mempelajari tentang AWSMigration Acceleration Program (MAP) untuk Mainframe atau hubungi spesialis mainframe untuk mempelajari tentang langkah-langkah yang diperlukan untuk memodernisasi aplikasi AWS mainframe.

Topik

Prasyarat

Sebelum Anda memulai tutorial ini, selesaikan prasyarat berikut.

- Unduh aplikasi BankDemo sampel dan unzip ke folder. Folder sumber berisi program COBOL dan Copybooks, dan definisi CICS BMS. Ini juga berisi folder JCL untuk referensi, meskipun Anda tidak perlu membangun JCL. Folder ini juga berisi file meta yang diperlukan untuk build.
- Di konsol Modernisasi AWS Mainframe, pilih Tools. Dalam Analisis, pengembangan, dan pembuatan aset, pilih Bagikan aset dengan akun AWS saya.

Langkah 1: Buat ember Amazon S3

Pada langkah ini, Anda membuat dua ember Amazon S3. Yang pertama adalah bucket input untuk menahan kode sumber, dan yang lainnya adalah bucket keluaran untuk menampung output build. Untuk informasi selengkapnya, lihat Membuat, mengonfigurasi, dan bekerja dengan bucket Amazon S3 di Panduan Pengguna Amazon S3.

1. Untuk membuat bucket input, masuk ke konsol Amazon S3 dan pilih Buat bucket.
2. Dalam konfigurasi Umum, berikan nama untuk bucket dan tentukan Wilayah AWS tempat Anda ingin membuat bucket. Contoh nama adalah`codebuild-regionId-accountId-input-bucket`, di mana `regionId` ember, dan `accountId` Akun AWS ID Anda. Wilayah AWS

   > **Note**
   >
   > Jika Anda membuat bucket berbeda Wilayah AWS dari US East (Virginia N.), tentukan `LocationConstraint` parameternya. Untuk informasi selengkapnya, lihat Membuat Bucket di Referensi API Amazon Simple Storage Service.

3. Pertahankan semua pengaturan lainnya dan pilih Buat ember.
4. Ulangi langkah 1-3 untuk membuat bucket keluaran. Contoh nama adalah`codebuild-regionId-accountId-output-bucket`, di mana `regionId` ember dan `accountId` Akun AWS ID Anda. Wilayah AWS

   Apa pun nama yang Anda pilih untuk ember ini, pastikan untuk menggunakannya di seluruh tutorial ini.

Langkah 2: Buat file spesifikasi build

Pada langkah ini, Anda membuat file spesifikasi build,. File ini menyediakan perintah build dan pengaturan terkait, dalam format YAMM, CodeBuild untuk menjalankan build. Untuk informasi selengkapnya, lihat Membangun referensi spesifikasi untuk CodeBuild di Panduan AWS CodeBuild Pengguna.

1. Buat file bernama `buildspec.yml` di direktori yang Anda buka ritsleting sebagai prasyarat.
2. Tambahkan konten berikut ke file dan simpan. Tidak ada perubahan yang diperlukan untuk file ini.

```
version: 0.2
env:
  exported-variables:
    - CODEBUILD_BUILD_ID
    - CODEBUILD_BUILD_ARN
phases:
  install:
    runtime-versions:
      python: 3.7
  pre_build:
    commands:
      - echo Installing source dependencies...
      - ls -lR $CODEBUILD_SRC_DIR/source
  build:
    commands:
      - echo Build started on `date`
      - /start-build.sh -Dbasedir=$CODEBUILD_SRC_DIR/source -Dloaddir=
$CODEBUILD_SRC_DIR/target
  post_build:
    commands:
      - ls -lR $CODEBUILD_SRC_DIR/target
      - echo Build completed on `date`
artifacts:
  files:
    - $CODEBUILD_SRC_DIR/target/**
```

Di sini`CODEBUILD_BUILD_ID`,`CODEBUILD_BUILD_ARN`,`$CODEBUILD_SRC_DIR/source`, dan `$CODEBUILD_SRC_DIR/target` merupakan variabel lingkungan yang tersedia di dalamnya CodeBuild. Untuk informasi selengkapnya, lihat Variabel lingkungan di lingkungan build.

Pada titik ini, direktori Anda akan terlihat seperti ini.

```
(root directory name)
    |-- build.xml
    |-- buildspec.yml
    |-- LICENSE.txt
    |-- source
         |... etc.
```

3. Zip isi folder ke file bernama`BankDemo.zip`.. Untuk tutorial ini, Anda tidak dapat zip folder. Sebagai gantinya, zip isi folder ke file`BankDemo.zip`.

Langkah 3: Unggah file sumber

Pada langkah ini, Anda mengunggah kode sumber untuk aplikasi BankDemo sampel ke bucket input Amazon S3 Anda.

1. Masuk ke konsol Amazon S3 dan pilih Bucket di panel navigasi kiri. Kemudian pilih bucket input yang Anda buat sebelumnya.
2. Di bawah Objek, pilih Unggah.
3. Di bagian File dan folder, pilih Tambahkan File.
4. Arahkan ke dan pilih `BankDemo.zip` file Anda.
5. Pilih Upload (Unggah).

Langkah 4: Buat kebijakan IAM

Pada langkah ini, Anda membuat dua kebijakan IAM. Satu kebijakan memberikan izin untuk Modernisasi AWS Mainframe untuk mengakses dan menggunakan image Docker yang berisi alat build Micro Focus. Kebijakan ini tidak disesuaikan untuk pelanggan. Kebijakan lainnya memberikan izin untuk Modernisasi AWS Mainframe untuk berinteraksi dengan bucket input dan output, dan dengan log Amazon yang menghasilkan. CloudWatch CodeBuild

Untuk mempelajari cara membuat kebijakan IAM, lihat Mengedit kebijakan IAM di Panduan Pengguna IAM.

Untuk membuat kebijakan untuk mengakses gambar Docker

1. Di konsol IAM, salin dokumen kebijakan berikut dan tempelkan ke editor kebijakan.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:GetAuthorizationToken"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage"
            ],
            "Resource": "arn:aws:ecr:*:673918848628:repository/m2-enterprise-build-
tools"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::aws-m2-repo-*/*"
        }
    ]
}
```

2. Berikan nama untuk kebijakan tersebut, misalnya,`m2CodeBuildPolicy`.

Untuk membuat kebijakan yang memungkinkan Modernisasi AWS Mainframe berinteraksi dengan bucket dan log

1. Di konsol IAM, salin dokumen kebijakan berikut dan tempelkan ke editor kebijakan. Pastikan untuk memperbarui `regionId` keWilayah AWS, dan `accountId` ke AndaAkun AWS.

```
{
      "Version": "2012-10-17",
      "Statement": [
          {
              "Action": [
                  "logs:CreateLogGroup",
                  "logs:CreateLogStream",
                  "logs:PutLogEvents"
              ],
              "Resource": [
                  "arn:aws:logs:regionId:accountId:log-group:/aws/codebuild/
codebuild-bankdemo-project",
                  "arn:aws:logs:regionId:accountId:log-group:/aws/codebuild/
codebuild-bankdemo-project:*"
              ],
              "Effect": "Allow"
          },
          {
              "Action": [
                  "s3:PutObject",
                  "s3:GetObject",
                  "s3:GetObjectVersion",
                  "s3:GetBucketAcl",
                  "s3:GetBucketLocation",
                  "s3:List*"
              ],
              "Resource": [
                  "arn:aws:s3:::codebuild-regionId-accountId-input-bucket",
                  "arn:aws:s3:::codebuild-regionId-accountId-input-bucket/*",
                  "arn:aws:s3:::codebuild-regionId-accountId-output-bucket",
                  "arn:aws:s3:::codebuild-regionId-accountId-output-bucket/*"
              ],
              "Effect": "Allow"
          }
      ]
  }
```

2. Berikan nama untuk kebijakan tersebut, misalnya,`BankdemoCodeBuildRolePolicy`.

Langkah 5: Buat peran IAM

Pada langkah ini, Anda membuat peran IAM baru yang memungkinkan CodeBuild untuk berinteraksi dengan AWS sumber daya untuk Anda, setelah Anda mengaitkan kebijakan IAM yang sebelumnya Anda buat dengan peran IAM baru ini.

Untuk informasi tentang membuat peran layanan, lihat Membuat Peran untuk Mendelegasikan Izin ke AWS Layanan di Panduan Pengguna IAM,.

1. Masuk ke konsol IAM dan pilih Peran di panel navigasi kiri.
2. Pilih Buat peran.
3. Di bawah Jenis entitas Tepercaya, pilih layanan AWS.
4. Di bawah Kasus penggunaan untuk layanan AWS lainnya CodeBuild, pilih, lalu pilih CodeBuildlagi.
5. Pilih Berikutnya.
6. Pada halaman Tambahkan izin, pilih Berikutnya. Anda menetapkan kebijakan untuk peran nanti.
7. Di bawah Rincian peran, berikan nama untuk peran tersebut, misalnya,`BankdemoCodeBuildServiceRole`.
8. Di bawah Pilih entitas tepercaya, verifikasi bahwa dokumen kebijakan terlihat seperti berikut:

```
{
          "Version": "2012-10-17",
          "Statement": [
            {
              "Effect": "Allow",
              "Principal": {
                "Service": "codebuild.amazonaws.com"
              },
              "Action": "sts:AssumeRole"
            }
          ]
        }
```

9. Pilih Buat peran.

Langkah 6: Lampirkan kebijakan IAM ke peran IAM

Pada langkah ini, Anda melampirkan dua kebijakan IAM yang sebelumnya Anda buat ke peran `BankdemoCodeBuildServiceRole` IAM.

1. Masuk ke konsol IAM dan pilih Peran di panel navigasi kiri.
2. Di Peran, pilih peran yang Anda buat sebelumnya, misalnya,`BankdemoCodeBuildServiceRole`.
3. Di kebijakan Izin, pilih Tambahkan izin, lalu Lampirkan kebijakan.
4. Dalam kebijakan izin lainnya, pilih kebijakan yang Anda buat sebelumnya, misalnya, `m2CodeBuildPolicy` dan`BankdemoCodeBuildRolePolicy`.
5. Pilih Lampirkan kebijakan.

Langkah 7: Buat CodeBuild proyek

Pada langkah ini, Anda membuat CodeBuild proyek.

1. Masuk ke CodeBuild konsol dan pilih Buat proyek build.
2. Di bagian konfigurasi Proyek, berikan nama untuk proyek, misalnya,`codebuild-bankdemo-project`.
3. Di bagian Sumber, untuk penyedia Sumber, pilih Amazon S3, lalu pilih bucket input yang Anda buat sebelumnya, misalnya,. `codebuild-regionId-accountId-input-bucket`
4. Di bidang kunci objek S3 atau folder S3, masukkan nama file zip yang Anda unggah ke bucket S3. Dalam hal ini, nama file adalah`bankdemo.zip`.
5. Di bagian Lingkungan, pilih Gambar kustom.
6. Di bidang Environment type, pilih Linux.
7. Di bawah Registri gambar, pilih Registri lain.
8. Di bidang URL registri eksternal, masukkan `673918848628.dkr.ecr.us-west-2.amazonaws.com/m2-enterprise-build-tools:latest`
9. Di bawah Peran layanan, pilih Peran layanan yang ada, dan di bidang ARN Peran, pilih peran layanan yang Anda buat sebelumnya; misalnya,. `BankdemoCodeBuildServiceRole`
10. Di bagian Buildspec, pilih Gunakan file buildspec.
11. Di bagian Artefak, di bawah Jenis, pilih Amazon S3, lalu pilih bucket keluaran Anda, misalnya,. `codebuild-regionId-accountId-output-bucket`

12. Di bidang Nama, masukkan nama folder di bucket yang ingin berisi artefak keluaran build, misalnya,`bankdemo-output.zip`.
13. Di bawah kemasan Artefak, pilih Zip.
14. Pilih Buat proyek build.

Langkah 8: Mulai membangun

Pada langkah ini, Anda memulai build.

1. Masuk ke CodeBuild konsol.
2. Di panel navigasi kiri, pilih Membangun proyek.
3. Pilih proyek build yang Anda buat sebelumnya, misalnya,`codebuild-bankdemo-project`.
4. Pilih Mulai membangun.

Perintah ini memulai build. Build berjalan secara asinkron. Output dari perintah adalah JSON yang menyertakan id atribut. Atribut ini adalah referensi ke id CodeBuild build build yang baru saja Anda mulai. Anda dapat melihat status build di CodeBuild konsol. Anda juga dapat melihat log terperinci tentang eksekusi build di konsol. Untuk informasi selengkapnya, lihat Melihat informasi build terperinci di Panduan AWS CodeBuild Pengguna.

Ketika fase saat ini SELESAI, itu berarti build Anda berhasil diselesaikan, dan artefak yang dikompilasi sudah siap di Amazon S3.

Langkah 9: Unduh artefak keluaran

Pada langkah ini, Anda mengunduh artefak keluaran dari Amazon S3. Alat build Micro Focus dapat membuat beberapa tipe executable yang berbeda. Dalam tutorial ini, menghasilkan objek bersama.

1. Masuk ke konsol Amazon S3.
2. Di bagian Bucket, pilih nama bucket keluaran Anda, misalnya,`codebuild-regionId-accountId-output-bucket`.
3. Pilih Unduh.
4. Unzip file yang diunduh. Arahkan ke folder target untuk melihat artefak build. Ini termasuk objek bersama `.so` Linux.

Pembersihan sumber daya

Jika Anda tidak lagi membutuhkan sumber daya yang Anda buat untuk tutorial ini, hapus untuk menghindari biaya tambahan. Untuk melakukannya, selesaikan langkah-langkah berikut:

- Hapus bucket S3 yang Anda buat untuk tutorial ini. Untuk informasi selengkapnya, lihat Menghapus bucket di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.
- Hapus kebijakan IAM yang Anda buat untuk tutorial ini. Untuk informasi selengkapnya, lihat Menghapus kebijakan IAM di Panduan Pengguna IAM.
- Hapus peran IAM yang Anda buat untuk tutorial ini. Untuk informasi lebih lanjut, lihat Menghapus peran atau profil instans dalam Panduan Pengguna IAM.
- Hapus CodeBuild proyek yang Anda buat untuk tutorial ini. Untuk informasi selengkapnya, lihat Menghapus proyek build CodeBuild di Panduan AWS CodeBuild Pengguna.

Tutorial: Menyiapkan pipeline CI/CD untuk digunakan dengan Pengembang Micro Focus Enterprise

Tutorial ini menunjukkan cara mengimpor, mengedit, mengkompilasi, dan menjalankan aplikasi BankDemo sampel di Micro Focus Enterprise Developer, dan kemudian melakukan perubahan Anda untuk memicu pipeline CI/CD.

Daftar Isi

Prasyarat

Unduh file-file berikut.

- `basic-infra.yaml`
  - Unduh dari Wilayah Eropa (Frankfurt).
  - Unduh dari Wilayah AS Timur (Virginia N.).
- `pipeline.yaml`
  - Unduh dari Wilayah Eropa (Frankfurt).
  - Unduh dari Wilayah AS Timur (Virginia N.).
- `m2-code-sync-function.zip`
  - Unduh dari Wilayah Eropa (Frankfurt).
  - Unduh dari Wilayah AS Timur (Virginia N.).
- `config_git.yml`
  - Unduh dari Wilayah Eropa (Frankfurt).
  - Unduh dari Wilayah AS Timur (Virginia N.).
- `BANKDEMO-source.zip`

  - Unduh dari Wilayah Eropa (Frankfurt).
  - Unduh dari Wilayah AS Timur (Virginia N.).
- `BANKDEMO-exercise.zip`
  - Unduh dari Wilayah Eropa (Frankfurt).
  - Unduh dari Wilayah AS Timur (Virginia N.).

Tujuan dari setiap file adalah sebagai berikut:

`basic-infra.yaml`

AWS CloudFormationTemplate ini menciptakan infrastruktur dasar yang diperlukan untuk pipeline CI/CD: VPC, bucket Amazon S3, dan sebagainya.

`pipeline.yaml`

AWS CloudFormationTemplate ini digunakan oleh fungsi Lambda untuk meluncurkan tumpukan pipeline. Pastikan template ini terletak di bucket Amazon S3 yang dapat diakses publik. Tambahkan tautan ke bucket ini sebagai nilai default untuk `PipelineTemplateURL` parameter dalam `basic-infra.yaml` template.

`m2-code-sync-function.zip`

Fungsi Lambda ini membuat CodeCommit repositori, struktur direktori berdasarkan`config_git.yaml`, dan meluncurkan tumpukan pipeline menggunakan. `pipeline.yaml` Pastikan file zip ini tersedia di bucket Amazon S3 yang dapat diakses publik di semua Wilayah AWS AWS tempat Modernisasi Mainframe didukung. Kami menyarankan Anda menyimpan file dalam ember dalam satu ember Wilayah AWS dan mereplikasi ke ember di semua. Wilayah AWS Gunakan konvensi penamaan untuk bucket dengan akhiran yang mengidentifikasi spesifik Wilayah AWS (misalnya,`m2-cicd-deployment-source-eu-west-1`) dan tambahkan awalan `m2-cicd-deployment-source` sebagai nilai default untuk parameter `DeploymentSourceBucket` dan bentuk bucket penuh dengan menggunakan fungsi AWS CloudFormation substitusi `!Sub {DeploymentSourceBucket}-${AWS::Region}` sambil merujuk ke bucket tersebut di template untuk sumber daya. `basic-infra.yaml` `SourceSyncLambdaFunction`

`config_git.yml`

CodeCommit definisi struktur direktori. Untuk informasi selengkapnya, lihat Contoh File Pemicu YAMAL config_git.yl.

`BANKDEMO-source.zip`.

BankDemo kode sumber dan file konfigurasi yang dibuat dari CodeCommit repositori.

`BANKDEMO-exercise.zip`.

BankDemo sumber untuk latihan tutorial yang dibuat dari CodeCommit repositori.

Buat infrastruktur dasar pipa CI/CD

Gunakan AWS CloudFormation template `basic-infra.yaml` untuk membuat tumpukan infrastruktur dasar pipa CI/CD melalui konsol. AWS CloudFormation Tumpukan ini membuat bucket Amazon S3 tempat Anda mengunggah kode dan data aplikasi, serta AWS Lambda fungsi pendukung untuk membuat sumber daya lain yang diperlukan seperti AWS CodeCommit repositori dan pipeline. AWS CodePipeline

> **Note**
>
> Untuk meluncurkan tumpukan ini, Anda memerlukan izin untuk mengelola IAM, Amazon S3, Lambda, dan dan izin untuk digunakan. AWS CloudFormation AWS KMS

1. Masuk ke AWS Management Console dan buka konsol AWS CloudFormation di https://console.aws.amazon.com/cloudformation.
2. Membuat tumpukan baru dengan menggunakan salah satu opsi berikut:
   - Pilih Buat tumpukan. Ini adalah satu-satunya opsi jika Anda memiliki tumpukan yang sedang berjalan.
   - Pada halaman Stacks, pilih Create Stack. Opsi ini hanya terlihat jika Anda tidak memiliki tumpukan yang sedang berjalan.
3. Pada halaman Tentukan template:
   - Dalam Siapkan template, pilih Template sudah siap.
   - Di Tentukan templat, pilih URL Amazon S3 sebagai sumber templat dan masukkan salah satu URL berikut tergantung pada URL Anda. Wilayah AWS
     - `https://m2-us-east-1.s3.us-east-1.amazonaws.com/cicd/mf/basic-infra.yaml`

- `https://m2-eu-central-1.s3.eu-central-1.amazonaws.com/cicd/mf/basic-infra.yaml`

- Untuk menerima pengaturan Anda, pilih Berikutnya.

Halaman Buat tumpukan terbuka.

Specify stack details

Stack name

Stack name

m2-cicd-pipeline-foundation

Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

Parameters

Parameters are defined in your template and allow you to input custom values when you create or update a stack.

Networking Configuration

Do you want to use an existing VPC in your account?

If you select 'Yes', then you must provide the VPC ID and the Subnet IDs.

No (Create one)

Which VPC ID should be used?

If you selected 'Yes' for UseExistingVPC, this parameter is required. Otherwise, this value will be ignored.

Which private subnet ID should be used?

If you selected 'Yes' for UseExistingVPC, this parameter is required. Otherwise, this value will be ignored.

Which private subnet ID in a different AZ should be used for HA?

If you selected 'Yes' for UseExistingVPC, this parameter is required. Otherwise, this value will be ignored.

Enter the CIDR block that should be used for the new VPC

If you selected 'No (Create one)' for UseExistingVPC, this parameter is required. Otherwise, this value will be ignored.

11.11.0.0/16

CIDR bits for creating subnets. Choose 5 for /27, 6 for /26, 7 for /25, 8 for /24 range

If you selected 'No (Create one)' for UseExistingVPC, this parameter is required. Otherwise, this value will be ignored.

5

Deployment Configuration

Name of the S3 bucket which contains the source files for this stack deployment

Don't change unless you know what you are doing.

m2

Name of the source package file for the infrastructure Lambda function

Don't change unless you know what you are doing.

cicd/mf/m2-code-sync-function.zip

Full URL of the pipeline CloudFormation template file

Don't change unless you know what you are doing.

https://m2-us-east-1.s3.us-east-1.amazonaws.com/cicd/mf/pipeline.yaml

What name prefix to use for the new S3 buckets?

A name prefix for the S3 buckets that will be created by this stack.

mf

Cancel Previous Next

Lakukan perubahan berikut:

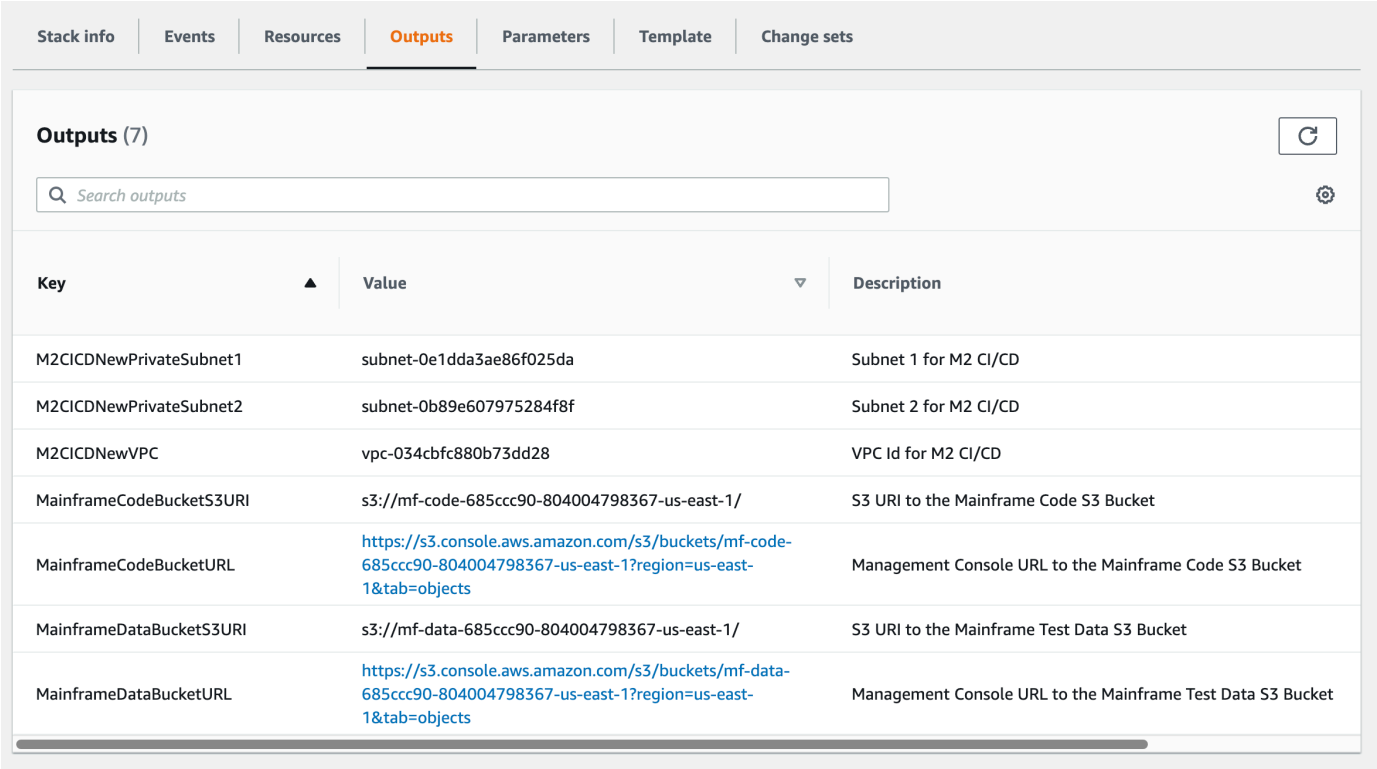
- Berikan nilai yang sesuai untuk nama Stack dan parameter untuk Konfigurasi Jaringan.
- Sebagian besar parameter dalam Konfigurasi Deployment telah diisi sebelumnya dengan tepat sehingga Anda tidak perlu memodifikasinya. Bergantung pada AndaWilayah AWS, ubah AWS CloudFormation template pipeline ke salah satu URL Amazon S3 berikut.
  - `https://m2-us-east-1.s3.amazonaws.com/cicd/mf/pipeline.yaml`
  - `https://m2-eu-central-1.s3.eu-central-1.amazonaws.com/cicd/mf/pipeline.yaml`
- Pilih Berikutnya.

> **Note**
>
> Jangan mengubah nilai parameter default kecuali Anda telah memodifikasi AWS CloudFormation template sendiri.

4. Di Configure stack options, pilih Next.
5. Dalam Kemampuan, pilih Saya mengakui yang AWS CloudFormation mungkin membuat sumber daya IAM untuk mengizinkan izin AWS CloudFormation untuk membuat Peran IAM atas nama Anda. Pilih Create stack.

> **Note**
>
> Diperlukan waktu 3 hingga 5 menit agar tumpukan ini disediakan.

6. Setelah tumpukan berhasil dibuat, navigasikan ke bagian Output dari tumpukan yang baru disediakan. Di sana Anda akan menemukan bucket Amazon S3 tempat Anda perlu mengunggah kode mainframe dan file dependen.



Buat AWS CodeCommit repositori dan pipa CI/CD

Pada langkah ini, Anda membuat CodeCommit repositori dan menyediakan tumpukan pipeline CI/CD dengan memanggil fungsi Lambda yang memanggil untuk membuat tumpukan pipeline. AWS CloudFormation

1. Unduh aplikasi BankDemo sampel ke mesin lokal Anda.
2. Unggah `bankdemo.zip` dari mesin lokal Anda ke bucket Amazon S3 yang dibuat di. Buat infrastruktur dasar pipa CI/CD
3. Unduh `config_git.yml`.
4. Ubah `config_git.yml` jika diperlukan, sebagai berikut:

   - Tambahkan nama repositori target Anda sendiri, cabang target, dan pesan komit.

   ```
   repository-config:
     target-repository: bankdemo-repo
     target-branch: main
     commit-message: Initial commit for bankdemo-repo main branch
   ```

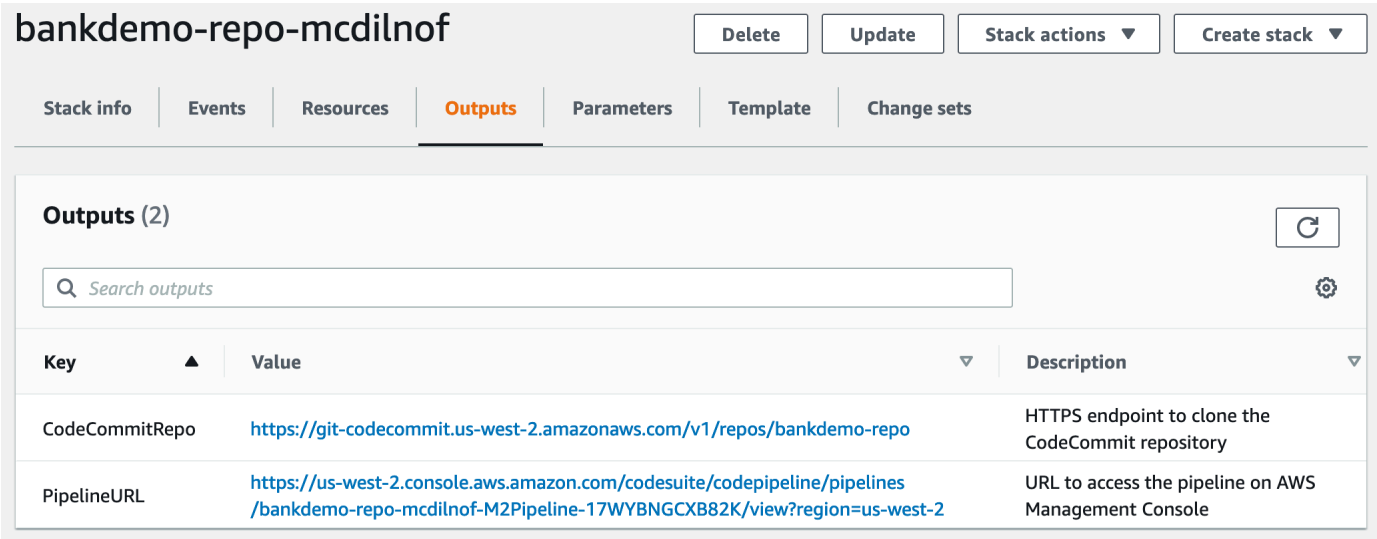
   - Tambahkan alamat email yang ingin Anda terima notifikasi.

```
pipeline-config:
  # Send pipeline failure notifications to these email addresses
  alert-notifications:
    - myname@mycompany.com
  # Send notifications for manual approval before production deployment to these
 email addresses
  approval-notifications:
    - myname@mycompany.com
```

5. Unggah `config_git.yml` file yang berisi definisi struktur folder CodeCommit repositori ke bucket Amazon S3 yang dibuat di. Buat infrastruktur dasar pipa CI/CD Ini akan memanggil fungsi Lambda yang secara otomatis akan menyediakan repositori dan pipeline.

   Ini akan membuat CodeCommit repositori dengan nama yang disediakan dalam `target-repository` didefinisikan dalam `config_git.yml` file; misalnya,. `bankdemo-repo`

   Fungsi Lambda juga akan membuat tumpukan pipa CI/CD. AWS CloudFormation AWS CloudFormationTumpukan akan memiliki awalan yang sama dengan `target-repository` nama yang diberikan diikuti oleh string acak (misalnya`bankdemo-repo-`*`01234567`*. Anda dapat menemukan URL CodeCommit repositori dan URL untuk mengakses pipeline yang dibuat di AWS Management Console.



6. Jika pembuatan CodeCommit repositori selesai, pipeline CI/CD akan segera dipicu untuk melakukan CI/CD penuh.
7. Setelah file didorong, file akan secara otomatis memicu pipeline yang akan dibangun, diterapkan dalam pementasan, menjalankan beberapa pengujian dan menunggu persetujuan manual sebelum menerapkannya di lingkungan produksi.

Contoh File Pemicu YAMAL config_git.yl

```
repository-config:
  target-repository: bankdemo-repo
  target-branch: main
  commit-message: Initial commit for bankdemo-repo main branch
  directory-structure:
    - '/':
        files:
        - build.xml
        - '*.yaml'
        - '*.yml'
        - '*.xml'
        - 'LICENSE.txt'
        readme: |
          # Root Folder
          - 'build.xml' : Build configuration for the application
    - tests:
        files:
        - '*.py'
        readme: |
          # Test Folder
          - '*.py' : Test scripts
    - config:
        files:
        - 'BANKDEMO.csd'
        - 'BANKDEMO.json'
        - 'BANKDEMO_ED.json'
        - 'dfhdrdat'
        - 'ESPGSQLXA.dll'
        - 'ESPGSQLXA64.so'
        - 'ESPGSQLXA64_S.so'
        - 'EXTFH.cfg'
        - 'm2-2021-04-28.normal.json'
        - 'MFDBFH.cfg'
        - 'application-definition-template-config.json'
        readme: |
          # Config Folder
          This folder contains the application configuration files.
          - 'BANKDEMO.csd'     : CICS Resource definitions export file
          - 'BANKDEMO.json'    : Enterprise Server configuration
          - 'BANKDEMO_ED.json' : Enterprise Server configuration for ED
          - 'dfhdrdat'         : CICS resource definition file
```

```
          - 'ESPGSQLXA.dll'    : XA switch module Windows
          - 'ESPGSQLXA64.so'   : XA switch module Linux
          - 'ESPGSQLXA64_S.so' : XA switch module Linux
          - 'EXTFH.cfg'        : Micro Focus File Handler configuration
          - 'm2-2021-04-28.normal.json' : M2 request document
          - 'MFDBFH.cfg'       : Micro Focus Database File Handler
          - 'application-definition-template-config.json' : Application definition for
 M2
    - source:
        subdirs:
        - .settings:
            files:
              - '.bms.mfdirset'
              - '.cbl.mfdirset'
        - copybook:
            files:
              - '*.cpy'
              - '*.inc'
            readme: |
              # Copy folder
              This folder contains the source for COBOL copy books, PLI includes, ...
              - .cpy COBOL copybooks
              - .inc PLI includes
#        - ctlcards:
#            files:
#              - '*.ctl'
#              - 'KBNKSRT1.txt'
#            readme: |
#              # Control Card folder
#              This folder contains the source for Batch Control Cards
#              - .ctl Control Cards
        - ims:
            files:
              - '*.dbd'
              - '*.psb'
            readme: |
              # ims folder
              This folder contains the IMS DB source files with the extensions
              - .dbd for IMS DBD source
              - .psb for IMS PSB source
        - jcl:
            files:
              - '*.jcl'
              - '*.ctl'
```

```
              - 'KBNKSRT1.txt'
              - '*.prc'
            readme: |
              # jcl folder
              This folder contains the JCL source files with the extensions
              - .jcl
#        - proclib:
#            files:
#              - '*.prc'
#            readme: |
#              # proclib folder
#              This folder contains the JCL procedures referenced via PROCLIB
 statements in the JCL with extensions
#              - .prc
        - rdbms:
            files:
              - '*.sql'
            readme: |
              # rdbms folder
              This folder contains any DB2 related source files with extensions
              - .sql for any kind of SQL source
        - screens:
            files:
            - '*.bms'
            - '*.mfs'
            readme: |
              # screens folder
              This folder contains the screens source files with the extensions
              - .bms for CICS BMS screens
              - .mfs for IMS MFS screens
            subdirs:
            - .settings:
                files:
                    - '*.bms.mfdirset'
        - cobol:
            files:
              - '*.cbl'
              - '*.pli'
            readme: |
              # source folder
              This folder contains the program source files with the extensions
              - .cbl for COBOL source
              - .pli for PLI source
            subdirs:
```

```
            - .settings:
                files:
                   - '*.cbl.mfdirset'
    - tests:
        files:
        - 'test_script.py'
        readme: |
          # tests Folder
          This folder contains the application test scripts
pipeline-config:
  alert-notifications:
    - myname@mycompany.com
  approval-notifications:
    - myname@mycompany.com
```

Pembuatan Pengembang Perusahaan AppStream 2.0

Untuk menyiapkan Pengembang Perusahaan Fokus Mikro di AppStream 2.0, lihatTutorial: Mengatur Pengembang Perusahaan Fokus Mikro di AppStream 2.0.

Untuk menghubungkan CodeCommit repositori ke Enterprise Developer, gunakan nama yang ditentukan dalam`target-repository`. Contoh File Pemicu YAMAL config_git.yl

Pengaturan dan Uji Pengembang Perusahaan

Topik

- Kloning BankDemo CodeCommit repositori di Enterprise Developer
- Buat proyek COBOL BankDemo mainframe dan bangun aplikasi
- Buat BankDemo CICS lokal dan lingkungan batch untuk pengujian
- Mulai server BANKDEMO dari Pengembang Perusahaan
- Mulai terminal Rumba 3270
- Jalankan BankDemo transaksi
- Hentikan server BANKDEMO dari Enterprise Developer

Connect ke instans Enterprise Developer AppStream 2.0 yang Anda buatPembuatan Pengembang Perusahaan AppStream 2.0.

1. Mulai Enterprise Developer dari Windows Start. Pilih Micro Focus Enterprise Developer, lalu pilih Enterprise Developer for Eclipse. Jika Anda memulai untuk pertama kalinya, mungkin perlu waktu.
2. Di Eclipse Launcher, di Workspace: enter **`C:\Users\<username>\workspace`** lalu pilih Launch.

   > **Note**
   >
   > Pastikan Anda memilih lokasi yang sama setelah menyambung kembali ke instance AppStream 2.0. Pemilihan ruang kerja tidak persisten.

3. Di Selamat Datang, pilih Buka Perspektif COBOL. Ini hanya akan ditampilkan pertama kali untuk ruang kerja baru.

Kloning BankDemo CodeCommit repositori di Enterprise Developer

1. Pilih Jendela/Perspektif/Perspektif Terbuka/Lainnya... /Git.
2. Pilih Kloning repositori Git.
3. Di Clone Git Repository, masukkan informasi berikut:

   - Di URI Lokasi, masukkan URL HTTPS dari CodeCommit repositori.

     > **Note**
     >
     > Salin URL Clone HTTPS untuk CodeCommit repositori di AWS Management Console dan tempel di sini. URI akan dibagi menjadi jalur Host dan Repository..

   - CodeCommit Kredensi repositori pengguna di Authentication User dan Password dan pilih Store in Secure Store.
4. Di Seleksi Cabang, pilih cabang Utama, lalu pilih Berikutnya.
5. Di Tujuan Lokal, di Direktori, masukkan `C:\Users\<username>\workspace` dan pilih Selesai.

   Proses klon selesai ketika `BANKDEMO [main]` ditampilkan dalam tampilan Git Repositories.

Buat proyek COBOL BankDemo mainframe dan bangun aplikasi

1. Ubah ke Perspektif COBOL.

2. Di Project, nonaktifkan Build Automatically.
3. Di File, pilih New, lalu Mainframe COBOL Project.
4. Di Proyek COBOL Mainframe Baru, masukkan informasi berikut:
   - Dalam nama Proyek, masukkan`BankDemo`.
   - Pilih template Micro Focus [64 bit].
   - Pilih Selesai.
5. Di COBOL Explorer, perluas BankDemo proyek baru.

   > **Note**
   >
   > `[BANKDEMO main]`dalam tanda kurung siku menunjukkan bahwa proyek terhubung dengan BankDemo CodeCommit repositori lokal.

6. Jika tampilan pohon tidak menampilkan entri untuk Program COBOL, Copybooks, Sumber BMS, dan File JCL, pilih Refresh dari menu konteks proyek. BankDemo
7. Dari menu BankDemo konteks, pilih Properties/Micro Focus/Pengaturan Proyek/COBOL:
   - Pilih Set Karakter - ASCII.
   - Pilih Terapkan, lalu Tutup.
8. Jika build sumber BMS dan COBOL tidak segera dimulai, periksa di menu Project, bahwa opsi Build Automatically diaktifkan.

   Output Build akan ditampilkan dalam tampilan Console dan akan selesai setelah beberapa menit dengan pesan `BUILD SUCCESSFUL` dan`Build finished with no errors`.

   BankDemo Aplikasi sekarang harus dikompilasi dan siap untuk eksekusi lokal.

Buat BankDemo CICS lokal dan lingkungan batch untuk pengujian

1. Di COBOL Explorer, perluas`BANKDEMO / config`.
2. Di editor, buka`BANKDEMO_ED.json`.
3. Temukan string `ED_Home=` dan ubah jalur untuk menunjuk ke proyek Enterprise Developer, sebagai berikut:`D:\\<username>\\workspace\\BANKDEMO`. Perhatikan penggunaan garis miring ganda (`\\`) dalam definisi jalur.
4. Simpan dan tutup file.

5. Pilih Server Explorer.
6. Dari menu konteks default, pilih Buka Halaman Administrasi. Halaman Administrasi Server Perusahaan Fokus Mikro dibuka di browser default.
7. Untuk sesi AppStream 2.0 saja, buat perubahan berikut sehingga Anda dapat mempertahankan wilayah Server Perusahaan lokal Anda untuk pengujian lokal:

   - Di Server Direktori/Default, pilih PROPERTIES/Configuration.
   - Ganti Lokasi Repositori dengan. `D:\<username>\My Files\Home Folder\MFDS`

   > **Note**
   >
   > Anda harus menyelesaikan langkah 5 - 8 setelah setiap koneksi baru ke instance AppStream 2.0.

8. Di Server Direktori/Default, pilih Impor, lalu selesaikan langkah-langkah berikut:

   - Pada Langkah 1: Jenis Impor, pilih JSON dan pilih Berikutnya.
   - Pada Langkah 2: Unggah, klik untuk mengunggah file dalam kotak biru.
   - Di Pilih File untuk Diunggah, masukkan:
     - Nama file:`D:\<username>\workspace\BANKDEMO\config\BANKDEMO_ED.json`.
     - Pilih Buka .
   - Pilih Selanjutnya.
   - Pada Langkah 3: Wilayah menghapus port yang jelas dari titik akhir.
   - Pilih Berikutnya.
   - Pada Langkah 4: Impor, pilih Impor.
   - Pilih Selesai.

   Daftar sekarang akan menampilkan nama server baru`BANKDEMO`.

Mulai server BANKDEMO dari Pengembang Perusahaan

1. Pilih Enterprise Developer.
2. Di Server Explorer, pilih Default, lalu pilih Refresh dari menu konteks.

Daftar server sekarang juga harus menampilkan BANKDEMO.

3. Pilih BANKDEMO.
4. Dari menu konteks, pilih Associate with project, lalu pilih BANKDEMO.
5. Dari menu konteks, pilih Mulai.

   Tampilan konsol harus menampilkan log untuk startup server.

   Jika pesan `BANKDEMO CASSI5030I PLTPI Phase 2 List(PI) Processing Completed` ditampilkan, Server siap untuk menguji aplikasi CICS BANKDEMO.

Mulai terminal Rumba 3270

1. Dari Windows Start, luncurkan Micro Focus Rumba+Desktop/Rumba+Desktop.
2. Di Selamat Datang, pilih BUAT SESI BARU/Tampilan Mainframe.
3. Di Tampilan Mainframe, pilih Koneksi/Konfigurasi.
4. Dalam Konfigurasi Sesi, pilih Koneksi/TN3270.
5. Di Nama Host/Alamat, pilih Sisipkan dan masukkan alamat IP`127.0.0.1`.
6. Di Telnet Port, masukkan port`6000`.
7. Pilih Apply (Terapkan).
8. Pilih Hubungkan.

   Layar selamat datang CICS menampilkan layar dengan pesan baris 1:`This is the Micro Focus MFE CICS region BANKDEMO`.
9. Tekan Ctrl+Shift+Z untuk menghapus layar.

Jalankan BankDemo transaksi

1. Di layar kosong, masukkan`BANK`.
2. Di layar BANK10, di bidang input untuk ID Pengguna... :, masuk `guest` dan tekan Enter.
3. Di layar BANK20, di bidang input sebelumnya Hitung biaya pinjaman, masukkan `/` (garis miring maju) dan tekan Enter.
4. Di layar BANK70:

   - Dalam jumlah yang ingin Anda pinjam... :, masukkan`10000`.

- Di Pada tingkat bunga... :, masukkan`5.0`.
- Dalam Untuk berapa bulan... :, masukkan`10`.
- Tekan Enter.

Hasil berikut harus ditampilkan:

```
Resulting monthly payment.............:   $1023.06
```

Ini melengkapi pengaturan aplikasi BANKDEMO di Enterprise Developer.

Hentikan server BANKDEMO dari Enterprise Developer

1. Di Server Explorer, pilih Default, lalu pilih Refresh dari menu konteks.
2. Pilih BANKDEMO.
3. Dari menu konteks, pilih Berhenti.

   Tampilan Konsol harus menampilkan log untuk server yang berhenti.

   Jika pesan `Server: BANKDEMO stopped successfully` ditampilkan, server telah berhasil dimatikan.

Latihan 1: Meningkatkan Perhitungan Pinjaman dalam Aplikasi BANKDEMO

Topik

Dalam skenario ini, Anda berjalan melalui proses membuat perubahan sampel pada kode, menerapkannya, dan mengujinya.

Departemen Pinjaman menginginkan bidang baru di layar Perhitungan Pinjaman BANK70 untuk menunjukkan Total Jumlah Pinjaman. Ini membutuhkan perubahan layar BMS MBANK70.CBL, menambahkan bidang baru dan program penanganan layar yang sesuai SBANK70P.CBL dengan copybook terkait. Selain itu, perhitungan rutin pinjaman di BBANK70P.CBL perlu diperpanjang dengan rumus tambahan.

Untuk menyelesaikan latihan ini, pastikan Anda menyelesaikan prasyarat berikut.

- Unduh BANKDEMO-exercise.zip ke`D:\PhotonUser\My Files\Home Folder`.
- Ekstrak file zip ke`D:\PhotonUser\My Files\Home Folder\BANKDEMO-exercise`.
- Buat folder`D:\PhotonUser\My Files\Home Folder\AnalysisRules`.
- Salin file aturan `Loan+Calculation+Update.General-1.xml` dari `BANKDEMO-exercise` folder ke`D:\PhotonUser\My Files\Home Folder\AnalysisRules`.

> **Note**
>
> Perubahan kode dalam*.CBL dan *.CPY ditandai dengan EXER01 di kolom 1 - 6 untuk latihan ini.

Tambahkan aturan analisis pinjaman ke Analisis Kode Pengembang Perusahaan

Aturan analisis yang didefinisikan dalam Micro Focus Enterprise Analyzer dapat diekspor dari Enterprise Analyzer dan diimpor ke Enterprise Developer untuk menjalankan aturan analisis yang sama di seluruh sumber dalam proyek Enterprise Developer.

1. Buka `Window/Preferences/Micro Focus/COBOL/Code Analysis/Rules`.
2. Pilih Edit... dan masukkan nama folder `D:\PhotonUser\My Files\Home Folder \AnalysisRules` yang berisi file aturan`Loan+Calculation+Update.General-1.xml`.
3. Pilih Selesai.
4. Pilih Terapkan, lalu pilih Tutup.
5. Dari menu konteks proyek BANKDEMO, pilih Analisis Kode.

   Anda akan melihat entri untuk Pembaruan Perhitungan Pinjaman.

Langkah 1: Lakukan analisis kode untuk perhitungan pinjaman

Dengan aturan analisis baru kami ingin mengidentifikasi program COBOL dan baris kode di sana yang cocok dengan pola pencarian`*PAYMENT*`, `*LOAN*` dan `*RATE*` dalam ekspresi, pernyataan dan variabel. Ini akan membantu menavigasi kode dan mengidentifikasi perubahan kode yang diperlukan.

1. Dari menu konteks proyek BANKDEMO, pilih Analisis Kode/Pembaruan Perhitungan Pinjaman.

   Ini akan menjalankan aturan pencarian dan daftar hasilnya di tab baru yang disebut Analisis Kode. Analisis berjalan selesai ketika bilah kemajuan hijau di kanan bawah menghilang.

   Tab Analisis Kode harus menampilkan daftar yang diperluas`BBANK20P.CBL`, `BBANK70P.CBL` dan`SBANK70P.CBL`, masing-masing mencantumkan pernyataan, ekspresi, dan variabel yang cocok dengan pola pencarian.

   Melihat hasilnya hanya `BBANK20P.CBL` ada literal yang dipindahkan yang memiliki kecocokan dengan pola pencarian. Jadi program ini bisa diabaikan.
2. Di bilah menu tab pilih - Ikon untuk menutup semua.
3. Perluas `SBANK70P.CBL` dan pilih baris apa pun dalam urutan apa pun dengan klik dua kali untuk melihat bagaimana ini akan membuka sumber dan sorot baris yang dipilih dalam kode sumber. Anda juga akan mengenali bahwa semua baris sumber yang diidentifikasi ditandai.

Langkah 2: Ubah peta CICS BMS dan program COBOL dan uji

Pertama kita akan mengubah peta BMS `MBANK70.BMS` dan program penanganan layar `SBANK70P.CBL` dan copybook `CBANKDAT.CPY` untuk menampilkan bidang baru. Untuk menghindari pengkodean yang tidak perlu dalam latihan ini, modul sumber yang dimodifikasi tersedia di `D:\PhotonUser\My Files\Home Folder\BANKDEMO-exercise\Exercise01` folder. Biasanya pengembang akan menggunakan hasil Analisis Kode untuk menavigasi dan memodifikasi sumber. Jika Anda memiliki waktu dan dan ingin melakukan perubahan manual melakukannya dengan informasi yang diberikan di*Perubahan manual di MBANK70.BMS dan SBANK70P.CBL (Opsional) *.

Untuk perubahan cepat, salin file berikut:

1. `..\BANKDEMO-exercise\Exercis01\screens\MBANK70.BMSkeD:\PhotonUser\workspace\bankdemo\source\screens`.

2. `.\BANKDEMO-exercise\Exercis01\cobol\SBANK70P.CBLkeD:\PhotonUser\workspace\bankdemo\source\cobol`.
3. `..\BANKDEMO-exercise\Exercis01\copybook\CBANKDAT.CPYkeD:\PhotonUser\workspace\bankdemo\source\copybook`.
4. Untuk memastikan bahwa semua program yang terkena dampak perubahan dikompilasi, pilih Project/Clean... /Bersihkan semua proyek.

Untuk perubahan manual ke `MBANK70.BMS` dan`SBANK70P.CBL`, selesaikan langkah-langkah berikut:

- Untuk perubahan manual dalam `MBANK70.BMS` sumber BMS tambahkan setelah `PAYMENT` bidang:
  - TXT09 dengan atribut yang sama dengan TXT08 dan nilai AWAL “Total Jumlah Pinjaman”
  - TOTAL dengan atribut yang sama dengan PEMBAYARAN

Uji perubahan

Untuk menguji perubahan, ulangi langkah-langkah di bagian berikut:

1. Mulai server BANKDEMO dari Pengembang Perusahaan
2. Mulai terminal Rumba 3270
3. Jalankan BankDemo transaksi

   Selain itu Anda sekarang juga harus melihat teksnya`Total Loan Amount.....................:`.
4. Hentikan server BANKDEMO dari Enterprise Developer

Langkah 3: Tambahkan perhitungan jumlah total dalam program COBOL

Pada langkah kedua kita akan mengubah `BBANK70P.CBL` dan menambahkan perhitungan untuk jumlah total pinjaman. Sumber yang disiapkan dengan perubahan yang diperlukan tersedia di `D:\PhotonUser\My Files\Home Folder\BANKDEMO-exercise\Exercise01` folder. Jika Anda memiliki waktu dan dan ingin melakukan perubahan manual, lakukan dengan informasi yang diberikan di*Perubahan manual di BBANK70P.CBL (Opsional) *.

Untuk perubahan cepat, salin file berikut:

- `..\BANKDEMO-exercise\Exercis01\source\cobol\BBANK70P.CBLkeD:\PhotonUser\workspace\bankdemo\source\cobol`.

Untuk membuat perubahan manualBBANK70P.CBL, selesaikan langkah-langkah berikut:

- Gunakan hasil Analisis Kode untuk mengidentifikasi perubahan yang diperlukan.

Uji perubahan

Untuk menguji perubahan, ulangi langkah-langkah di bagian berikut:

1. Mulai server BANKDEMO dari Pengembang Perusahaan
2. Mulai terminal Rumba 3270
3. Jalankan BankDemo transaksi

   Selain itu Anda sekarang juga harus melihat teksnya`Total Loan Amount.....................: $10230.60`.
4. Hentikan server BANKDEMO dari Enterprise Developer

Langkah 4: Komit perubahan dan jalankan pipa CI/CD

Komit perubahan ke CodeCommit repositori pusat dan picu pipeline CI/CD untuk membangun, menguji, dan menyebarkan perubahan.

1. Dari proyek BANKDEMO, dalam menu konteks, pilih Team/Commit.
2. Di tab Git Staging, masukkan pesan komit berikut:`Added Total Amount Calculation`.
3. Pilih Komit dan Dorong... .
4. Buka CodePipeline konsol dan periksa status eksekusi pipeline.

> **Note**
>
> Jika Anda menghadapi masalah dengan fungsi Enterprise Developer atau Teams Commit atau Push, gunakan antarmuka baris perintah Git Bash.

Latihan 2: Ekstrak perhitungan pinjaman dalam BankDemo aplikasi

Topik

Dalam latihan berikutnya ini, Anda mengerjakan permintaan perubahan sampel lainnya. Dalam skenario ini, departemen Pinjaman ingin menggunakan kembali perhitungan rutin pinjaman sebagai standalone WebService. Rutinitas harus tetap dalam COBOL dan juga harus tetap dapat dipanggil dari program CICS COBOL yang ada. `BBANK70P.CBL`

Langkah 1: Rutinitas perhitungan pinjaman refactor menjadi bagian COBOL

Pada langkah pertama kami mengekstrak rutin perhitungan pinjaman ke dalam Bagian COBOL. Langkah ini diperlukan untuk mengekstrak kode ke dalam program COBOL yang berdiri sendiri di langkah berikutnya.

1. Buka `BBANK70P.CBL` di Editor COBOL.
2. Di editor, pilih dari menu konteks Analisis Kode/Pembaruan Perhitungan Pinjaman. Ini hanya akan memindai sumber saat ini untuk pola yang ditentukan dalam aturan analisis.
3. Dalam hasil di tab Analisis Kode, temukan pernyataan aritmatika pertama. `DIVIDE WS-LOAN-INTEREST BY 12`
4. Klik dua kali pada pernyataan untuk menavigasi ke baris sumber di Editor. Ini adalah pernyataan pertama dari rutinitas perhitungan pinjaman.
5. Tandai blok kode berikut untuk rutinitas perhitungan pinjaman yang akan diekstraksi ke bagian.

```
DIVIDE WS-LOAN-INTEREST BY 12
             GIVING WS-LOAN-INTEREST ROUNDED.
           COMPUTE WS-LOAN-MONTHLY-PAYMENT ROUNDED =
             ((WS-LOAN-INTEREST * ((1 + WS-LOAN-INTEREST)
                 ** WS-LOAN-TERM)) /
             (((1 + WS-LOAN-INTEREST) * WS-LOAN-TERM) - 1 ))
                 * WS-LOAN-PRINCIPAL.
EXER01     COMPUTE WS-LOAN-TOTAL-PAYMENT =
EXER01               (WS-LOAN-MONTHLY-PAYMENT * WS-LOAN-TERM).
```

6. Dari menu konteks di editor, pilih Refactor/Extract to Section... .
7. Masukkan Nama bagian baru: PERHITUNGAN PINJAMAN.
8. Pilih OKE.

   Blok kode yang ditandai sekarang telah diekstraksi ke `LOAN-CALCULATION` bagian baru dan blok kode telah diganti dengan `PERFROM LOAN-CALCULATION` pernyataan.

Uji perubahan

Untuk menguji perubahan, ulangi langkah-langkah yang dijelaskan di bagian berikut.

1. Mulai server BANKDEMO dari Pengembang Perusahaan
2. Mulai terminal Rumba 3270
3. Jalankan BankDemo transaksi

   Selain itu Anda sekarang juga harus melihat teksnya`Total Loan Amount.....: $10230.60`.
4. Hentikan server BANKDEMO dari Enterprise Developer

> **Note**
>
> Jika Anda ingin menghindari langkah-langkah di atas untuk mengekstrak blok kode ke bagian, Anda dapat menyalin sumber yang dimodifikasi untuk Langkah 1 dari `..\BANKDEMO-exercise\Exercis02\Step1\cobol\BBANK70P.CBL` ke`D:\PhotonUser\workspace\bankdemo\source\cobol`.

Langkah 2: Ekstrak rutin perhitungan pinjaman ke program COBOL mandiri

Pada Langkah 2 blok kode di `LOAN-CALCULATION` bagian akan diekstraksi ke program mandiri dan kode asli akan diganti dengan kode untuk memanggil subprogram baru.

1. Buka `BBANK70P.CBL` di editor dan temukan `PERFORM LOAN-CALCULATION` pernyataan baru yang dibuat di Langkah 1.
2. Tempatkan kursor di dalam nama bagian. Ini akan ditandai abu-abu.
3. Dari menu konteks, pilih Refactor-> Ekstrak Bagian/Paragraf ke Program... .
4. Di Bagian Ekstrak/Paragraf ke Program, masukkan Nama file baru: LOANCALC.CBL.

5. Pilih OKE.

   `LOANCALC.CBL`Program baru akan terbuka di editor.

6. Gulir ke bawah dan tinjau kode yang diekstraksi dan dihasilkan untuk antarmuka panggilan.

7. Pilih editor dengan `BBANK70P.CBL` dan pergi ke`LOAN-CALCULATION SECTION`. Tinjau kode yang dihasilkan untuk memanggil sub-program `LOANCALC.CBL` baru.

> **Note**
>
> `CALL`Pernyataan ini menggunakan `DFHEIBLK` dan `DFHCOMMAREA` memanggil `LOANCALC` dengan blok kontrol CICS. Karena kita ingin memanggil `LOANCALC.CBL` sub-program baru sebagai program non-CICS, kita harus menghapus `DFHEIBLK` dan `DFHCOMMAREA` dari panggilan baik dengan mengomentari atau menghapus.

Uji perubahan

Untuk menguji perubahan, ulangi langkah-langkah yang dijelaskan di bagian berikut.

1. Mulai server BANKDEMO dari Pengembang Perusahaan
2. Mulai terminal Rumba 3270
3. Jalankan BankDemo transaksi

   Selain itu Anda sekarang juga harus melihat teksnya`Total Loan Amount.....................: $10230.60`.

4. Hentikan server BANKDEMO dari Enterprise Developer

> **Note**
>
> Jika Anda ingin menghindari langkah-langkah di atas untuk mengekstrak blok kode ke bagian, Anda dapat menyalin sumber yang dimodifikasi untuk Langkah 1 dari `..\BANKDEMO-exercise\Exercis02\Step2\cobol\BBANK70P.CBL` dan `LOANCALC.CBL keD:\PhotonUser\workspace\bankdemo\source\cobol`.

Langkah 3: Lakukan perubahan dan jalankan pipeline CI/CD

Komit perubahan ke CodeCommit reposiotry pusat dan picu Pipeline CI/CD untuk membangun, menguji, dan menerapkan perubahan.

1. Dari proyek BANKDEMO, dalam menu konteks, pilih Team/Commit.
2. Di tab Git Staging
   - Tambahkan Tahapan Unstaged LOANCALC.CBL dan Loancalc.cbl.mfdirset.
   - Masukkan pesan komit:`Added Total Amount Calculation`.
3. Pilih Komit dan Dorong... .
4. Buka CodePipeline konsol dan periksa status eksekusi pipeline.

> **Note**
>
> Jika Anda menghadapi masalah dengan fungsi Enterprise Developer atau Teams Commit atau Push, gunakan antarmuka baris perintah Git Bash.

Pembersihan sumber daya

Jika Anda tidak lagi membutuhkan sumber daya yang Anda buat untuk tutorial ini, hapus sehingga Anda tidak akan terus dikenakan biaya untuk itu. Selesaikan langkah-langkah berikut:

- Hapus CodePipeline pipa. Untuk informasi selengkapnya, lihat Menghapus pipeline CodePipeline di Panduan AWS CodePipeline Pengguna.
- Hapus CodeCommit repositori. Untuk informasi selengkapnya, lihat Menghapus CodeCommit repositori di AWS CodeCommitPanduan Pengguna.
- Hapus ember S3;. Untuk informasi selengkapnya, lihat Menghapus bucket di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.
- Hapus AWS CloudFormation tumpukan. Untuk informasi selengkapnya, lihat Menghapus tumpukan di AWS CloudFormation konsol di Panduan AWS CloudFormation Pengguna.

Utilitas Batch dalam AWS Modernisasi Mainframe

Aplikasi mainframe sering menggunakan program utilitas batch untuk melakukan fungsi tertentu seperti menyortir data, mentransfer file menggunakan FTP, memuat data ke dalam database seperti DB2, membongkar data dari database, dan sebagainya.

Saat Anda memigrasikan aplikasi Anda ke Modernisasi AWS Mainframe, Anda memerlukan utilitas pengganti yang setara secara fungsional yang dapat melakukan tugas yang sama seperti yang Anda gunakan pada mainframe. Beberapa utilitas ini mungkin sudah tersedia sebagai bagian dari mesin runtime Modernisasi AWS Mainframe, tetapi kami menyediakan utilitas pengganti berikut:

- M2SFTP - memungkinkan transfer file aman menggunakan protokol SFTP.
- M2WAIT - menunggu jumlah waktu tertentu sebelum melanjutkan dengan langkah berikutnya dalam pekerjaan batch.
- TXT2PDF - mengonversi file teks ke format PDF.
- M2DFUTIL - menyediakan fungsi backup, restore, delete, dan copy pada set data yang mirip dengan dukungan yang disediakan oleh utilitas ADRDSSU mainframe.
- M2RUNCMD - memungkinkan Anda menjalankan perintah Micro Focus, skrip, dan panggilan sistem langsung dari JCL.

Kami mengembangkan utilitas batch ini berdasarkan umpan balik pelanggan dan mendesainnya untuk menyediakan fungsionalitas yang sama dengan utilitas mainframe. Tujuannya adalah untuk membuat transisi Anda dari mainframe ke Modernisasi AWS Mainframe semulus mungkin.

Topik

Lokasi Biner

Utilitas ini sudah diinstal sebelumnya pada produk Micro Focus Enterprise Developer (ED) dan Micro Focus Enterprise Server (ES). Anda dapat menemukannya di lokasi berikut untuk semua varian ED dan ES:

- Linux: `/opt/aws/m2/microfocus/utilities/64bit`
- Windows (32 bit): `C:\AWS\M2\MicroFocus\Utilities\32bit`
- Windows (64 bit): `C:\AWS\M2\MicroFocus\Utilities\64bit`

Utilitas Batch M2SFTP

M2SFTP adalah program utilitas JCL yang dirancang untuk melakukan transfer file aman antar sistem menggunakan Secure File Transfer Protocol (SFTP). Program ini menggunakan klien Putty SFTP,`psftp`, untuk melakukan transfer file yang sebenarnya. Program ini bekerja mirip dengan program utilitas FTP mainframe dan menggunakan otentikasi pengguna dan kata sandi.

> **Note**
>
> Otentikasi kunci publik tidak didukung.

Untuk mengonversi JCL FTP mainframe Anda untuk menggunakan SFTP, ubah ke. `PGM=FTP` `PGM=M2SFTP`

Topik

Platform yang didukung

Anda dapat menggunakan M2SFTP di salah satu platform berikut:

- AWS Modernisasi Mainframe Mikro Fokus Dikelola
- Waktu Proses Fokus Mikro (di Amazon EC2)
- Semua varian produk Micro Focus Enterprise Developer (ED) dan Micro Focus Enterprise Server (ES).

Menginstal dependensi

Untuk menginstal klien Putty SFTP di Windows

- Unduh klien Putty SFTP dan instal.

Untuk menginstal klien Putty SFTP di Linux:

- Jalankan perintah berikut untuk menginstal klien Putty SFTP:

```
sudo yum -y install putty
```

Konfigurasikan M2SFTP untuk Modernisasi Mainframe Dikelola AWS

Jika aplikasi yang dimigrasi berjalan di Modernisasi AWS Mainframe Dikelola, Anda perlu mengonfigurasi M2SFTP sebagai berikut.

- Tetapkan variabel lingkungan Micro Focus Enterprise Server yang sesuai untuk MFFTP. Berikut adalah beberapa contoh:
  - `MFFTP_TEMP_DIR`
  - `MFFTP_SENDEOL`
  - `MFFTP_TIME`
  - `MFFTP_ABEND`

Anda dapat mengatur sesedikit atau sebanyak variabel ini yang Anda inginkan. Anda dapat mengaturnya di JCL Anda menggunakan `ENVAR DD` pernyataan. Untuk informasi selengkapnya tentang variabel-variabel ini, lihat Variabel Kontrol MFFTP dalam dokumentasi Fokus Mikro.

Untuk menguji konfigurasi Anda, lihatContoh JCL.

Konfigurasikan M2SFTP untuk runtime Modernisasi AWS Mainframe di Amazon EC2 (termasuk 2.0) AppStream

Jika aplikasi yang dimigrasi berjalan pada runtime Modernisasi AWS Mainframe di Amazon EC2, konfigurasikan M2SFTP sebagai berikut.

1. Ubah Jalur Program Micro Focus JES untuk menyertakan lokasi biner untuk utilitas batch. Jika Anda perlu menentukan beberapa jalur, gunakan titik dua (`:`) untuk memisahkan jalur di Linux dan titik koma (`;`) di Windows.

   - Linux: `/opt/aws/m2/microfocus/utilities/64bit`
   - Windows (32bit): `C:\AWS\M2\MicroFocus\Utilities\32bit`
   - Windows (64bit): `C:\AWS\M2\MicroFocus\Utilities\64bit`

2. Tetapkan variabel lingkungan Micro Focus Enterprise Server yang sesuai untuk MFFTP. Berikut adalah beberapa contoh:

   - `MFFTP_TEMP_DIR`
   - `MFFTP_SENDEOL`
   - `MFFTP_TIME`
   - MFFTP_ABEND

   Anda dapat mengatur sesedikit atau sebanyak variabel ini yang Anda inginkan. Anda dapat mengaturnya di JCL Anda menggunakan `ENVAR DD` pernyataan. Untuk informasi selengkapnya tentang variabel-variabel ini, lihat Variabel Kontrol MFFTP dalam dokumentasi Fokus Mikro.

Untuk menguji konfigurasi Anda, lihatContoh JCL.

Contoh JCL

Untuk menguji instalasi, Anda dapat menggunakan salah satu dari contoh file JCL berikut.

M2SFTP1.jcl

JCL ini menunjukkan cara memanggil M2SFTP untuk mengirim file ke server SFTP jarak jauh. Perhatikan variabel lingkungan yang diatur dalam `ENVVAR DD` pernyataan.

```
//M2SFTP1 JOB 'M2SFTP1',CLASS=A,MSGCLASS=X,TIME=1440
//*
//* Copyright Amazon.com, Inc. or its affiliates.*
//* All Rights Reserved.*
//*
//*--------------------------------------------------------------------**
//* Sample SFTP JCL step to send a file to SFTP server*
//*--------------------------------------------------------------------**
//*
//STEP01 EXEC PGM=M2SFTP,
//            PARM='127.0.0.1 (EXIT=99 TIMEOUT 300'
//*
//SYSFTPD  DD  *
RECFM FB
LRECL 80
SBSENDEOL CRLF
MBSENDEOL CRLF
TRAILINGBLANKS FALSE
/*
//NETRC    DD  *
machine 127.0.0.1 login sftpuser password sftppass
/*
//SYSPRINT DD  SYSOUT=*
//OUTPUT   DD  SYSOUT=*
//STDOUT   DD  SYSOUT=*
//INPUT    DD  *
type a
locsite notrailingblanks
cd files
put 'AWS.M2.TXT2PDF1.PDF' AWS.M2.TXT2PDF1.pdf
put 'AWS.M2.CARDDEMO.CARDDATA.PS' AWS.M2.CARDDEMO.CARDDATA.PS1.txt
quit
/*
//ENVVAR   DD *
```

```
MFFTP_VERBOSE_OUTPUT=ON
MFFTP_KEEP=N
/*
//*
//
```

M2SFTP2.jcl

JCL ini menunjukkan cara memanggil M2SFTP untuk menerima file dari server SFTP jarak jauh. Perhatikan variabel lingkungan yang ditetapkan dalam `ENVVAR DD` pernyataan.

```
//M2SFTP2 JOB 'M2SFTP2',CLASS=A,MSGCLASS=X,TIME=1440
//*
//* Copyright Amazon.com, Inc. or its affiliates.*
//* All Rights Reserved.*
//*
//*-------------------------------------------------------------------**
//* Sample SFTP JCL step to receive a file from SFTP server*
//*-------------------------------------------------------------------**
//*
//STEP01 EXEC PGM=M2SFTP
//*
//SYSPRINT DD  SYSOUT=*
//OUTPUT   DD  SYSOUT=*
//STDOUT   DD  SYSOUT=*
//INPUT    DD  *
open 127.0.0.1
sftpuser
sftppass
cd files
locsite recfm=fb lrecl=150
get AWS.M2.CARDDEMO.CARDDATA.PS.txt +
'AWS.M2.CARDDEMO.CARDDATA.PS2' (replace
quit
/*
//ENVVAR   DD *
MFFTP_VERBOSE_OUTPUT=ON
MFFTP_KEEP=N
/*
//*
//
```

> **Note**
>
> Kami sangat menyarankan untuk menyimpan kredensil FTP dalam file NETRC dan membatasi akses hanya ke pengguna yang berwenang.

Referensi perintah klien Putty SFTP (PSFTP)

Klien PSFTP tidak mendukung semua perintah FTP. Daftar berikut menunjukkan semua perintah yang didukung PSFTP.

Perintah	Deskripsi
!	Jalankan perintah lokal
selamat tinggal	Selesaikan sesi SFTP Anda
cd	Ubah direktori kerja jarak jauh Anda
chmod	Ubah izin dan mode file
tutup	Selesaikan sesi SFTP Anda tetapi jangan keluar dari PSFTP
del	Hapus file di server jarak jauh
dir	Daftar file jarak jauh
keluar	Selesaikan sesi SFTP Anda
memperoleh	Unduh file dari server ke mesin lokal Anda
help	Berikan bantuan
lcd	Ubah direktori kerja lokal
lpwd	Cetak direktori kerja lokal
ls	Daftar file jarak jauh
mget	Unduh beberapa file sekaligus

Perintah	Deskripsi
mkdir	Buat direktori di server jarak jauh
mput	Unggah beberapa file sekaligus
mv	Memindahkan atau mengganti nama file di server jarak jauh
terbuka	Connect ke host
menempatkan	Unggah file dari mesin lokal Anda ke server
pwd	Cetak direktori kerja jarak jauh Anda
berhenti	Selesaikan sesi SFTP Anda
reget	Lanjutkan mengunduh file
ren	Memindahkan atau mengganti nama file di server jarak jauh
reput	Lanjutkan mengunggah file
rm	Hapus file di server jarak jauh
rmdir	Hapus direktori di server jarak jauh

Langkah selanjutnya

Untuk mengunggah dan mengunduh file ke Amazon Simple Storage Service menggunakan SFTP, Anda dapat menggunakan M2SFTP bersama dengan AWS Transfer Family, seperti yang dijelaskan dalam posting blog berikut.

- Menggunakan direktori logis AWS SFTP untuk membangun layanan distribusi data sederhana
- Aktifkan otentikasi kata sandi untuk menggunakan AWS Transfer for SFTPAWS Secrets Manager

Utilitas Batch M2WAIT

M2WAIT adalah program utilitas mainframe yang memungkinkan Anda untuk memperkenalkan periode tunggu dalam skrip JCL Anda dengan menentukan durasi waktu dalam detik, menit, atau jam. Anda dapat memanggil M2WAIT langsung dari JCL dengan melewatkan waktu yang ingin Anda tunggu sebagai parameter input. Secara internal, program M2WAIT memanggil modul yang disediakan Micro Focus `C$SLEEP` untuk menunggu waktu yang ditentukan.

> **Note**
>
> Anda dapat menggunakan alias Micro Focus untuk mengganti apa yang Anda miliki di skrip JCL Anda. Untuk informasi selengkapnya, lihat JES Alias di dokumentasi Micro Focus.

Topik

Platform yang didukung

Anda dapat menggunakan M2WAIT di salah satu platform berikut:

- AWS Modernisasi Mainframe Mikro Fokus Dikelola
- Waktu Proses Fokus Mikro (di Amazon EC2)
- Semua varian produk Micro Focus Enterprise Developer (ED) dan Micro Focus Enterprise Server (ES).

Konfigurasikan M2WAIT untuk Modernisasi AWS Mainframe Dikelola

Jika aplikasi yang dimigrasi berjalan di Modernisasi AWS Mainframe Dikelola, Anda perlu mengonfigurasi M2WAIT sebagai berikut.

- Gunakan program M2WAIT di JCL Anda dengan melewatkan parameter input seperti yang ditunjukkan pada. Sampel JCL

Konfigurasikan M2WAIT untuk runtime Modernisasi AWS Mainframe di Amazon EC2 (termasuk 2.0) AppStream

Jika aplikasi yang dimigrasi berjalan pada runtime Modernisasi AWS Mainframe di Amazon EC2, konfigurasikan M2WAIT sebagai berikut.

1. Ubah Jalur Program Micro Focus JES untuk menyertakan lokasi biner untuk utilitas batch. Jika Anda perlu menentukan beberapa jalur, gunakan titik dua (`:`) untuk memisahkan jalur di Linux dan titik koma (`;`) di Windows.
   - Linux: `/opt/aws/m2/microfocus/utilities/64bit`
   - Windows (32bit): `C:\AWS\M2\MicroFocus\Utilities\32bit`
   - Windows (64bit): `C:\AWS\M2\MicroFocus\Utilities\64bit`
2. Gunakan program M2WAIT di JCL Anda dengan meneruskan parameter input seperti yang ditunjukkan pada. Sampel JCL

Sampel JCL

Untuk menguji instalasi, Anda dapat menggunakan `M2WAIT1.jcl` program ini.

Contoh JCL ini menunjukkan cara memanggil M2WAIT dan meneruskannya beberapa durasi yang berbeda.

```
//M2WAIT1 JOB 'M2WAIT',CLASS=A,MSGCLASS=X,TIME=1440
//*
//* Copyright Amazon.com, Inc. or its affiliates.*
//* All Rights Reserved.*
//*
//*-------------------------------------------------------------------**
//* Wait for 12 Seconds*
//*-------------------------------------------------------------------**
//*
//STEP01 EXEC PGM=M2WAIT,PARM='S012'
//SYSOUT DD SYSOUT=*
//*
//*-------------------------------------------------------------------**
//* Wait for 0 Seconds (defaulted to 10 Seconds)*
//*-------------------------------------------------------------------**
//*
//STEP02 EXEC PGM=M2WAIT,PARM='S000'
```

```
//SYSOUT DD SYSOUT=*
//*
//*--------------------------------------------------------------------**
//* Wait for 1 Minute*
//*--------------------------------------------------------------------**
//*
//STEP03 EXEC PGM=M2WAIT,PARM='M001'
//SYSOUT DD SYSOUT=*
//*
//
```

Utilitas Batch TXT2PDF

TXT2PDF adalah program utilitas mainframe yang biasa digunakan untuk mengonversi file teks ke file PDF. Utilitas ini menggunakan kode sumber yang sama untuk TXT2PDF (z/OS freeware). Kami memodifikasinya untuk berjalan di bawah lingkungan runtime AWS Mainframe Modernization Micro Focus.

Topik

Platform yang didukung

Anda dapat menggunakan TXT2PDF di salah satu platform berikut:

- AWS Modernisasi Mainframe Mikro Fokus Dikelola
- Waktu Proses Fokus Mikro (di Amazon EC2)
- Semua varian produk Micro Focus Enterprise Developer (ED) dan Micro Focus Enterprise Server (ES).

Konfigurasikan TXT2PDF untuk Modernisasi Mainframe Dikelola AWS

Jika aplikasi yang dimigrasi berjalan di Modernisasi AWS Mainframe Dikelola, konfigurasikan TXT2PDF sebagai berikut.

- Buat perpustakaan REXX EXEC yang disebut. `AWS.M2.REXX.EXEC` Unduh modul REXX ini dan salin ke perpustakaan.
  - `TXT2PDF.rex`- TXT2PDF z/OS freeware (dimodifikasi)
  - `TXT2PDFD.rex`- TXT2PDF z/OS freeware (tidak dimodifikasi)
  - `TXT2PDFX.rex`- TXT2PDF z/OS freeware (dimodifikasi)
  - `M2GETOS.rex`- Untuk memeriksa jenis OS (Windows atau Linux)

Untuk menguji konfigurasi Anda, lihatSampel JCL.

Konfigurasikan TXT2PDF untuk runtime Modernisasi AWS Mainframe di Amazon EC2 (termasuk 2.0) AppStream

Jika aplikasi yang dimigrasi berjalan pada runtime Modernisasi AWS Mainframe di Amazon EC2, konfigurasikan TXT2PDF sebagai berikut.

1. Atur variabel lingkungan Micro Focus `MFREXX_CHARSET` ke nilai yang sesuai, seperti “`A`" untuk data ASCII.

   > **Important**
   >
   > Memasukkan nilai yang salah dapat menyebabkan masalah konversi data (dari EBCDIC ke ASCII), membuat PDF yang dihasilkan tidak dapat dibaca atau tidak dapat dioperasikan. Kami merekomendasikan pengaturan `MFREXX_CHARSET` untuk mencocokkan`MF_CHARSET`.

2. Ubah Jalur Program Micro Focus JES untuk menyertakan lokasi biner untuk utilitas batch. Jika Anda perlu menentukan beberapa jalur, gunakan titik dua (`:`) untuk memisahkan jalur di Linux dan titik koma (`;`) di Windows.
   - Linux: `/opt/aws/m2/microfocus/utilities/64bit`
   - Windows (32bit): `C:\AWS\M2\MicroFocus\Utilities\32bit`
   - Windows (64bit): `C:\AWS\M2\MicroFocus\Utilities\64bit`

3. Buat perpustakaan REXX EXEC yang disebut. `AWS.M2.REXX.EXEC` ` Unduh modul REXX ini dan salin ke perpustakaan.

   - `TXT2PDF.rex`- TXT2PDF z/OS freeware (dimodifikasi)
   - `TXT2PDFD.rex`- TXT2PDF z/OS freeware (tidak dimodifikasi)
   - `TXT2PDFX.rex`- TXT2PDF z/OS freeware (dimodifikasi)
   - `M2GETOS.rex`- Untuk memeriksa jenis OS (Windows atau Linux)

Untuk menguji konfigurasi Anda, lihatSampel JCL.

Sampel JCL

Untuk menguji instalasi, Anda dapat menggunakan salah satu dari contoh file JCL berikut.

Txt2pdf1.jcl

Contoh file JCL ini menggunakan nama DD untuk konversi TXT2PDF.

```
//TXT2PDF1 JOB 'TXT2PDF1',CLASS=A,MSGCLASS=X,TIME=1440
//*
//* Copyright Amazon.com, Inc. or its affiliates.*
//* All Rights Reserved.*
//*
//*--------------------------------------------------------------------**
//* PRE DELETE*
//*--------------------------------------------------------------------**
//*
//PREDEL  EXEC PGM=IEFBR14
//*
//DD01     DD DSN=AWS.M2.TXT2PDF1.PDF.VB,
//            DISP=(MOD,DELETE,DELETE)
//*
//DD02     DD DSN=AWS.M2.TXT2PDF1.PDF,
//            DISP=(MOD,DELETE,DELETE)
//*
//*--------------------------------------------------------------------**
//* CALL TXT2PDF TO CONVERT FROM TEXT TO PDF (VB)*
//*--------------------------------------------------------------------**
//*
//STEP01 EXEC PGM=IKJEFT1B
//*
//SYSEXEC  DD DISP=SHR,DSN=AWS.M2.REXX.EXEC
```

```
//*
//INDD     DD *
1THIS IS THE FIRST LINE ON THE PAGE 1
0THIS IS THE THIRD LINE ON THE PAGE 1
-THIS IS THE   6TH LINE ON THE PAGE 1
THIS IS THE   7TH LINE ON THE PAGE 1
+_____________________________________ - OVERSTRIKE 7TH LINE
1THIS IS THE FIRST LINE ON THE PAGE 2
0THIS IS THE THIRD LINE ON THE PAGE 2
-THIS IS THE   6TH LINE ON THE PAGE 2
THIS IS THE   7TH LINE ON THE PAGE 2
+_____________________________________ - OVERSTRIKE 7TH LINE
/*
//*
//OUTDD    DD DSN=AWS.M2.TXT2PDF1.PDF.VB,
//            DISP=(NEW,CATLG,DELETE),
//            DCB=(LRECL=256,DSORG=PS,RECFM=VB,BLKSIZE=0)
//*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD DDNAME=SYSIN
//*
//SYSIN    DD *
%TXT2PDF BROWSE Y IN DD:INDD +
OUT DD:OUTDD +
CC YES
/*
//*
//*--------------------------------------------------------------------**
//* CONVERT PDF (VB) TO PDF (LSEQ - BYTE STREAM)*
//*--------------------------------------------------------------------**
//*
//STEP02 EXEC PGM=VB2LSEQ
//*
//INFILE   DD DSN=AWS.M2.TXT2PDF1.PDF.VB,DISP=SHR
//*
//OUTFILE  DD DSN=AWS.M2.TXT2PDF1.PDF,
//            DISP=(NEW,CATLG,DELETE),
//            DCB=(LRECL=256,DSORG=PS,RECFM=LSEQ,BLKSIZE=0)
//*
//SYSOUT   DD SYSOUT=*
//*
//
```

Txt2pdf2.jcl

Contoh JCL ini menggunakan nama DSN untuk konversi TXT2PDF.

```
//TXT2PDF2 JOB 'TXT2PDF2',CLASS=A,MSGCLASS=X,TIME=1440
//*
//* Copyright Amazon.com, Inc. or its affiliates.*
//* All Rights Reserved.*
//*
//*--------------------------------------------------------------------**
//* PRE DELETE*
//*--------------------------------------------------------------------**
//*
//PREDEL  EXEC PGM=IEFBR14
//*
//DD01     DD DSN=AWS.M2.TXT2PDF2.PDF.VB,
//            DISP=(MOD,DELETE,DELETE)
//*
//DD02     DD DSN=AWS.M2.TXT2PDF2.PDF,
//            DISP=(MOD,DELETE,DELETE)
//*
//*--------------------------------------------------------------------**
//* CALL TXT2PDF TO CONVERT FROM TEXT TO PDF (VB)*
//*--------------------------------------------------------------------**
//*
//STEP01 EXEC PGM=IKJEFT1B
//*
//SYSEXEC  DD DISP=SHR,DSN=AWS.M2.REXX.EXEC
//*
//INDD     DD *
1THIS IS THE FIRST LINE ON THE PAGE 1
0THIS IS THE THIRD LINE ON THE PAGE 1
-THIS IS THE   6TH LINE ON THE PAGE 1
THIS IS THE   7TH LINE ON THE PAGE 1
+_____________________________________ - OVERSTRIKE 7TH LINE
1THIS IS THE FIRST LINE ON THE PAGE 2
0THIS IS THE THIRD LINE ON THE PAGE 2
-THIS IS THE   6TH LINE ON THE PAGE 2
THIS IS THE   7TH LINE ON THE PAGE 2
+_____________________________________ - OVERSTRIKE 7TH LINE
/*
//*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD DDNAME=SYSIN
```

```
//*
//SYSIN    DD *
%TXT2PDF BROWSE Y IN DD:INDD +
OUT 'AWS.M2.TXT2PDF2.PDF.VB' +
CC YES
/*
//*
//*-------------------------------------------------------------------**
//* CONVERT PDF (VB) TO PDF (LSEQ - BYTE STREAM)*
//*-------------------------------------------------------------------**
//*
//STEP02 EXEC PGM=VB2LSEQ
//*
//INFILE   DD DSN=AWS.M2.TXT2PDF2.PDF.VB,DISP=SHR
//*
//OUTFILE  DD DSN=AWS.M2.TXT2PDF2.PDF,
//            DISP=(NEW,CATLG,DELETE),
//            DCB=(LRECL=256,DSORG=PS,RECFM=LSEQ,BLKSIZE=0)
//*
//SYSOUT   DD SYSOUT=*
//*
//
```

Pengubahan

Untuk membuat program TXT2PDF berjalan di lingkungan runtime AWS Mainframe Modernization Micro Focus, kami membuat perubahan berikut:

- Perubahan pada kode sumber untuk memastikan kompatibilitas dengan runtime Micro Focus REXX
- Perubahan untuk memastikan bahwa program dapat berjalan pada sistem operasi Windows dan Linux
- Modifikasi untuk mendukung runtime EBCDIC dan ASCII

Referensi

Referensi TXT2PDF dan kode sumber:

- Konverter teks ke PDF
- z/OS Freeware TCP/IP dan Mail Tools

- Panduan Referensi Pengguna TXT2PDF

Utilitas Batch M2DFUTIL

M2DFUTIL adalah program utilitas JCL yang menyediakan fungsi backup, restore, delete, dan copy pada dataset, mirip dengan dukungan yang disediakan oleh utilitas mainframe ADRDSSU. Program ini mempertahankan banyak parameter SYSIN dari ADRDSSU, yang merampingkan proses untuk bermigrasi ke utilitas baru ini.

Topik

Platform yang didukung

Anda dapat menggunakan M2DFUTIL di salah satu platform berikut:

- Micro Focus ES pada Windows (64 bit dan 32 bit)
- Micro Focus ES di Linux (64 bit)

Persyaratan platform

M2DFUTIL bergantung pada memanggil skrip untuk melakukan tes ekspresi reguler. Di Windows, Anda harus menginstal Windows Services for Linux (WSL) agar skrip ini dapat dijalankan.

Dukungan future yang direncanakan

Fitur yang saat ini tidak tersedia dari utilitas ADRDSSU mainframe, tetapi berada dalam lingkup future meliputi:

- M2 Dikelola
- VSAM
- Dukungan COPY untuk penggantian nama file
- Ganti nama dukungan untuk RESTORE
- Beberapa INCLUDE dan EXCLUDE
- BY klausa untuk subpemilihan oleh DSORG, CREDT, EXPDT
- Klausa MWAIT untuk mencoba lagi kegagalan enqueue
- Dukungan penyimpanan S3 untuk DUMP/RESTORE

Lokasi aset

Modul beban untuk utilitas ini disebut `M2DFUTIL.so` di Linux dan `M2DFUTIL.dll` Windows. Modul beban ini dapat ditemukan di lokasi berikut:

- Linux: `/opt/aws/m2/microfocus/utilities/64bit`
- Windows (32 bit): `C:\AWS\M2\MicroFocus\Utilities\32bit`
- Windows (64 bit): `C:\AWS\M2\MicroFocus\Utilities\64bit`

Script yang digunakan untuk pengujian ekspresi reguler disebut`compare.sh`. Skrip ini dapat ditemukan di lokasi berikut:

- Linux: `/opt/aws/m2/microfocus/utilities/scripts`
- Windows (32 bit): `C:\AWS\M2\MicroFocus\Utilities\scripts`

Konfigurasikan runtime Modernisasi M2DFUTIL atau AWS Mainframe di Amazon EC2 (termasuk 2.0) AppStream

Konfigurasikan wilayah Server Perusahaan Anda dengan yang berikut ini:

- Tambahkan variabel berikut di [ES-Environment]
  - `M2DFUTILS_BASE_LOC`- Lokasi default untuk output DUMP
  - `M2DFUTILS_SCRIPTPATH`- Lokasi `compare.sh` skrip yang didokumentasikan di Lokasi Aset
  - `M2DFUTILS_VERBOSE`- [VERBOSE atau NORMAL]. Ini mengontrol tingkat detail dalam `SYSPRINT` output

- Verifikasi bahwa jalur modul beban ditambahkan ke `JES\Configuration\JES Program Path` pengaturan
- Verifikasi bahwa skrip di direktori utilitas telah menjalankan izin. Anda dapat menambahkan izin jalankan menggunakan `chmod + x <script name>` perintah, di lingkungan Linux

Sintaks umum

MEMBUANG

Menyediakan kemampuan untuk menyalin file dari lokasi katalog saat ini ke lokasi cadangan. Lokasi ini saat ini harus berupa sistem file.

Proses

DUMP akan melakukan hal berikut:

1. Buat direktori lokasi target.
2. Katalog direktori lokasi target sebagai anggota PDS.
3. Tentukan file yang akan disertakan dengan memproses parameter INCLUDE.
4. Hapus pilihan file yang disertakan dengan memproses parameter EXCLUDE.
5. Tentukan apakah file yang dibuang akan DIHAPUS.
6. Enqueue file yang akan diproses.
7. Salin file.
8. Ekspor file yang disalin yang dikatalogkan informasi DCB ke file samping di lokasi target untuk membantu operasi RESTORE di masa mendatang.

Sintaks

```
DUMP
TARGET ( TARGET LOCATION  )    -
INCLUDE ( DSN. )
[ EXCLUDE ( DSN ) ]
[ CANCEL | IGNORE ]
[ DELETE ]
```

Parameter yang diperlukan

Berikut ini adalah parameter yang diperlukan untuk DUMP:

- `SYSPRINT DD NAME`- Untuk memuat informasi pencatatan tambahan
- `TARGET`- Lokasi target. Itu bisa berupa:
  - Jalur lengkap lokasi pembuangan
  - Nama subdirektori dibuat di lokasi yang ditentukan dalam variabel M2DFUTILS_BASE_LOC
- `INCLUDE`- Entah DSNAME bernama tunggal atau string pencarian DSN mainframe yang valid
- `EXCLUDE`- Entah DSNAME bernama tunggal atau string pencarian DSN mainframe yang valid

Parameter opsional

- BATAL - Batalkan jika terjadi kesalahan. File yang diproses akan disimpan
- (Default) IGNORE - Abaikan kesalahan dan proses sampai akhir
- DELETE - Jika tidak ada kesalahan ENQ terjadi, maka file tersebut dihapus dan tidak dikatalogkan

DELETE

Memberikan kemampuan untuk menghapus massal dan file uncatalog. File tidak dicadangkan.

Proses

DELETE akan melakukan hal berikut:

1. Tentukan file yang akan disertakan dengan memproses parameter INCLUDE.
2. Hapus pilihan file yang disertakan dengan memproses parameter EXCLUDE.
3. Enqueue file yang akan diproses. Mengatur disposisi ke OLD, DELETE, KEEP.

Sintaks

```
DELETE
INCLUDE ( DSN )
[ EXCLUDE ( DSN ) ]
[ CANCEL | IGNORE ]
[ DELETE ]
```

Parameter yang diperlukan

Berikut ini adalah parameter yang diperlukan untuk DELETE:

- `SYSPRINT DD NAME`- Untuk memuat informasi pencatatan tambahan
- `INCLUDE`- Entah DSNAME bernama tunggal atau string pencarian DSN mainframe yang valid
- `EXCLUDE`- Entah DSNAME bernama tunggal atau string pencarian DSN mainframe yang valid

Parameter opsional

- BATAL - Batalkan jika terjadi kesalahan. File yang diproses akan disimpan
- (Default) IGNORE - Abaikan kesalahan dan proses sampai akhir

MEMULIHKAN

Memberikan kemampuan untuk memulihkan file yang sebelumnya dicadangkan menggunakan DUMP. File dikembalikan ke lokasi katalog asli kecuali RENAME digunakan untuk mengubah DSNAME yang dipulihkan.

Proses

RESTORE akan melakukan hal berikut:

1. Validasi direktori lokasi sumber.
2. Tentukan file yang akan disertakan dengan memproses file ekspor katalog.
3. Hapus pilihan file yang disertakan dengan memproses parameter EXCLUDE.
4. Enqueue file yang akan diproses.
5. File katalog yang tidak dikatalogkan berdasarkan informasi ekspornya.
6. Jika file sudah dikatalogkan dan informasi katalog ekspor sama, RESTORE akan menggantikan kumpulan data yang dikatalogkan jika opsi REPLACE disetel.

Sintaks

```
RESTORE
SOURCE ( TARGET LOCATION )
INCLUDE ( DSN )
[ EXCLUDE ( DSN ) ]
[ CANCEL | IGNORE ]
[ REPLACE]
```

Parameter yang diperlukan

Berikut ini adalah parameter yang diperlukan untuk RESTORE:

- `SYSPRINT DD NAME`- Untuk memuat informasi pencatatan tambahan
- `SOURCE`- Lokasi sumber. Itu bisa berupa:
  - Jalur lengkap lokasi pembuangan
  - Nama subdirektori dibuat di lokasi yang ditentukan dalam variabel M2DFUTILS_BASE_LOC
- `INCLUDE`- Entah DSNAME bernama tunggal atau string pencarian DSN mainframe yang valid
- `EXCLUDE`- Entah DSNAME bernama tunggal atau string pencarian DSN mainframe yang valid

Parameter opsional

- BATAL - Batalkan jika ada kesalahan. File yang diproses dipertahankan
- (Default) IGNORE - Abaikan kesalahan dan proses sampai akhir
- REPLACE - Jika file yang dipulihkan sudah dikatalogkan dan catatan katalognya sama, maka ganti file yang dikatalogkan

Contoh JCL

Lowongan kerja DUMP

Pekerjaan ini akan membuat subdirektori yang disebut`TESTDUMP`. Ini adalah lokasi cadangan default yang ditentukan oleh variabel M2DFUTILS_BASE_LOC. Ini akan membuat pustaka PDS untuk cadangan ini disebut`M2DFUTILS.TESTDUMP`. Data katalog yang diekspor disimpan dalam file berurutan baris di direktori cadangan yang disebut. `CATDUMP.DAT` Semua file yang dipilih akan disalin ke direktori cadangan ini.

```
//M2DFDMP JOB 'M2DFDMP',CLASS=A,MSGCLASS=X
//STEP001  EXEC PGM=M2DFUTIL
//SYSPRINT DD DSN=TESTDUMP.SYSPRINT,
//        DISP=(NEW,CATLG,DELETE),
//        DCB=(RECFM=LSEQ,LRECL=256)
//SYSIN    DD *
DUMP TARGET(TESTDUMP)               -
     INCLUDE(TEST.FB.FILE*.ABC)     -
 CANCEL
```

```
/*
//
```

HAPUS pekerjaan

Pekerjaan ini akan menghapus semua file dari katalog yang cocok dengan parameter INCLUDE.

```
/M2DFDEL JOB 'M2DFDEL',CLASS=A,MSGCLASS=X
//STEP001  EXEC PGM=M2DFUTIL
//SYSPRINT DD DSN=TESTDEL.SYSPRINT,
//        DISP=(NEW,CATLG,DELETE),
//        DCB=(RECFM=LSEQ,LRECL=256)
//SYSPRINT DD SYSOUT=A
//SYSIN    DD *
  DELETE                              -
     INCLUDE(TEST.FB.FILE*.ABC)       -
 CANCEL
/*
 //
```

MENGEMBALIKAN pekerjaan

Pekerjaan ini akan mengembalikan file yang cocok dengan parameter INCLUDE dari lokasi TESTDUMP cadangan. File yang dikatalogkan akan diganti jika file yang dikatalogkan sama dengan yang ada di ekspor CATDUMP dan opsi REPLACE ditentukan.

```
//M2DFREST JOB 'M2DFREST',CLASS=A,MSGCLASS=X
//STEP001  EXEC PGM=M2DFUTIL
////SYSPRINT DD DSN=TESTREST.SYSPRINT,
//        DISP=(NEW,CATLG,DELETE),
//        DCB=(RECFM=LSEQ,LRECL=256)
//SYSPRINT DD SYSOUT=A
//SYSIN    DD *
RESTORE SOURCE(TESTDUMP)              -
     INCLUDE(TEST.FB.FILE*.ABC)       -
 IGNORE
 REPLACE
/*
//
```

Utilitas Batch M2RUNCMD

Anda dapat menggunakan M2RUNCMD, program utilitas batch, untuk menjalankan perintah Micro Focus, skrip, dan panggilan sistem langsung dari JCL alih-alih menjalankannya dari terminal atau command prompt. Output dari perintah dicatat ke log spool pekerjaan batch.

Topik

Platform yang didukung

Anda dapat menggunakan M2RUNCMD pada platform berikut:

- Waktu Proses Fokus Mikro (di Amazon EC2)
- Semua varian produk Micro Focus Enterprise Developer (ED) dan Micro Focus Enterprise Server (ES).

Konfigurasikan M2RUNCMD untuk runtime Modernisasi AWS Mainframe di Amazon EC2 (termasuk 2.0) AppStream

Jika aplikasi yang dimigrasi berjalan pada runtime Modernisasi AWS Mainframe di Amazon EC2, konfigurasikan M2RUNCMD sebagai berikut.

- Ubah Jalur Program Micro Focus JES untuk menyertakan lokasi biner untuk utilitas batch. Jika Anda harus menentukan beberapa jalur, gunakan titik dua (:) untuk memisahkan jalur di Linux dan titik koma (;) di Windows.
  - Linux: `/opt/aws/m2/microfocus/utilities/64bit`
  - Windows (32bit): `C:\AWS\M2\MicroFocus\Utilities\32bit`
  - Windows (64bit): `C:\AWS\M2\MicroFocus\Utilities\64bit`

Contoh JCL

Untuk menguji instalasi, Anda dapat menggunakan salah satu dari contoh JCL berikut.

Runscrl1.jcl

Contoh JCL ini membuat skrip dan menjalankannya. Langkah pertama membuat skrip yang disebut `/tmp/TEST_SCRIPT.sh` dan dengan konten dari data SYSUT1 in-stream. Langkah kedua menetapkan izin jalankan dan menjalankan skrip yang dibuat pada langkah pertama. Anda juga dapat memilih untuk hanya melakukan langkah kedua untuk menjalankan Micro Focus dan perintah sistem yang sudah ada.

```
//RUNSCRL1 JOB 'RUN SCRIPT',CLASS=A,MSGCLASS=X,TIME=1440
//*
//*
//*--------------------------------------------------------------------*
//*  CREATE SCRIPT (LINUX)
//*--------------------------------------------------------------------*
//*
//STEP0010 EXEC PGM=IEBGENER
//*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//*
//SYSUT1   DD *
#!/bin/bash

set -x

## ECHO PATH ENVIRONMNET VARIABLE
echo $PATH

## CLOSE/DISABLE VSAM FILE
casfile -r$ES_SERVER -oc  -ed -dACCTFIL

## OPEN/ENABLE VSAM FILE
casfile -r$ES_SERVER -ooi -ee -dACCTFIL

exit $?
/*
//SYSUT2   DD DSN=&&TEMP,
//            DISP=(NEW,CATLG,DELETE),
//            DCB=(RECFM=LSEQ,LRECL=300,DSORG=PS,BLKSIZE=0)
//*MFE: %PCDSN='/tmp/TEST_SCRIPT.sh'
//*
//*--------------------------------------------------------------------*
//*   RUN SCRIPT (LINUX)                                               *
```

```
//*-------------------------------------------------------------------*
//*
//STEP0020 EXEC PGM=RUNCMD
//*
//SYSOUT  DD  SYSOUT=*
//*
//SYSIN   DD *
*RUN SCRIPT
 sh /tmp/TEST_SCRIPT.sh
/*
//
```

SYSOUT

Output dari perintah atau script yang dijalankan, ditulis ke dalam SYSOUT log. Untuk setiap perintah yang dilakukan, ini menampilkan perintah, output, dan kode pengembalian.

```
************ CMD Start ************

CMD_STR: sh /tmp/TEST_SCRIPT.sh

CMD_OUT:


+ echo /opt/microfocus/EnterpriseServer/bin:/sbin:/bin:/usr/sbin:/usr/bin
/opt/microfocus/EnterpriseServer/bin:/sbin:/bin:/usr/sbin:/usr/bin
+ casfile -rMYDEV -oc -ed -dACCTFIL

-Return Code:   0

Highest return code:    0

+ casfile -rMYDEV -ooi -ee -dACCTFIL

-Return Code:   8

Highest return code:    8

+ exit 8


CMD_RC=8
```

```
************  CMD End  ************
```

RunCMDL1.jcl

Contoh JCL ini menggunakan RUNCMD untuk menjalankan beberapa perintah.

```
//RUNCMDL1 JOB 'RUN CMD',CLASS=A,MSGCLASS=X,TIME=1440
//*
//*
//*--------------------------------------------------------------------*
//*   RUN SYSTEM COMMANDS                                              *
//*--------------------------------------------------------------------*
//*
//STEP0001 EXEC PGM=RUNCMD
//*
//SYSOUT  DD  SYSOUT=*
//*
//SYSIN   DD *
*LIST DIRECTORY
 ls
*ECHO PATH ENVIRONMNET VARIABLE
 echo $PATH
/*
//
```

AWS Replikasi data Modernisasi Mainframe dengan Tepat

AWS Modernisasi Mainframe menawarkan berbagai Amazon Machine Images (AMI). AMI ini memfasilitasi penyediaan instans Amazon EC2 secara cepat, menciptakan lingkungan yang disesuaikan untuk replikasi data dari sistem Mainframe hingga menggunakan Exactly. AWS Panduan ini memberikan langkah-langkah yang diperlukan untuk mengakses dan menggunakan AMI ini.

Prasyarat

- Pastikan Anda memiliki akses administrator ke AWS akun tempat Anda dapat membuat instans Amazon EC2.
- Verifikasi bahwa layanan Modernisasi AWS Mainframe tersedia di Wilayah tempat Anda berencana membuat instans Amazon EC2. Lihat Daftar Layanan AWS yang Tersedia berdasarkan Wilayah.
- Identifikasi Amazon Virtual Private Cloud (Amazon VPC) tempat instans Amazon EC2 akan dibuat.
- Saat membuat instans Amazon EC2 di VPC Amazon, pastikan tabel rute terkait memiliki gateway internet atau gateway NAT.

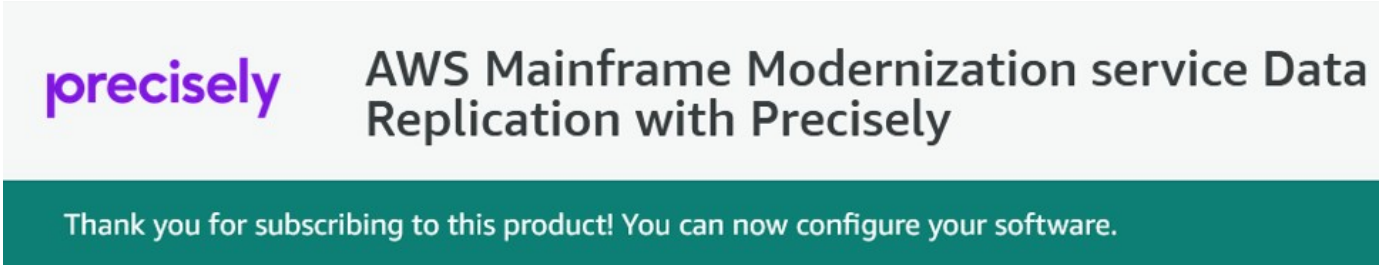
> **Note**
>
> Replikasi data yang berhasil mengharuskan instans AWS EC2 memiliki akses komunikasi ke AWS Marketplace. Jika ada masalah konektivitas dengan AWS Marketplace, proses replikasi akan gagal.

Berlangganan Gambar Mesin Amazon

Saat berlangganan produk AWS Marketplace, Anda dapat meluncurkan instans dari AMI produk.

1. Masuk ke AWS Management Console dan buka AWS Marketplace konsol di https://console.aws.amazon.com/marketplace.
2. Pilih Kelola langganan.
3. Salin dan tempel tautan berikut ke bilah alamat browser: https://aws.amazon.com/marketplace/pp/prodview-en3xrbgzbs3dk
4. Pilih Lanjutkan Berlangganan.

5. Jika syarat dan ketentuan dapat diterima, pilih Terima Syarat. Langganan mungkin memakan waktu beberapa menit untuk diproses.
6. Tunggu pesan terima kasih muncul, seperti yang ditunjukkan di bawah ini. Pesan ini mengonfirmasi bahwa Anda telah berhasil berlangganan produk.



7. Di panel navigasi kiri, pilih Kelola langganan. Tampilan ini menunjukkan kepada Anda semua langganan yang telah Anda langgani.

Luncurkan AWS replikasi data Modernisasi Mainframe dengan Tepat

1. Buka AWS Marketplace konsol di https://console.aws.amazon.com/marketplace.
2. Di panel navigasi kiri, pilih Kelola langganan.
3. Temukan AMI yang ingin Anda luncurkan, dan pilih Luncurkan instance baru.
4. Di bawah Wilayah, pilih Wilayah yang diizinkan terdaftar.
5. Pilih Lanjutkan untuk meluncurkan melalui EC2. Tindakan ini membawa Anda ke konsol Amazon EC2.
6. Masukkan nama untuk server.
7. Pilih jenis instans yang sesuai dengan kinerja proyek dan persyaratan biaya Anda. Titik awal yang disarankan untuk ukuran misalnya adalah`c5.2xLarge`.
8. Pilih key pair yang ada atau buat dan simpan yang baru. Untuk informasi tentang pasangan kunci, lihat pasangan kunci Amazon EC2 dan instans Linux di Panduan Pengguna Amazon EC2 untuk Instans Linux.
9. Edit pengaturan jaringan dan pilih VPC yang terdaftar yang diizinkan dan subnet yang sesuai.
10. Pilih grup keamanan yang ada atau buat yang baru. Selain memungkinkan akses SSH (secara default pada port 22), untuk replikasi data dengan instance EC2 server yang tepat, biasanya memungkinkan lalu lintas TCP ke port defaultnya 2626.
11. Konfigurasikan penyimpanan untuk instans Amazon EC2.

12. Tinjau ringkasan dan pilih Launch instance. Agar peluncuran berhasil, jenis instance harus valid. Jika peluncuran gagal, pilih Edit konfigurasi instans dan pilih jenis instans yang berbeda.
13. Setelah Anda melihat pesan sukses, pilih Connect to instance.
14. Buka konsol Amazon EC2 di https://console.aws.amazon.com/ec2/.
15. Di panel navigasi kiri, di bawah menu Instans, pilih Instans.
16. Di panel utama, periksa status instans Anda.

Buat kebijakan IAM

Agar berhasil mengoperasikan instans EC2 Modernisasi AWS Mainframe yang diterapkan melalui AWS Marketplace daftar kami, Anda harus mengonfigurasi peran dan kebijakan IAM. Penyiapan IAM yang disesuaikan secara khusus ini bukan opsional; ini mengizinkan instans Amazon EC2 Anda untuk berinteraksi dengan layanan. AWS Marketplace Peran dan kebijakan IAM memungkinkan Modernisasi AWS Mainframe untuk merekam data penggunaan secara akurat, yang penting untuk penagihan yang tepat. Gagal mengimplementasikan konfigurasi ini dapat menyebabkan upaya replikasi data yang gagal dan gangguan operasional.

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pada panel navigasi di sebelah kiri, pilih Kebijakan.
3. Jika ini adalah pertama kalinya Anda memilih Kebijakan, halaman Selamat Datang di Kebijakan Terkelola akan muncul. Pilih Memulai.
4. Di bagian atas halaman, pilih Buat kebijakan.
5. Di bagian Editor kebijakan, pilih opsi JSON.
6. Masukkan kebijakan JSON berikut.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": ["aws-marketplace:MeterUsage"],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

Membuat peran IAM

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi, pilih Peran, lalu pilih Buat peran.
3. Di bagian Jenis entitas tepercaya, pilih AWS layanan.
4. Di bagian Kasus penggunaan, di bawah Layanan atau kasus penggunaan, pilih Amazon EC2.
5. Pilih Berikutnya.
6. Dalam daftar kebijakan, pilih Pelanggan yang dikelola dari menu tarik-turun Filter menurut Jenis, lalu masukkan nama kebijakan yang Anda buat. Pilih kotak centang di sebelah nama kebijakan.
7. Pilih Berikutnya.
8. Masukkan nama dan, secara opsional, deskripsi untuk peran tersebut.
9. Tinjau kebijakan kepercayaan dan izin, lalu pilih Buat peran.

Lampirkan peran IAM ke instans Amazon EC2

1. Buka konsol Amazon EC2 di https://console.aws.amazon.com/ec2/.
2. Di panel navigasi, pilih Instans.
3. Pilih instans Amazon EC2 Anda.
4. Dari menu Tindakan, pilih Keamanan, lalu pilih Ubah peran IAM.
5. Pilih peran yang akan dilampirkan ke instance Anda, lalu pilih Perbarui peran IAM.

Integrasi Charon

Pengantar Charon-SSP

Pada tahun 1987, Sun Microsystems merilis prosesor SPARC V7, prosesor RISC 32-bit. SPARC V8 diikuti pada tahun 1990 - revisi dari SPARC V7 asli, dengan penyertaan instruksi pembagian dan perkalian perangkat keras yang paling menonjol. Prosesor SPARC V8 membentuk dasar untuk sejumlah server dan workstation seperti SparcStation 5, 10 dan 20. Pada tahun 1993, SPARC V8 diikuti oleh prosesor SPARC V9 64-bit. Ini juga menjadi dasar untuk sejumlah server dan workstation, seperti Enterprise 250 dan 450.

Karena keusangan perangkat keras dan kurangnya suku cadang atau suku cadang yang diperbaharui, perangkat lunak dan sistem yang dikembangkan untuk workstation dan server berbasis SPARC yang lebih tua ini menjadi lebih sulit untuk dirawat. Untuk memenuhi kebutuhan berkelanjutan untuk sistem end-of-life berbasis SPARC tertentu, Stromasys S.A. mengembangkan jajaran produk emulator SPARC Charon-SSP. Produk berikut adalah pengganti mesin virtual berbasis perangkat lunak untuk sistem SPARC perangkat keras asli yang ditentukan. Berikut ini adalah gambaran umum dari keluarga perangkat keras yang ditiru.

Charon-SSP/4M mengemulasi perangkat keras SPARC berikut:

- Keluarga Sun-4m (diwakili oleh Sun SparcStation 20): awalnya, varian multiprosesor Sun-4, berdasarkan bus modul prosesor MBus yang diperkenalkan dalam seri SparcServer 600MP. Arsitektur Sun-4m kemudian juga mencakup sistem uniprosesor non-MBUS seperti SPAR Cstation 5, memanfaatkan prosesor SPARC V8-Architecture. Didukung mulai dengan SunOS 4.1.2 dan oleh Solaris 2.1 hingga Solaris 9. Dukungan SparcServer 600MP dijatuhkan setelah Solaris 2.5.1.

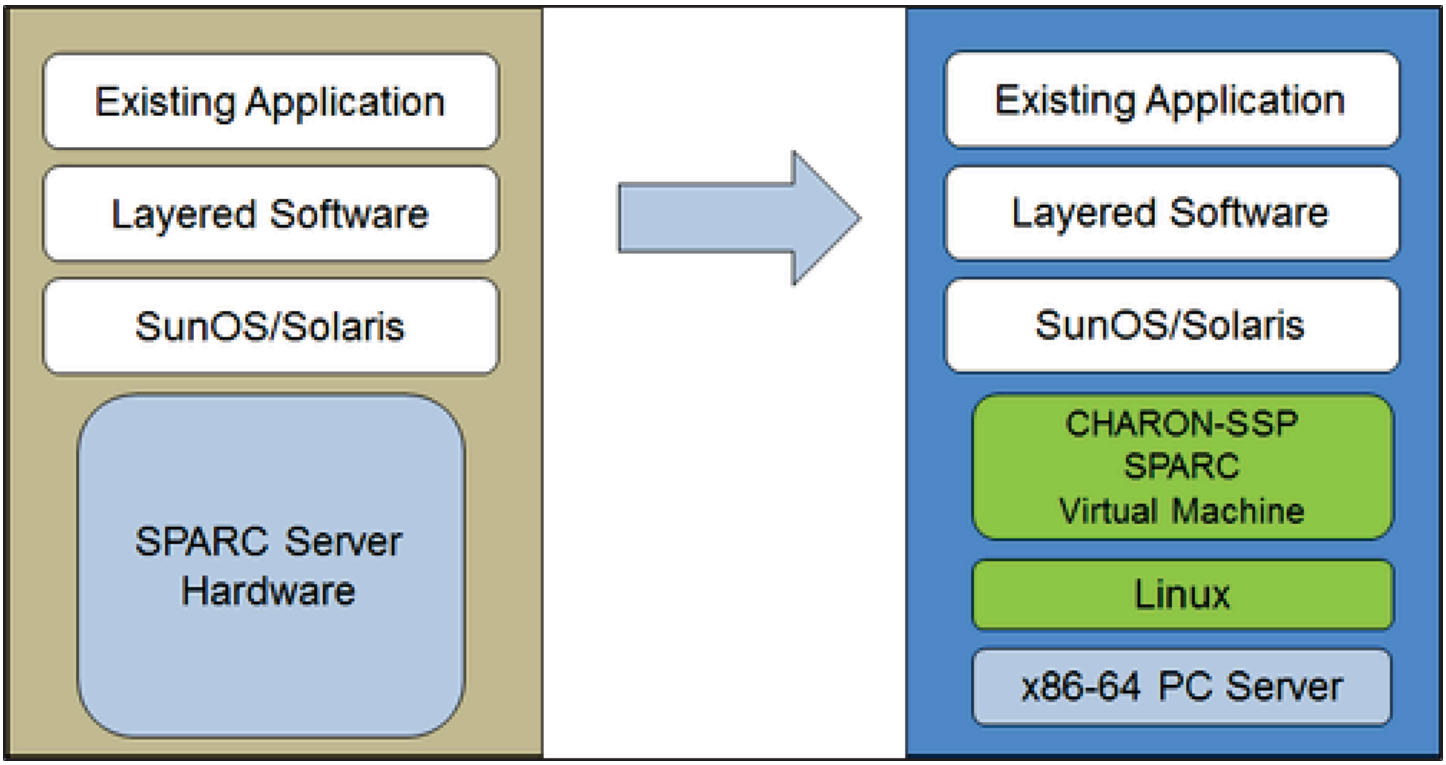
Charon-SSP/4U (+) mengemulasi perangkat keras SPARC berikut:

- Keluarga Sun-4u (diwakili oleh Sun Enterprise 450): (U untuk UltraSPARC) - varian ini memperkenalkan arsitektur prosesor SPARC V9 64-bit dan interkoneksi prosesor UPA yang pertama kali digunakan dalam seri Sun Ultra. Didukung oleh Solaris versi 32-bit mulai dari versi 2.5.1. Rilis Solaris 64-bit pertama untuk Sun-4u adalah Solaris 7. Dukungan UltraSparc I dijatuhkan setelah Solaris 9. Solaris 10 mendukung implementasi Sun-4U dari UltraSP ARC II ke UltraSPARC IV.

Charon-SSP/4V (+) mengemulasi perangkat keras SPARC berikut:

- Keluarga Sun-4v (diwakili oleh SPARC T2 dan T4): variasi ini menambahkan virtualisasi prosesor hypervisor ke Sun-4u; diperkenalkan dalam prosesor multicore Ultra SPARC T1. Perangkat keras yang dipilih didukung oleh Solaris versi 10 mulai dari rilis 3/05 HW2 (sebagian besar model - termasuk perangkat keras yang ditiru oleh Charon-SSP - memerlukan versi Solaris 10 yang lebih baru). Beberapa versi Solaris 11 juga didukung.

Gambar berikut menunjukkan konsep dasar migrasi perangkat keras fisik ke emulator.



Mesin virtual Charon-SSP memungkinkan pengguna komputer berbasis Sun dan Oracle Spark untuk mengganti perangkat keras asli mereka dengan cara yang memerlukan sedikit atau tidak ada perubahan pada konfigurasi sistem asli. Ini berarti Anda dapat terus menjalankan aplikasi dan data Anda tanpa perlu beralih atau port ke platform lain. Perangkat lunak Charon-SSP berjalan pada komoditas, sistem Intel 64-bit yang memastikan perlindungan berkelanjutan atas investasi Anda.

Charon-SSP/4U+ mendukung platform SPARC virtual yang sama dengan Charon-SSP/4U, dan Charon-SSP/4V+ sama dengan Charon-SSP/4V. Namun, versi 4U+dan 4V+memanfaatkan teknologi virtualisasi berbantuan perangkat keras Intel VTX/EPT dan AMD AMD-V/NPT dalam CPU modern untuk menawarkan kinerja CPU virtual yang lebih baik. Charon-SSP/4U+ dan Charon-SSP/4V

+memerlukan CPU dengan dukungan VT-X/EPT atau AMD-V/NPT dan harus diinstal pada sistem host khusus. Menjalankan varian produk ini dalam VM (misalnya, di VMware) tidak didukung.

> **Note**
>
> Jika Anda berencana untuk menjalankan Charon-SSP/4U+ atau 4V+di lingkungan cloud, hubungi Stromasys atau Stromasys VAR untuk mendiskusikan kebutuhan Anda.

Sistem operasi tamu yang didukung

Mesin virtual Charon-SSP/4M mendukung rilis sistem operasi tamu berikut:

- SunOS 4.1.3 - 4.1.4
- Solaris 2.3 untuk Solaris 9

Mesin virtual Charon-SSP/4U (+) mendukung rilis sistem operasi tamu berikut:

- Solaris 2.5.1 para Solaris 10

Mesin virtual Charon-SSP/4V (+) mendukung rilis sistem operasi tamu berikut:

- Solaris 10 (dimulai dengan pembaruan 4, 08/07) dan Solaris 11.1 ke Solaris 11.4

Untuk Charon-SSP/4V (+), perhatikan hal berikut:

- Untuk SPARC T4 yang ditiru, versi Solaris 10 yang didukung adalah: Oracle Solaris 10 1/13, Oracle Solaris 10 8/11, dan Solaris 10 9/10, atau Solaris 10 10/09 dengan set patch Oracle Solaris 10 8/11.
- Model SPARC T4 yang ditiru adalah prasyarat untuk menjalankan Solaris 11.4 di emulator.
- Zona kernel Solaris tidak didukung.

Prasyarat instance cloud Charon-SSP

Dengan memilih jenis atau bentuk instans, Anda memilih perangkat keras virtual yang akan digunakan untuk instance host Charon-SSP di cloud. Oleh karena itu, pemilihan jenis atau bentuk

instance menentukan karakteristik perangkat keras dari perangkat keras host virtual Charon-SSP (misalnya, berapa banyak inti CPU dan berapa banyak memori yang akan dimiliki sistem host Charon virtual Anda).

> **Note**
>
> Jika Anda menggunakan image marketplace Charon-SSP untuk meluncurkan instans Anda, semua persyaratan sistem operasi host Linux terpenuhi.

Persyaratan perangkat keras minimum dijelaskan di bawah ini.

Poin penting mengenai pedoman ukuran:

- Pedoman ukuran di bawah ini - khususnya mengenai jumlah inti CPU host dan memori host - menunjukkan persyaratan minimum. Setiap situasi penerapan harus ditinjau dan ukuran host yang sebenarnya harus disesuaikan seperlunya. Misalnya, jumlah core CPU yang tersedia untuk I/O harus ditingkatkan jika aplikasi tamu menghasilkan beban I/O yang tinggi. Juga, sistem dengan banyak CPU yang ditiru biasanya dapat membuat beban I/O yang lebih tinggi dan dengan demikian jumlah inti CPU yang tersedia untuk I/O mungkin harus ditingkatkan. Dalam lingkungan hyper-threading, untuk kinerja terbaik, jumlah core CPU (yaitu, CPU real/fisik) harus cukup untuk memenuhi persyaratan CPU emulator aktif, sehingga menghindari thread beban kerja tinggi berbagi satu inti CPU fisik.
- Alokasi inti CPU untuk CPU yang ditiru dan inti CPU untuk pemrosesan I/O ditentukan oleh konfigurasi. Lihat Konfigurasi CPU di Panduan Pengguna Charon-SSP umum untuk informasi selengkapnya tentang ini dan alokasi default inti CPU untuk pemrosesan I/O.

> **Informasi umum yang penting**
>
> - Untuk memfasilitasi transfer cepat data emulator dari satu instance cloud ke yang lain, sangat disarankan untuk menyimpan semua data emulator yang relevan pada volume disk terpisah yang dapat dengan mudah terlepas dari instance lama dan dilampirkan ke instance baru.
> - Pastikan untuk mengukur instance Anda dengan benar dari awal (periksa persyaratan minimum di bawah). Lisensi Charon-SSP untuk Charon-SSP AL dibuat saat instance pertama kali diluncurkan. Mengubah nanti ke ukuran/jenis instance lain dan dengan demikian mengubah jumlah inti CPU akan membatalkan lisensi dan dengan demikian

mencegah instance Charon dimulai (instance baru diperlukan). Jika berencana untuk menggunakan instance Charon-SSP AL dalam mode AutoVE, pastikan untuk menyertakan informasi server AutoVE sebelum peluncuran pertama, jika tidak, server lisensi publik akan digunakan. Lisensi untuk Charon-SSP VE dibuat berdasarkan sidik jari yang diambil pada server lisensi. Jika server lisensi dijalankan langsung pada host emulator dan host emulator kemudian memerlukan, misalnya, perubahan jumlah inti CPU, lisensi akan dibatalkan (lisensi baru dan mungkin instance baru diperlukan).

Prasyarat instans

Persyaratan CPU umum: Charon-SSP mendukung prosesor arsitektur x86-64 modern berbasis instans Amazon EC2.

Persyaratan minimum untuk Charon-SSP:

- Jumlah minimum inti CPU sistem host:
  - Setidaknya satu inti CPU untuk sistem operasi host, ditambah:
  - Untuk setiap sistem SPARC yang ditiru:
    - Satu inti CPU untuk setiap CPU yang ditiru dari instance, ditambah:
    - Setidaknya satu inti CPU tambahan untuk pemrosesan I/O (setidaknya dua, jika optimasi JIT server digunakan). Lihat bagian Konfigurasi CPU yang disebutkan di atas untuk opsi konfigurasi. Secara default, Charon akan menetapkan 1/3 (min. 1; dibulatkan ke bawah) dari jumlah CPU yang terlihat oleh host Charon ke pemrosesan I/O.
- Persyaratan memori minimum:
  - RAM 4GB atau lebih untuk sistem operasi host Linux. Persyaratan sebenarnya mungkin lebih tinggi dan akan tergantung pada persyaratan layanan non-emulator yang berjalan di host Linux. Rekomendasi sebelumnya setidaknya 2GB RAM untuk host Linux masih akan berlaku untuk banyak sistem, tetapi meningkatnya persyaratan sistem operasi dan aplikasi Linux telah menyebabkan rekomendasi yang diperbarui untuk instalasi baru. Ditambah:
  - Untuk setiap sistem SPARC yang ditiru:
    - Memori yang dikonfigurasi dari instance yang ditiru, ditambah:
    - 2GB RAM (6GB RAM jika server JIT digunakan) untuk memungkinkan optimasi DIT, persyaratan emulator, buffer run-time, SMP dan emulasi grafis.

- Jika hyper-threading diaktifkan pada CPU x86-64 modern, dua thread dapat berjalan pada satu inti CPU fisik yang menyediakan dua CPU logis ke sistem operasi host. Jika memungkinkan, nonaktifkan hyper-threading pada host Charon-SSP. Namun, ini seringkali tidak mungkin dilakukan di lingkungan VMware dan cloud, atau tidak jelas apakah hyper-threading digunakan atau tidak. Opsi hyper-threading Charon-SSP memungkinkan Charon-SSP beradaptasi dengan lingkungan seperti itu. Lihat bagian Konfigurasi CPU di Panduan Pengguna Charon-SSP umum Anda yang disebutkan di atas untuk informasi konfigurasi terperinci. Catatan sewa: untuk kinerja terbaik, utas Charon-SSP tidak boleh berbagi inti CPU fisik - inti fisik yang cukup harus tersedia pada sistem host untuk memenuhi persyaratan emulator yang dikonfigurasi.
- Satu atau lebih antarmuka jaringan, tergantung pada kebutuhan pelanggan.
- Charon-SSP/4U+ dan Charon-SSP/4V+ harus berjalan pada perangkat keras fisik yang mendukung Intel VT-X/EPT atau AMD-V/NPT (instance baremetal) dan karenanya tidak dapat berjalan di semua lingkungan cloud. Silakan periksa dokumentasi penyedia cloud Anda untuk ketersediaan perangkat keras tersebut. Selain itu, perhatikan poin-poin berikut:
  - Charon-SSP/4U+ dan Charon-SSP/4V+hanya tersedia saat menggunakan kernel Linux yang didukung oleh Stromasys.
  - Jika Anda memerlukan jenis perangkat keras SPARC yang ditiru ini, hubungi Stromasys atau Stromasys VAR Anda untuk mendiskusikan kebutuhan Anda secara rinci.

Membuat dan mengonfigurasi instance AWS cloud untuk Charon (GUI Baru)

Bagian ini mencerminkan pada AWS Management Console musim semi 2022. Jika Anda masih menggunakan konsol lama, lihat Lampiran panduan Memulai AWS Charon-SSP.

Prasyarat umum

Deskripsi ini menunjukkan pengaturan dasar dari instance Linux di AWS. Itu tidak mencantumkan prasyarat khusus. Namun, tergantung pada kasus penggunaan Anda, pertimbangkan prasyarat berikut:

- Akun Amazon dan AWS Marketplace langganan
  - Untuk mengatur instance Linux AWS, Anda memerlukan AWS akun dengan akses administrator.

  - Identifikasi AWS Wilayah tempat Anda berencana untuk meluncurkan instans Anda. Pastikan bahwa AWS layanan yang Anda rencanakan untuk digunakan tersedia di Wilayah tersebut. Lihat AWS Layanan berdasarkan Wilayah.
  - Identifikasi VPC dan subnet tempat Anda berencana meluncurkan instans Anda.
  - Jika instans Anda memerlukan akses internet, pastikan bahwa tabel rute yang terkait dengan VPC Anda memiliki gateway internet. Jika instans Anda memerlukan akses VPN ke jaringan lokal Anda, pastikan gateway VPN tersedia. Konfigurasi yang tepat dari VPC Anda dan subnetnya akan tergantung pada desain jaringan dan persyaratan aplikasi Anda.
  - Untuk berlangganan AWS Marketplace layanan tertentu, pilih Langganan AWS Marketplace di bagian AWS Management Console lalu pilih Kelola langganan.
  - Cari layanan yang Anda rencanakan untuk digunakan dan berlangganan. Setelah berlangganan berhasil, Anda akan menemukan langganan di bagian Kelola langganan. Dari sana Anda dapat langsung meluncurkan instance baru.
- Prasyarat perangkat keras dan perangkat lunak instance akan berbeda tergantung pada penggunaan instance yang direncanakan:
  - Opsi 1: instance ini akan digunakan sebagai sistem host emulator Charon:
    - Lihat bagian prasyarat perangkat keras dan perangkat lunak dari Panduan Pengguna dan/atau panduan Memulai produk Charon Anda untuk menentukan prasyarat perangkat keras dan perangkat lunak yang tepat yang harus dipenuhi oleh instans Linux. Gambar yang Anda gunakan untuk meluncurkan instans dan jenis instans yang Anda pilih menentukan perangkat lunak dan perangkat keras instance cloud Anda.
    - Lisensi produk Charon diperlukan untuk menjalankan sistem warisan yang ditiru. Lihat informasi lisensi dalam dokumentasi produk Charon Anda, atau hubungi perwakilan Stromasys Anda atau Stromasys VAR untuk informasi tambahan.
  - Opsi 2: instance ini akan digunakan sebagai server lisensi VE khusus:
    - Lihat Panduan Server Lisensi VE untuk prasyarat terperinci.
- Sistem operasi warisan tertentu yang dapat berjalan dalam sistem yang ditiru yang disediakan oleh produk emulator Charon memerlukan lisensi dari vendor asli dari sistem operasi. Pengguna bertanggung jawab atas kewajiban perizinan apa pun yang terkait dengan sistem operasi lama dan harus memberikan lisensi yang sesuai.

Menggunakan AWS Management Console untuk meluncurkan instance baru

Untuk membuat instance baru

1. Masuk ke AWS Management Console dan buka konsol Amazon EC2 di https://console.aws.amazon.com/ec2/.
2. Pilih Luncurkan instans.
3. Masukkan nama untuk contoh.
4. Pilih AMI. AMI adalah gambar prepackaged yang digunakan untuk meluncurkan instance cloud. Ini termasuk sistem operasi dan perangkat lunak aplikasi yang berlaku. Pilihan AMI tergantung pada bagaimana Anda berencana untuk menggunakan instance:

   - Jika instance akan digunakan sebagai sistem host emulator Charon, beberapa pilihan AMI dimungkinkan:
     - Menginstal sistem host Charon dari gambar pasar Charon yang dikemas sebelumnya: mereka berisi sistem operasi yang mendasarinya dan perangkat lunak Charon yang sudah diinstal sebelumnya.
       - Tanyakan kepada perwakilan Stromasys Anda opsi mana yang saat ini tersedia di pasar penyedia cloud Anda.
       - Bergantung pada penyedia cloud dan paket rilis produk Stromasys, mungkin ada dua varian:
         - Lisensi otomatis (AL) untuk digunakan dengan server lisensi publik yang dioperasikan oleh Stromass, atau dengan server lisensi AutoVE pribadi yang dioperasikan pelanggan
         - Lingkungan virtual (VE) untuk digunakan dengan server lisensi VE pribadi yang dioperasikan pelanggan
     - Menginstal sistem host Charon menggunakan instalasi emulator Charon konvensional dengan paket RPM instalasi emulator Charon untuk Linux:
       - Pilih AMI Linux dari distribusi yang didukung oleh produk dan versi Charon pilihan Anda. Lihat panduan pengguna untuk produk Anda di situs dokumentasi Stromasys.
   - Jika instance akan digunakan sebagai server lisensi VE khusus, lihat Panduan Server Lisensi VE dalam Dokumentasi Lisensi untuk persyaratan instance Linux.

Setelah Anda memutuskan AMI mana yang diperlukan, pilih AMI produk Linux atau Charon yang cocok. Jika Anda tidak melihat AMI yang Anda butuhkan, pilih Jelajahi AMI lainnya. Pilih AMI Linux yang cocok dengan cara Anda berencana menggunakan instance. Ini bisa menjadi salah satu dari yang berikut:

- Gambar pasar Charon VE yang dikemas sebelumnya. Nama AMI akan menyertakan string “ve”.
- Gambar pasar Charon AL yang dikemas untuk Lisensi Otomatis atau AutoVE.
- Versi Linux yang didukung untuk instalasi produk RPM.
- Versi Linux yang didukung untuk server lisensi VE.

5. Pilih jenis instance. Amazon EC2 menawarkan tipe instans dengan berbagai kombinasi CPU, memori, penyimpanan, dan kapasitas jaringan. Pilih jenis instance yang sesuai dengan persyaratan produk Charon yang ingin Anda gunakan. Beberapa gambar pasar memiliki pilihan tipe instance yang terbatas.
6. Pilih key pair yang ada atau buat dan simpan yang baru. Jika Anda memilih key pair yang ada, pastikan Anda memiliki kunci pribadi yang cocok. Jika tidak, Anda tidak akan dapat terhubung ke instans Anda.

> **Note**
>
> Jika sistem manajemen Anda mendukungnya, untuk RHEL 9.x, Rocky Linux 9.x, dan Oracle Linux 9.x, gunakan kunci SSH tipe ECDSA atau ED25519. Jenis ini memungkinkan Anda untuk terhubung ke sistem Linux host Charon ini dengan menggunakan terowongan SSH tanpa perlu mengubah pengaturan kebijakan kripto default pada host Charon ke pengaturan yang kurang aman. Misalnya, ini penting untuk Manajer Charon-SSP. Lihat Menggunakan kebijakan kriptografi seluruh sistem dalam dokumentasi Red Hat.

7. Di bagian Pengaturan jaringan, pilih Edit. Pilih pengaturan yang sesuai dengan lingkungan Anda.
   - Tentukan VPC.
   - Tentukan subnet yang ada atau buat yang baru.
   - Aktifkan atau nonaktifkan penetapan otomatis alamat IP publik ke antarmuka utama. Penugasan otomatis hanya mungkin jika instance memiliki antarmuka jaringan tunggal.

- Tetapkan grup keamanan kustom yang sudah ada atau baru. Grup keamanan harus mengizinkan setidaknya SSH untuk mengakses instance. Port apa pun yang diperlukan oleh aplikasi yang Anda rencanakan untuk dijalankan pada instance juga harus diizinkan. Anda dapat memodifikasi grup keamanan kapan saja setelah membuat instance.

8. Di bagian Penyimpanan, untuk volume root (disk sistem), pilih ukuran yang sesuai untuk lingkungan Anda. Ukuran disk sistem minimum yang disarankan untuk sistem Linux adalah 30 GiB. Untuk menyediakan ruang bagi wadah disk virtual dan persyaratan penyimpanan lainnya, Anda dapat menambahkan lebih banyak penyimpanan sekarang atau setelah meluncurkan instance. Tetapi ukuran disk sistem harus mencakup persyaratan sistem Linux, termasuk aplikasi dan utilitas apa pun yang Anda rencanakan untuk diinstal.

> **Note**
>
> Kami menyarankan Anda membuat volume penyimpanan terpisah untuk data aplikasi Charon (misalnya, gambar disk). Jika perlu, nantinya Anda dapat memigrasikan volume tersebut ke instance lain.

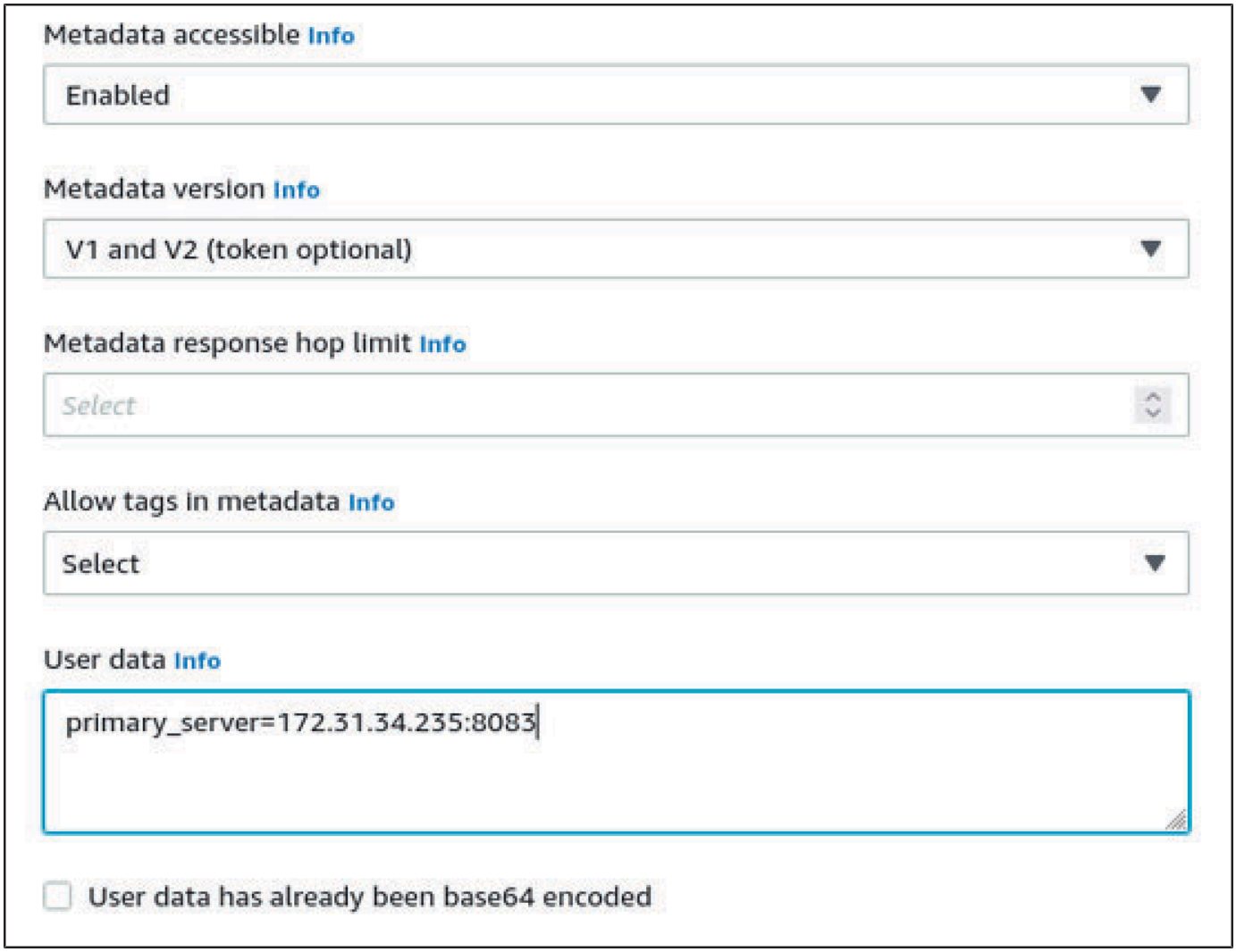
9. Perluas bagian Detail lanjutan, gulir ke bawah, dan pilih Tentukan opsi CPU. Tiga yang lebih mungkin berguna untuk lingkungan emulator Charon ditampilkan pada gambar berikut sebagai contoh.

10. Untuk sistem server lisensi VE dengan versi lebih awal dari 1.1.23, Anda harus menetapkan peran IAM yang diperlukan ke instance. Itu harus menjadi peran yang memungkinkan `ListUsers` tindakan. Untuk menetapkan peran, di bagian Detail lanjutan yang diperluas, pilih

peran di bawah profil instans IAM, atau pilih Buat profil IAM baru. Untuk informasi selengkapnya, lihat Peran IAM untuk Amazon EC2.

11. Jika instans Anda didasarkan pada AWS Marketplace gambar Charon AL dan Anda berencana untuk menggunakan server lisensi publik yang dioperasikan StromAsys, Anda harus menambahkan informasi yang sesuai ke konfigurasi instans sebelum meluncurkan instance.

    Masukkan informasi untuk server lisensi AutoVE seperti yang ditunjukkan pada gambar berikut.



    Berikut ini adalah opsi konfigurasi data pengguna yang valid:

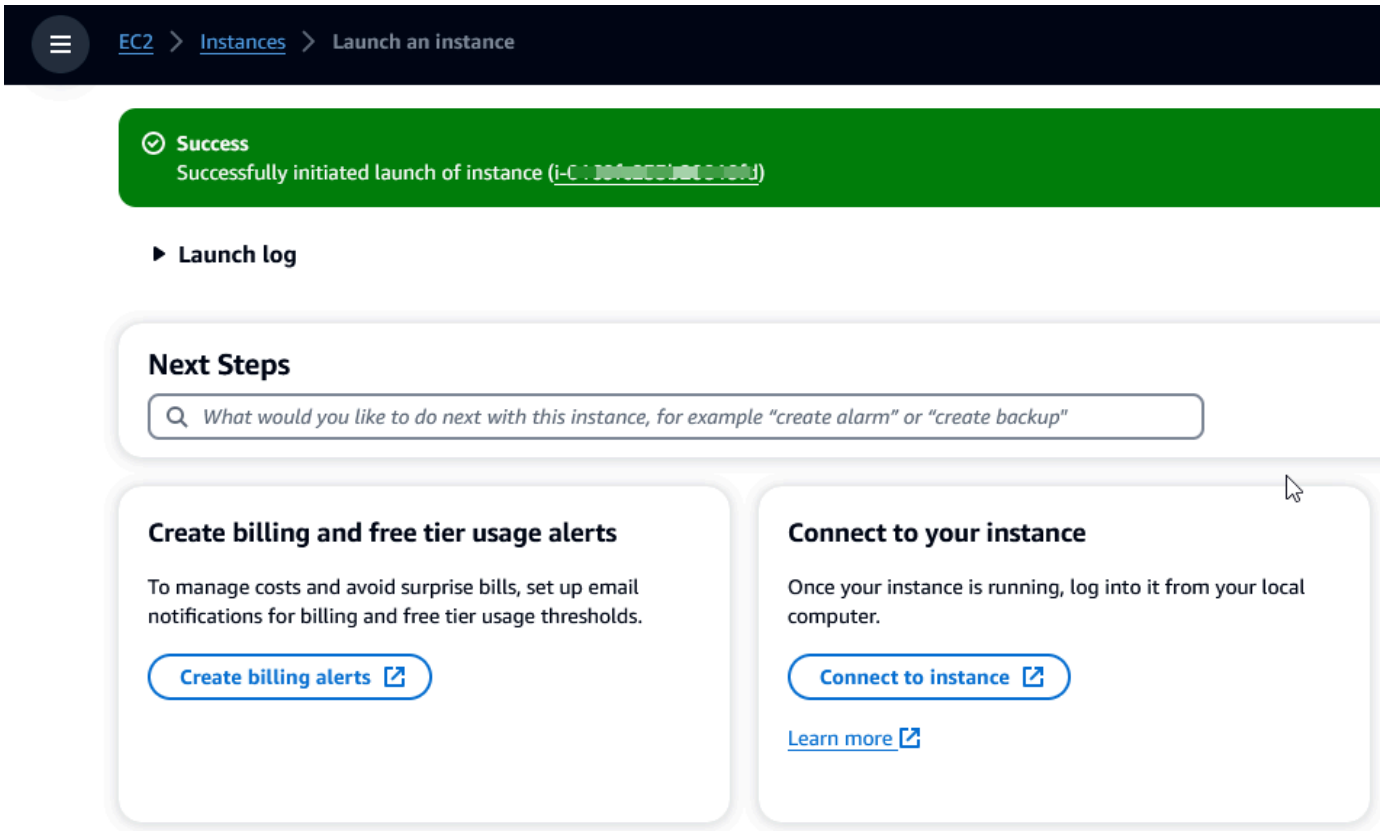
    - **`primary_server=`***`<ip-address>`*`[:`*`<port>`*`]`
    - **`backup_server=`***`<ip-address>`*`[:`*`<port>`*`]`

    Di mana

- <ip-address>singkatan dari alamat IP primer dan server cadangan sebagaimana berlaku.
- <port>singkatan dari port TCP non-default yang digunakan untuk berkomunikasi dengan server lisensi (default: TCP/8083).

> **Note**
>
> Setidaknya satu server lisensi harus dikonfigurasi pada peluncuran awal untuk mengaktifkan mode AutoVe. Jika tidak, instance akan mengikat ke salah satu server lisensi publik yang dioperasikan oleh Stromasys.

12. Di bagian Ringkasan, pilih Launch instance. Setelah beberapa saat, Anda akan melihat pesan sukses berikut:



13. Di sudut kanan bawah layar, pilih Lihat semua instance.
14. Untuk melihat detail instance Anda, pilih kotak centang di sebelah kiri baris yang mewakili instance dalam tabel Instances. Detail instans Anda akan muncul di bagian bawah layar. Untuk informasi tentang cara menyambung ke instans, lihat Connect di Panduan Pengguna Amazon EC2 untuk Instans Linux.

AWS Modernisasi Mainframe Replatforming dengan NTT DATA

AWS Modernisasi Mainframe menawarkan berbagai Amazon Machine Images (AMI). AMI ini memfasilitasi penyediaan cepat instans Amazon EC2, menciptakan lingkungan yang disesuaikan untuk rehosting dan replatforming aplikasi mainframe dengan menggunakan Data NTT. AWS Panduan ini memberikan langkah-langkah yang diperlukan untuk mengakses dan menggunakan AMI ini.

Prasyarat

- Pastikan Anda memiliki akses administrator ke AWS akun tempat Anda dapat membuat instans Amazon EC2.
- Pastikan layanan Modernisasi AWS Mainframe tersedia di Wilayah tempat Anda berencana membuat instans Amazon EC2. Lihat Daftar Layanan AWS yang Tersedia berdasarkan Wilayah.
- Identifikasi VPC Amazon tempat Anda ingin membuat instans Amazon EC2.

Berlangganan Gambar Mesin Amazon

Saat berlangganan produk AWS Marketplace, Anda dapat meluncurkan instans dari AMI produk.

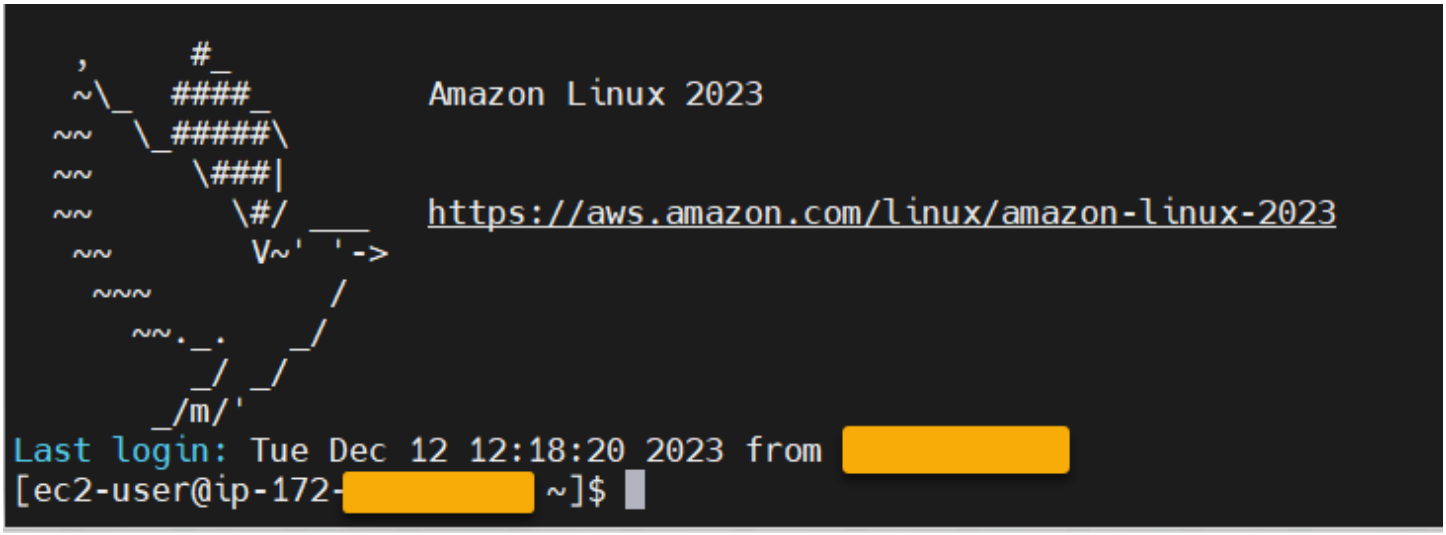
1. Masuk ke AWS Management Console dan buka AWS Marketplace konsol di https://console.aws.amazon.com/marketplace.
2. Pilih Kelola langganan.
3. Salin dan tempel tautan berikut ke bilah alamat browser: https://aws.amazon.com/marketplace/pp/prodview-eg227ymldsnx2
4. Pilih Lanjutkan Berlangganan.
5. Jika syarat dan ketentuan dapat diterima, pilih Terima Syarat. Langganan mungkin memakan waktu beberapa menit untuk diproses.
6. Tunggu pesan terima kasih muncul. Pesan ini menegaskan bahwa Anda telah berhasil berlangganan produk.
7. Di panel navigasi kiri, pilih Kelola langganan. Tampilan ini menunjukkan semua langganan Anda.

Luncurkan replatform Modernisasi AWS Mainframe dengan instans DATA NTT

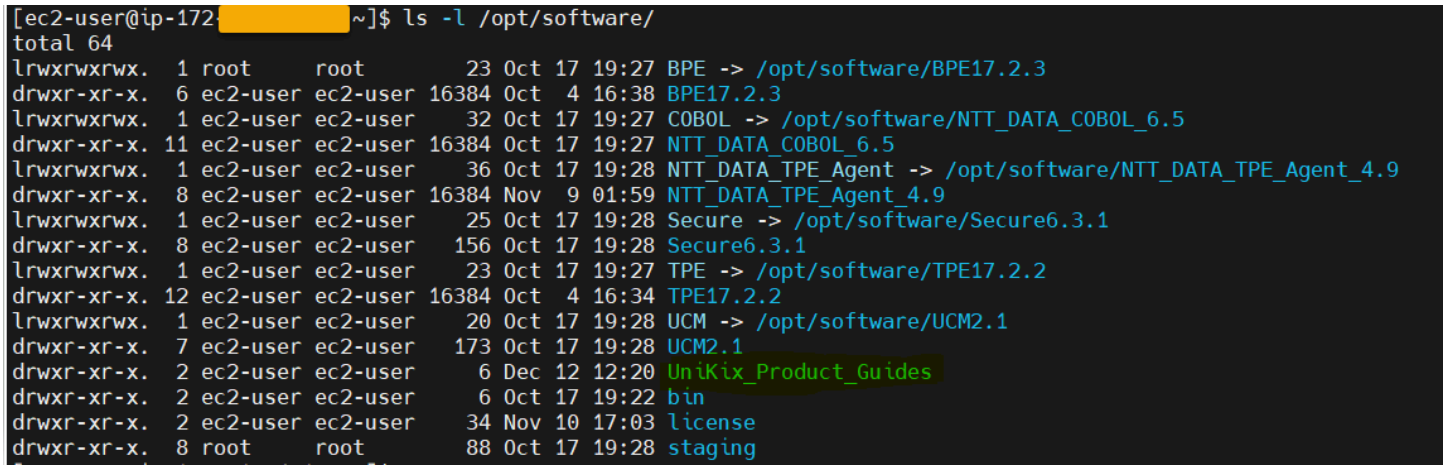
1. Buka AWS Marketplace konsol di https://console.aws.amazon.com/marketplace.
2. Di panel navigasi kiri, pilih Kelola langganan.
3. Temukan AMI yang ingin Anda luncurkan, dan pilih Luncurkan instance baru.
4. Di bawah Wilayah, pilih Wilayah yang diizinkan terdaftar.
5. Pilih Lanjutkan untuk meluncurkan melalui EC2. Tindakan ini membawa Anda ke konsol Amazon EC2.
6. Masukkan nama untuk server.
7. Pilih jenis instans yang sesuai dengan kinerja proyek dan persyaratan biaya Anda. Titik awal yang disarankan untuk ukuran misalnya adalah`c5.2xLarge`.
8. Pilih key pair yang ada atau buat dan simpan yang baru. Untuk informasi tentang pasangan kunci, lihat pasangan kunci Amazon EC2 dan instans Linux di Panduan Pengguna Amazon EC2 untuk Instans Linux.
9. Edit pengaturan jaringan dan pilih VPC yang terdaftar yang diizinkan dan subnet yang sesuai.
10. Pilih grup keamanan yang ada atau buat yang baru. Jika ini adalah instance Enterprise Server Amazon EC2, biasanya memungkinkan lalu lintas TCP ke port 86 dan 10086 untuk mengelola konfigurasi Micro Focus.
11. Konfigurasikan penyimpanan untuk instans Amazon EC2.
12. Tinjau ringkasan dan pilih Launch instance. Agar peluncuran berhasil, jenis instance harus valid. Jika peluncuran gagal, pilih Edit konfigurasi instance dan pilih jenis instans yang berbeda.
13. Setelah Anda melihat pesan sukses, pilih Connect to instance.
14. Buka konsol Amazon EC2 di https://console.aws.amazon.com/ec2/.
15. Di panel navigasi kiri, di bawah menu Instans, pilih Instans.
16. Di panel utama, periksa status instance Anda.

Memulai dengan Data NTT

Setelah Anda menyediakan instans Amazon EC2, SSH ke dalamnya dengan nama pengguna. `ec2-user` Layar akan terlihat seperti gambar berikut.



Di bawah `/opt/software/` folder, ada folder bernama`UniKix_Product_Guides`, seperti yang ditunjukkan pada gambar berikut.



`UniKix_Product_Guides`Folder tersebut menyertakan dokumentasi untuk komponen berikut yang diinstal pada instans Amazon EC2 ini:

- NTT DATA TPE
- NTT DATA BPE
- NTT DATA Perusahaan COBOL
- NTT DATA Aman UniKix
- Manajer Pusat DATA UniKix NTT

`software`Folder yang muncul pada gambar sebelumnya memiliki binari untuk komponen yang tercantum di atas.

Setelah berhasil memvalidasi instans Amazon EC2, mulailah AWS menggunakan Mainframe Modernization Replatform dengan NTT DATA dengan mengikuti dokumentasi Data NTT.

Aplikasi dalam Modernisasi AWS Mainframe

Jika Anda baru mengenal Modernisasi AWS Mainframe, lihat topik berikut untuk memulai:

- Apa itu Modernisasi AWS Mainframe?
- Menyiapkan AWS Modernisasi Mainframe
- Tutorial: Runtime Terkelola untuk AWS Blu Age
- Tutorial: Runtime terkelola untuk Micro Focus

Aplikasi dalam Modernisasi AWS Mainframe berisi beban kerja mainframe yang dimigrasi. Aplikasi ini analog dengan beban kerja pada mainframe dan dikaitkan dengan lingkungan runtime. Anda dapat menambahkan file batch dan kumpulan data ke aplikasi dan memantau aplikasi saat dijalankan. Anda membuat aplikasi Modernisasi AWS Mainframe untuk setiap beban kerja yang Anda migrasi. Saat Anda membuat aplikasi Modernisasi AWS Mainframe, Anda menentukan mesin tempat aplikasi berjalan saat Anda membuatnya. Pilih AWS Blu Age jika Anda menggunakan pola refactoring otomatis, dan pilih Micro Focus jika Anda menggunakan pola replatforming.

Topik

- Buat aplikasi Modernisasi AWS Mainframe
- Menyebarkan aplikasi Modernisasi AWS Mainframe
- Perbarui aplikasi Modernisasi AWS Mainframe
- Hapus aplikasi Modernisasi AWS Mainframe dari lingkungan
- Hapus aplikasi Modernisasi AWS Mainframe
- Kirim pekerjaan batch untuk aplikasi Modernisasi AWS Mainframe
- Impor set data untuk aplikasi Modernisasi AWS Mainframe
- Mengelola transaksi untuk aplikasi Modernisasi AWS Mainframe
- Membuat AWS sumber daya untuk aplikasi yang dimigrasi
- Konfigurasikan aplikasi yang dikelola
- AWS Referensi definisi aplikasi Modernisasi Mainframe
- AWSReferensi definisi kumpulan data modernisasi mainframe

Buat aplikasi Modernisasi AWS Mainframe

Gunakan konsol Modernisasi AWS Mainframe untuk membuat aplikasi Modernisasi AWS Mainframe.

Petunjuk ini berasumsi bahwa Anda telah menyelesaikan langkah-langkah di Menyiapkan AWS Modernisasi Mainframe.

Membuat aplikasi

Untuk membuat aplikasi

1. Buka konsol Modernisasi AWS Mainframe di https://console.aws.amazon.com/m2/.
2. Di Wilayah AWS pemilih, pilih Wilayah tempat Anda ingin membuat aplikasi.
3. Pada halaman Aplikasi, pilih Buat aplikasi.
4. Pada halaman Tentukan informasi dasar, di bagian Nama dan deskripsi, masukkan nama untuk aplikasi.
5. (Opsional) Di bidang Deskripsi aplikasi, masukkan deskripsi untuk aplikasi. Deskripsi ini dapat membantu Anda dan pengguna lain mengidentifikasi tujuan aplikasi.
6. Di bagian tipe Engine, pilih Blu Age untuk refactoring otomatis, atau Micro Focus untuk replatforming.
7. Di bagian kunci KMS, pilih Sesuaikan pengaturan enkripsi jika Anda ingin menggunakan AWS KMS kunci yang dikelola pelanggan. Untuk informasi selengkapnya, lihat Enkripsi data saat istirahat untuk layanan Modernisasi AWS Mainframe.

   > **Note**
   >
   > Secara default, Modernisasi AWS Mainframe mengenkripsi data Anda dengan AWS KMS kunci yang dimiliki dan dikelola oleh Modernisasi AWS Mainframe untuk Anda. Namun, Anda dapat memilih untuk menggunakan AWS KMS kunci yang dikelola pelanggan.

8. (Opsional) Pilih AWS KMS kunci berdasarkan nama atau Amazon Resource Name (ARN), atau pilih Buat AWS KMS kunci untuk pergi ke AWS KMS konsol dan membuat kunci baruAWS KMS.
9. (Opsional) Di bagian Tag, pilih Tambahkan tag baru untuk menambahkan satu atau beberapa tag aplikasi ke aplikasi Anda. Tag aplikasi adalah label atribut khusus yang membantu Anda mengatur dan mengelola AWS sumber daya Anda).

10. Pilih Berikutnya.
11. Di bagian Sumber Daya dan konfigurasi, gunakan editor sebaris untuk memasukkan definisi aplikasi. Atau, pilih Gunakan file JSON definisi aplikasi di bucket Amazon S3 dan berikan lokasi definisi aplikasi yang ingin Anda gunakan. Untuk informasi selengkapnya, lihat AWS Sampel definisi aplikasi Blu Age atau Definisi aplikasi Fokus Mikro.
12. Pilih Berikutnya.
13. Pada halaman Tinjau dan buat, tinjau informasi yang Anda masukkan, lalu pilih Buat aplikasi.

Menyebarkan aplikasi Modernisasi AWS Mainframe

Gunakan konsol Modernisasi AWS Mainframe untuk menyebarkan aplikasi Modernisasi Mainframe. AWS

Petunjuk ini berasumsi bahwa Anda telah menyelesaikan langkah-langkah di Menyiapkan AWS Modernisasi Mainframe.

Menyebarkan aplikasi

Untuk menjalankan aplikasi Modernisasi AWS Mainframe, Anda harus terlebih dahulu menerapkannya ke lingkungan runtime. Sebuah aplikasi dapat memiliki lebih dari satu versi. Setiap versi aplikasi memiliki definisi aplikasinya sendiri. Untuk menyebarkan aplikasi, Anda harus menentukan versi yang ingin Anda terapkan.

Anda hanya dapat menerapkan satu versi aplikasi tertentu pada satu waktu. Jika Anda menerapkan versi aplikasi, kemudian memutuskan untuk menerapkan versi yang berbeda, Anda harus terlebih dahulu menghentikan aplikasi jika sedang berjalan.

Untuk menyebarkan aplikasi

1. Buka konsol Modernisasi AWS Mainframe di https://console.aws.amazon.com/m2/.
2. Di Wilayah AWS pemilih, pilih Wilayah tempat Anda ingin membuat aplikasi.
3. Pada halaman Aplikasi, pilih aplikasi yang ingin Anda gunakan.
4. Pilih Menyebarkan aplikasi.
5. Di bagian Versi tersedia, pilih versi yang ingin Anda gunakan.
6. Di bagian Environments, pilih lingkungan runtime di mana Anda ingin aplikasi Anda berjalan.
7. Pilih Deploy.

Untuk menyebarkan versi berbeda dari aplikasi yang digunakan

1. Buka konsol Modernisasi AWS Mainframe di https://console.aws.amazon.com/m2/.
2. Di Wilayah AWS pemilih, pilih Wilayah tempat Anda ingin membuat aplikasi.
3. Pada halaman Aplikasi, pilih aplikasi yang ingin Anda gunakan.
4. Dari menu Actions, pilih Stop application.
5. Setelah aplikasi berhenti, pilih Menyebarkan aplikasi.
6. Di bagian Versi tersedia, pilih versi yang ingin Anda gunakan. Di bagian Lingkungan, lingkungan tempat aplikasi sudah digunakan telah dipilih sebelumnya.
7. Pilih Deploy.

Perbarui aplikasi Modernisasi AWS Mainframe

Gunakan konsol Modernisasi AWS Mainframe untuk memperbarui aplikasi Modernisasi AWS Mainframe.

Petunjuk ini berasumsi bahwa Anda telah menyelesaikan langkah-langkah di Menyiapkan AWS Modernisasi Mainframe.

Memperbarui sebuah aplikasi

Aplikasi Modernisasi AWS Mainframe dapat memiliki beberapa versi, masing-masing dengan definisi aplikasinya sendiri. Untuk memperbarui aplikasi, berikan definisi aplikasi baru. Ini menciptakan versi baru dari aplikasi.

Untuk memperbarui aplikasi

1. Buka konsol Modernisasi AWS Mainframe di https://console.aws.amazon.com/m2/.
2. Di Wilayah AWS pemilih, pilih Wilayah tempat aplikasi yang ingin Anda perbarui dibuat.
3. Pada halaman Aplikasi, pilih aplikasi yang ingin Anda perbarui.
4. Pada halaman detail aplikasi, di bagian Definisi saat ini, pilih Edit untuk memperbarui definisi aplikasi saat ini.
5. Pada halaman Perbarui aplikasi, gunakan editor sebaris untuk memperbarui definisi aplikasi saat ini.

Atau, pilih Gunakan file JSON definisi aplikasi di bucket Amazon S3 dan berikan lokasi definisi aplikasi yang ingin Anda gunakan. Untuk informasi selengkapnya, lihat AWS Sampel definisi aplikasi Blu Age atau Definisi aplikasi Fokus Mikro.

6. Setelah selesai memperbarui definisi aplikasi, pilih Perbarui.

> **Note**
>
> Setelah Anda memperbarui aplikasi, Anda harus menerapkannya lagi. Untuk informasi selengkapnya, lihat Menyebarkan aplikasi Modernisasi AWS Mainframe.

Hapus aplikasi Modernisasi AWS Mainframe dari lingkungan

Anda dapat menghapus aplikasi Modernisasi AWS Mainframe dari lingkungan menggunakan konsol Modernisasi AWS Mainframe.

Petunjuk ini berasumsi bahwa Anda telah menyelesaikan langkah-langkah di Menyiapkan AWS Modernisasi Mainframe.

Hapus aplikasi dari lingkungan

Jika Anda perlu menghapus aplikasi Modernisasi AWS Mainframe, dan sedang berjalan, pastikan Anda menghentikannya terlebih dahulu. Anda dapat melihat status aplikasi di halaman Aplikasi.

Untuk menghapus aplikasi dari lingkungan

1. Buka konsol Modernisasi AWS Mainframe di https://console.aws.amazon.com/m2/.
2. Di Wilayah AWS pemilih, pilih Wilayah tempat aplikasi yang ingin Anda hapus dari lingkungan dibuat.
3. Pada halaman Aplikasi, pilih aplikasi yang ingin Anda hapus dari lingkungan, lalu pilih Tindakan.
4. (Opsional) Jika status aplikasi adalah`Running`, pilih Stop aplikasi.
5. Pilih Hapus dari lingkungan.

   Proses penghapusan segera dimulai.

Hapus aplikasi Modernisasi AWS Mainframe

Gunakan konsol Modernisasi AWS Mainframe untuk menghapus aplikasi Modernisasi AWS Mainframe.

Petunjuk ini berasumsi bahwa Anda telah menyelesaikan langkah-langkah di Menyiapkan AWS Modernisasi Mainframe.

Menghapus sebuah aplikasi

Jika Anda perlu menghapus aplikasi Modernisasi AWS Mainframe, dan sedang berjalan, pastikan Anda menghentikannya terlebih dahulu. Anda dapat melihat status aplikasi di halaman Aplikasi.

Untuk menghapus aplikasi

1. Buka konsol Modernisasi AWS Mainframe di https://console.aws.amazon.com/m2/.
2. Di Wilayah AWS pemilih, pilih Wilayah tempat aplikasi yang ingin Anda hapus dibuat.
3. Pada halaman Aplikasi, pilih aplikasi yang ingin Anda hapus, lalu pilih Tindakan.
4. (Opsional) Jika status aplikasi adalah`Running`, pilih Stop aplikasi.
5. Pilih Hapus aplikasi.
6. Di jendela Hapus aplikasi, masukkan `delete` untuk mengonfirmasi bahwa Anda ingin menghapus aplikasi, lalu pilih Hapus.

Kirim pekerjaan batch untuk aplikasi Modernisasi AWS Mainframe

Dalam Modernisasi AWS Mainframe Anda dapat mengirimkan pekerjaan batch untuk aplikasi Anda. Anda dapat mengirimkan atau membatalkan pekerjaan batch dan meninjau detail tentang eksekusi pekerjaan batch. Setiap kali Anda mengirimkan pekerjaan batch, Modernisasi AWS Mainframe membuat eksekusi pekerjaan batch terpisah. Anda dapat memantau pelaksanaan pekerjaan ini. Anda dapat mencari pekerjaan batch berdasarkan nama dan menyediakan file JCL atau skrip ke pekerjaan batch.

> **Important**
>
> Jika Anda membatalkan pekerjaan batch, ini tidak menghapus pekerjaan. Ini membatalkan eksekusi tertentu dari pekerjaan batch. Catatan pekerjaan batch tetap tersedia untuk Anda lihat di detail untuk pelaksanaan pekerjaan batch.

Jika pekerjaan batch Anda memerlukan akses ke satu atau beberapa kumpulan data, gunakan konsol Modernisasi AWS Mainframe atau AWS Command Line Interface (AWS CLI) untuk mengimpor kumpulan data. Untuk informasi selengkapnya, lihat Impor set data untuk aplikasi Modernisasi AWS Mainframe.

Instruksi ini mengasumsikan bahwa Anda telah menyelesaikan langkah-langkah masuk Menyiapkan AWS Modernisasi Mainframe dan masukBuat aplikasi Modernisasi AWS Mainframe.

Kirim pekerjaan batch

Untuk mengirimkan pekerjaan batch

1. Buka konsol Modernisasi AWS Mainframe di https://console.aws.amazon.com/m2/.
2. Di Wilayah AWS pemilih, pilih Wilayah tempat aplikasi yang ingin Anda kirimkan pekerjaan batch dibuat.
3. Pada halaman Aplikasi, pilih aplikasi yang ingin Anda kirimkan pekerjaan batch.

   > **Note**
   >
   > Sebelum Anda dapat mengirimkan pekerjaan batch ke aplikasi, Anda harus menerapkan aplikasi dengan sukses.

4. Pada halaman detail aplikasi, pilih Pekerjaan Batch.
5. Pilih Submit job (Kirim tugas).
6. Di bagian Pilih skrip, pilih skrip. Anda dapat mencari skrip yang Anda inginkan berdasarkan nama.
7. Pilih Submit job (Kirim tugas).

Impor set data untuk aplikasi Modernisasi AWS Mainframe

Dengan Modernisasi AWS Mainframe Anda dapat mengimpor set data untuk digunakan dengan aplikasi Anda. Anda dapat menentukan kumpulan data dalam file JSON yang disimpan di bucket Amazon S3, atau Anda dapat menentukan nilai konfigurasi kumpulan data secara terpisah. Setelah mengimpor kumpulan data, Anda dapat meninjau detail tugas impor untuk mengonfirmasi bahwa kumpulan data yang Anda inginkan telah diimpor. Semua kumpulan data yang dikatalogkan untuk aplikasi dicantumkan bersama di konsol.

Gunakan konsol Modernisasi AWS Mainframe untuk mengimpor kumpulan data untuk aplikasi Modernisasi AWS Mainframe.

Instruksi ini mengasumsikan bahwa Anda telah menyelesaikan langkah-langkah masuk Menyiapkan AWS Modernisasi Mainframe dan masukBuat aplikasi Modernisasi AWS Mainframe.

Impor kumpulan data

Untuk mengimpor kumpulan data

1. Buka konsol Modernisasi AWS Mainframe di https://console.aws.amazon.com/m2/.
2. Di Wilayah AWS pemilih, pilih Wilayah tempat aplikasi yang ingin Anda impor set data dibuat.
3. Pada halaman Aplikasi, pilih aplikasi yang ingin Anda impor set data.
4. Pada halaman detail aplikasi, pilih Kumpulan data.
5. Pilih Impor.
6. Lakukan salah satu dari cara berikut:
   - Pilih Gunakan file JSON konfigurasi kumpulan data di bucket Amazon S3 dan berikan lokasi konfigurasi kumpulan data.
   - Pilih Tentukan nilai konfigurasi kumpulan data secara terpisah dengan konfigurasi terpandu. Lihat the section called “Referensi definisi kumpulan data” detail definisi spesifik.

     Masukkan nama, organisasi kumpulan data (VSAM, GDG, PO, PS), lokasi, dan lokasi Amazon S3 eksternal, dan pengaturan parameter untuk setiap nilai konfigurasi kumpulan data. Dalam konfigurasi terpandu, Anda juga dapat memilih Generate JSON untuk meninjau konfigurasi JSON dari input Anda.
7. Pilih Kirim.

Mengelola transaksi untuk aplikasi Modernisasi AWS Mainframe

Dengan Modernisasi AWS Mainframe Anda dapat menjalankan aplikasi, berdasarkan permintaan, pada saat yang sama dengan banyak pengguna lain yang mengirimkan permintaan untuk menjalankan aplikasi yang sama menggunakan file dan program yang sama. Transaksi tunggal terdiri dari satu atau lebih program aplikasi yang melakukan pemrosesan yang diperlukan.

Instruksi ini mengasumsikan bahwa Anda telah menyelesaikan langkah-langkah masuk Menyiapkan AWS Modernisasi Mainframe dan masukBuat aplikasi Modernisasi AWS Mainframe.

Mengelola transaksi untuk aplikasi

Anda dapat menampilkan dan mengedit transaksi untuk aplikasi.

Mengelola transaksi untuk aplikasi

1. Buka konsol Modernisasi AWS Mainframe di https://console.aws.amazon.com/m2/.
2. Di Wilayah AWS pemilih, pilih Wilayah AWS tempat aplikasi yang ingin Anda jalankan dibuat.
3. Pada halaman Aplikasi, pilih aplikasi yang ingin Anda kelola transaksi.
4. Pada tab Transaksi, di bawah Sumber daya transaksi, pilih bagaimana Anda ingin sumber daya Anda ditampilkan dari daftar tarik-turun. Anda dapat menampilkan sumber daya sesuai dengan sumber daya transaksi, grup, daftar, atau SIT.

   - Sumber daya transaksi memungkinkan Anda memilih jenis sumber daya sesuai dengan definisi file, definisi transaksi, definisi program, atau definisi antrian data sementara.

     > **Note**
     >
     > Transaksi Modernisasi AWS Mainframe Managed mendukung lebih banyak jenis sumber daya daripada ini, tetapi Anda tidak dapat langsung mengeditnya di sini. Sumber daya lain harus diedit secara eksternal dan Anda perlu membuat ulang aplikasi dengan entri sumber daya yang diperbarui.

   - Grup adalah kumpulan sumber daya transaksi. Anda juga dapat memilih grup yang ingin Anda kaitkan dengan sumber daya transaksi Anda.
   - Daftar diurutkan kumpulan grup. Anda dapat melihat semua sumber daya transaksi dan grup dalam tampilan daftar. Daftar startup menentukan sumber daya mana yang dimuat saat server diinisialisasi.
     - Dengan mesin refactor Blu Age, Anda menentukan daftar yang akan disertakan saat startup dan tidak ada batasan jumlah daftar.
     - Dengan mesin replatform Micro Focus, Anda dapat menentukan hingga empat daftar dalam satu SIT.
   - SIT (Tabel Inisialisasi Sistem) menampilkan semua konfigurasi transaksi yang tersedia. Anda dapat menemukan SIT sesuai dengan properti (nama, deskripsi, dan daftar startup). Anda juga dapat memilih daftar untuk dikaitkan dengan SIT pilihan Anda.

> **Note**
>
> SIT hanya berlaku untuk mesin replatform Micro Focus.

5. Pilih sumber daya transaksi untuk menampilkan semua informasi sumber daya. Anda juga dapat melihat semua atribut yang terkait dengan sumber daya transaksi Anda dan melihat atau mengedit atribut tambahan apa pun.
6. Jika Anda ingin mengedit informasi apa pun yang terkait dengan sumber daya transaksi Anda saat ini, pilih Edit.

    a. Pada halaman Edit, Anda dapat menambahkan atau mengubah deskripsi sumber daya.

        1. Untuk sumber daya transaksi yang ditampilkan dalam daftar, Anda juga dapat menambahkan, menghapus, atau menyusun ulang daftar sesuai keinginan.
        2. Untuk jenis sumber daya tertentu (definisi file, definisi transaksi, definisi program, dan definisi antrian data sementara), Anda dapat mengedit properti sumber daya individual. Properti ini bervariasi menurut jenis mesin dan sumber daya.

    b. Setelah melakukan perubahan yang diinginkan, pilih Simpan perubahan.

        Anda melihat pesan ketika sumber daya berhasil diperbarui.

Membuat AWS sumber daya untuk aplikasi yang dimigrasi

Untuk menjalankan aplikasi yang dimigrasiAWS, Anda harus membuat beberapa AWS sumber daya dengan yang lainLayanan AWS. Sumber daya yang harus Anda buat meliputi yang berikut:

- Bucket S3 untuk menyimpan kode aplikasi, konfigurasi, file data, dan artefak lain yang diperlukan.
- Basis data Amazon RDS atau Amazon Aurora untuk menyimpan data yang dibutuhkan aplikasi.
- AnAWS KMS key, yang diperlukan oleh AWS Secrets Manager untuk membuat dan menyimpan rahasia.
- Rahasia Secrets Manager untuk menyimpan kredensil database.

> **Note**
>
> Setiap aplikasi yang dimigrasi memerlukan kumpulan sumber daya ini sendiri. Ini adalah set minimum. Aplikasi Anda mungkin juga memerlukan sumber daya tambahan, seperti rahasia Amazon Cognito atau antrian MQ.

Izin yang diperlukan

Pastikan Anda memiliki izin berikut:

- `s3:CreateBucket`, `s3:PutObject`
- `rds:CreateDBInstance`
- `kms:CreateKey`
- `secretsmanager:CreateSecret`

Bucket Amazon S3

Aplikasi refactored dan replatformed memerlukan bucket S3 yang Anda konfigurasikan sebagai berikut:

```
bucket-name/root-folder-name/application-name
```

nama-ember

Nama apa pun dalam batasan penamaan Amazon S3. Kami menyarankan Anda memasukkan Wilayah AWS nama sebagai bagian dari nama bucket Anda. Pastikan Anda membuat bucket di Wilayah yang sama dengan tempat Anda berencana untuk menerapkan aplikasi yang dimigrasi.

root-folder-name

Nama diperlukan untuk memenuhi batasan dalam definisi aplikasi, yang Anda buat sebagai bagian dari aplikasi Modernisasi AWS Mainframe. Anda dapat menggunakan `root-folder-name` untuk membedakan antara berbagai versi aplikasi, misalnya, V1 dan V2.

nama aplikasi

Nama aplikasi Anda yang dimigrasi, misalnya, PlanetsDemo atau BankDemo.

Basis data

Baik aplikasi refactored maupun replatformed mungkin memerlukan database. Anda harus membuat, mengkonfigurasi, dan mengelola database sesuai dengan persyaratan khusus untuk setiap mesin runtime. AWS Modernisasi Mainframe mendukung enkripsi dalam perjalanan pada database ini. Jika Anda mengaktifkan SSL pada database Anda, pastikan bahwa Anda menentukan `sslMode` dalam rahasia database bersama dengan rincian koneksi database. Untuk informasi selengkapnya, lihat AWS Secrets Managerrahasia.

Jika Anda menggunakan pola refactoring AWS Blu Age, dan Anda memerlukan database, mesin runtime AWS Blu Age mengharapkan BluSam database Amazon Aurora PostgreSQL, yang harus Anda buat, konfigurasi, dan kelola. BluSam Database adalah opsional. Buat database ini hanya jika aplikasi Anda membutuhkannya. Untuk membuat database, ikuti langkah-langkah dalam Membuat cluster Amazon Aurora DB di Panduan Pengguna Amazon Aurora.

Jika Anda menggunakan pola replatforming Micro Focus, Anda dapat membuat Amazon RDS atau database Amazon Aurora PostgreSQL. Untuk membuat database, ikuti langkah-langkah dalam Membuat instans Amazon RDS DB di Panduan Pengguna Amazon RDS atau dalam Membuat klaster DB Amazon Aurora di Panduan Pengguna Amazon Aurora.

Untuk kedua mesin runtime, Anda harus menyimpan kredensi database dalam AWS Secrets Manager menggunakan AWS KMS key untuk mengenkripsi mereka.

Kunci AWS Key Management Service

Anda harus menyimpan kredensil untuk database aplikasi dengan aman. AWS Secrets Manager Untuk membuat rahasia di Secrets Manager, Anda harus membuat fileAWS KMS key. Untuk membuat kunci KMS, ikuti langkah-langkah dalam Membuat kunci di Panduan AWS Key Management Service Pengembang.

Setelah membuat kunci, Anda harus memperbarui kebijakan kunci untuk memberikan izin dekripsi Modernisasi AWS Mainframe. Tambahkan pernyataan kebijakan berikut:

```
{
   "Effect" : "Allow",
   "Principal" : {
   "Service" : "m2.amazonaws.com"
   },
   "Action" : "kms:Decrypt",
```

```
   "Resource" : "*"
   }
```

AWS Secrets Managerrahasia

Anda harus menyimpan kredensil untuk database aplikasi dengan aman. AWS Secrets Manager Untuk membuat rahasia ikuti langkah-langkah di Buat rahasia database di Panduan AWS Secrets Manager Pengguna.

AWSModernisasi Mainframe mendukung enkripsi dalam perjalanan pada database ini. Jika Anda mengaktifkan SSL pada database Anda, pastikan bahwa Anda menentukan `sslMode` dalam rahasia database bersama dengan rincian koneksi database. Anda dapat menentukan salah satu nilai berikut untuk`sslMode`:`verify-full`,`verify-ca`, atau`disable`.

Selama proses pembuatan kunci, pilih Izin sumber daya - opsional, lalu pilih Edit izin. Di editor kebijakan, tambahkan kebijakan berbasis sumber daya, seperti berikut ini, untuk mengambil konten bidang terenkripsi.

```
{
   "Effect" : "Allow",
   "Principal" : {
   "Service" : "m2.amazonaws.com"
   },
   "Action" : "secretsmanager:GetSecretValue",
   "Resource" : "*"
   }
```

Konfigurasikan aplikasi yang dikelola

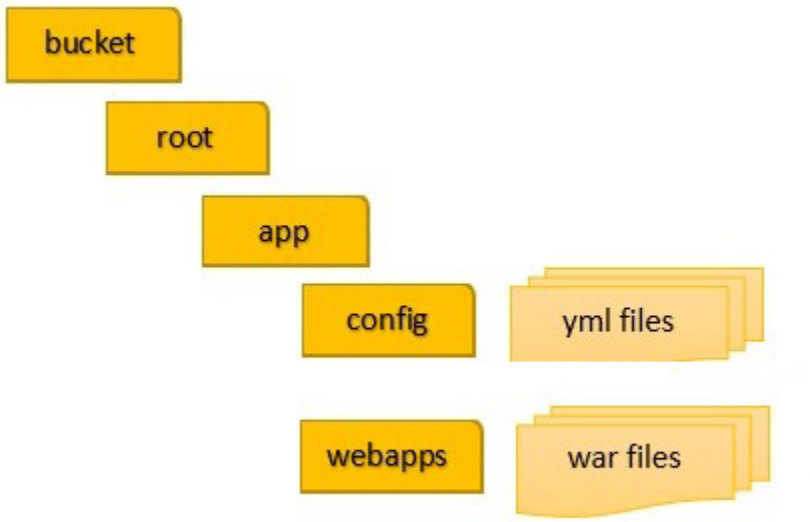
Anda dapat mengonfigurasi aplikasi Anda untuk menyertakan akses ke utilitas lama. Anda juga dapat menyesuaikan properti tambahan. Untuk memahami apa yang dapat Anda konfigurasikan dan di mana, ada baiknya untuk memahami struktur keseluruhan aplikasi modern AWS Blu Age.

Topik

- Struktur aplikasi terkelola AWS Blu Age
- Mengkonfigurasi akses ke utilitas untuk aplikasi terkelola
- Tambahkan properti konfigurasi untuk mesin AWS Blu Age

Struktur aplikasi terkelola AWS Blu Age

Jika Anda menggunakan pola refactoring AWS Blu Age, mesin runtime AWS Blu Age mengharapkan struktur berikut di dalam folder di bucket S3 Anda: `application-name`



config

Berisi file YAMM untuk proyek Anda. Ini adalah file YAMM khusus untuk aplikasi Anda, biasanya bernama sesuatu seperti `application-planetsdemo.yaml` dan bukan `application-main.yaml` file yang disediakan dan disiapkan oleh Modernisasi AWS Mainframe secara otomatis untuk Anda.

aplikasi web

Berisi `war` file untuk aplikasi Anda. File-file tersebut merupakan output dari proses modernisasi.

Aplikasi juga dapat memiliki folder opsional berikut:

jics/sql

Berisi `initJics.sql` skrip yang menginisialisasi database JICS untuk aplikasi Anda.

skrip

Berisi skrip aplikasi, yang juga dapat Anda suplai langsung di dalam `war` file.

sql

Berisi file SQL aplikasi, yang juga dapat Anda suplai langsung di dalam `war` file.

lnk

Berisi file LNK aplikasi, yang juga dapat Anda suplai langsung di dalam file. `war`

Mengelola konsumsi memori Java aplikasi

Untuk mengelola konsumsi memori Java untuk aplikasi, tambahkan file properti bernama `tomcat.properties` ke `application-name` folder. File ini dapat memiliki dua properti:`xms`, yang menentukan konsumsi memori Java minimum, dan`xmx`, yang menentukan konsumsi memori Java maksimum. Berikut ini adalah contoh isi `tomcat.properties` file yang valid.

```
xms=512M
xmx=1G
```

Nilai yang Anda tentukan untuk dua properti ini dapat berada di salah satu unit berikut:

- Bytes: jangan tentukan unit.
- Kilobyte: tambahkan K ke nilainya.
- Megabyte: tambahkan M ke nilainya.
- Gigabytes: tambahkan G ke nilainya.

Mengkonfigurasi akses ke utilitas untuk aplikasi terkelola

Saat Anda memfaktorkan ulang aplikasi mainframe dengan AWS Blu Age, Anda mungkin perlu memberikan dukungan untuk berbagai program utilitas platform lama, seperti IDCAMS, INFUTILB, SORT, dan sebagainya, jika aplikasi Anda bergantung padanya. AWS Refactoring Blu Age menyediakan akses ini dengan aplikasi web khusus yang digunakan bersama aplikasi modern. Aplikasi web ini membutuhkan file konfigurasi,`application-utility-pgm.yml`, yang harus Anda berikan. Jika Anda tidak menyediakan file konfigurasi ini, aplikasi web tidak dapat digunakan bersama aplikasi Anda dan tidak akan tersedia.

Topik

- Properti konfigurasi

Topik ini menjelaskan semua kemungkinan properti yang dapat Anda tentukan dalam file `application-utility-pgm.yml` konfigurasi, bersama dengan defaultnya. Topik menjelaskan

properti wajib dan opsional. Contoh berikut adalah file konfigurasi lengkap. Ini mencantumkan properti dalam urutan yang kami rekomendasikan. Anda dapat menggunakan contoh ini sebagai titik awal untuk file konfigurasi Anda sendiri.

```
# If the datasource support mode is not static-xa, spring JTA transactions
 autoconfiguration must be disabled
 spring.jta.enabled: false
 logging.config: 'classpath:logback-utility.xml'

 # Encoding
 encoding: cp1047

 # Encoding to be used by INFUTILB and DSNUTILB to generate and read SYSPUNCH files
 sysPunchEncoding: cp1047

 # Utility database access
 spring.aws.client.datasources.primary.secret: `arn:aws:secretsmanager:us-
west-2:111122223333:secret:business-FfmXLG`

 treatLargeNumberAsInteger: false

 # Zoned mode : valid values = EBCDIC_STRICT, EBCDIC_MODIFIED, AS400
 zonedMode: EBCDIC_STRICT

 jcl.type: mvs

 # Unload properties
 # For date/time: if use database configuration is enabled, formats are ignored
 # For nbi; use hexadecimal syntaxe to specify the byte value
 unload:
  sqlCodePointShift: 384
  nbi:
    whenNull: "6F"
    whenNotNull: "00"
  useDatabaseConfiguration: false
  format:
    date: MM/dd/yyyy
    time: HH.mm.ss
    timestamp: yyyy-MM-dd-HH.mm.ss.SSSSSS
  chunkSize:500
  fetchSize: 500
  varCharIsNull: false
  columnFiller: space
```

```
# Load properties
# Batch size for DSNUTILB Load Task
load:
 sqlCodePointShift: 384
 batchSize: 500
 format:
   localDate: dd.MM.yyyy|dd/MM/yyyy|yyyy-MM-dd
   dbDate: yyyy-MM-dd
   localTime: 'HH:mm:ss|HH.mm.ss'
   dbTime: 'HH:mm:ss'

table-mappings:
 TABLE_1_NAME : LEGACY_TABLE_1_NAME
 TABLE_2_NAME : LEGACY_TABLE_2_NAME
```

Properti konfigurasi

Anda dapat menentukan properti berikut dalam file konfigurasi Anda.

spring.jta.enabled

Opsional. Mengontrol apakah dukungan JTA diaktifkan. Untuk utilitas, kami sarankan Anda menetapkan nilai ini ke`false`.

```
spring.jta.enabled : false
```

logging.config

Wajib. Menentukan path ke file konfigurasi logger khusus. Kami menyarankan Anda menggunakan nama `logback-utility.xml` dan memberikan file ini sebagai bagian dari aplikasi modern. Cara umum untuk mengatur file-file ini adalah dengan meletakkan semua file konfigurasi logger di tempat yang sama, biasanya di subfolder `/config/logback` di mana `/config` folder yang berisi file konfigurasi yaml. Untuk informasi selengkapnya, lihat Bab 3: Konfigurasi logback dalam dokumentasi Logback.

```
logging.config : classpath:logback-utility.xml
```

encoding

Wajib. Menentukan set karakter yang digunakan program utilitas. Untuk kebanyakan kasus, ketika Anda bermigrasi dari platform z/OS, set karakter ini adalah varian EBCDIC, dan harus

cocok dengan set karakter yang dikonfigurasi untuk aplikasi modern. Default jika tidak disetel adalah`ASCII`.

```
encoding : cp1047
```

sysPunchEncoding

Opsional. Menentukan set karakter yang INFUTILB dan DSNUTILB gunakan untuk menghasilkan dan membaca file SYSPUNCH. Jika Anda menggunakan file SYSPUNCH dari platform lama apa adanya, nilai ini harus menjadi varian EBCDIC. Default jika tidak disetel adalah`ASCII`.

```
sysPunchEncoding : cp1047
```

Konfigurasi sumber data primer

Beberapa utilitas terkait database, seperti LOAD dan UNLOAD, memerlukan akses ke database target melalui sumber data. Seperti definisi sumber data lainnya dalam Modernisasi AWS Mainframe, akses ini mengharuskan Anda menggunakannya. AWS Secrets Manager Properti yang mengarah ke rahasia yang tepat di Secrets Manager adalah sebagai berikut:

spring.aws.client.datasources.primary.secret

Opsional. Menentukan rahasia di Secrets Manager yang berisi properti sumber data.

```
spring.aws.client.datasources.primary.secret: datasource-secret-ARN
```

spring.aws.client.datasources.primary.dbname

Opsional. Menentukan nama database target jika nama database tidak disediakan langsung dalam rahasia database, dengan `dbname` properti.

```
spring.aws.client.datasources.primary.dbname: target-database-name
```

treatLargeNumberAsInteger

Opsional. Terkait dengan spesifikasi mesin database Oracle dan penggunaan utilitas DSNTEP2/DSNTEP4. Jika Anda mengatur flag ini ke true, angka besar yang berasal dari database Oracle (NUMBER (38,0)) diperlakukan sebagai bilangan bulat. Default: `false`

```
treatLargeNumberAsInteger : false
```

ZonedMode

Opsional. Menetapkan mode yang dikategorikan untuk menyandikan atau memecahkan kode tipe data yang dikategorikan. Pengaturan ini memengaruhi cara digit tanda direpresentasikan. Nilai berikut ini valid:

- EBCDIC_STRICT: Default. Gunakan definisi yang ketat untuk penanganan tanda. Bergantung pada apakah set karakter EBCDIC atau ASCII, representasi digit tanda menggunakan karakter berikut:
  - Karakter EBCDIC yang sesuai dengan byte (`Cn+Dn`) untuk mewakili rentang digit positif dan negatif (`+0`ke`+9`, `-0` ke). `-9` Karakter ditampilkan sebagai`{`, `A` ke`I`,`}`, `J` ke `R`
  - Karakter ASCII yang sesuai dengan byte (`3n+7n`) untuk mewakili rentang digit positif dan negatif (`+0`ke`+9`, `-0` ke). `-9` Karakter ditampilkan `0` untuk9, p untuk y
- EBCDIC_MODIFIED: Gunakan definisi yang dimodifikasi untuk penanganan tanda. Untuk EBDIC dan ASCII, daftar karakter yang sama mewakili digit tanda, yaitu, untuk dipetakan ke+ `+0` ke `I` dan +9 dipetakan `A` ke `{` + `-0` ke`-9`. `}` `J` `R` \
- AS400: Gunakan untuk aset warisan modern yang berasal dari platform iSeries (AS400).

```
zonedMode:EBCDIC_STRICT
```

jcl.tipe

Opsional. Menunjukkan jenis warisan skrip JCL yang dimodernisasi. Utilitas IDCAMS menggunakan pengaturan ini untuk menyesuaikan kode pengembalian jika JCL pemanggilan bertipe. `vse` Nilai yang valid adalah sebagai berikut:

- `mvs`(Default)
- `vse`

```
jcl.type : mvs
```

Database Bongkar utilitas properti terkait

Gunakan properti ini untuk mengonfigurasi utilitas yang membongkar tabel database ke kumpulan data. Semua properti berikut adalah opsional.

Contoh ini menunjukkan semua properti bongkar yang mungkin.

```
# Unload properties
 # For date/time: if use database configuration is enabled, formats are ignored
 # For nbi; use hexadecimal syntaxe to specify the byte value
 unload:
 sqlCodePointShift: 0
 nbi:
 whenNull: "6F"
 whenNotNull: "00"
 useDatabaseConfiguration: false
 format:
 date: MM/dd/yyyy
 time: HH.mm.ss
 timestamp: yyyy-MM-dd-HH.mm.ss.SSSSSS
 chunkSize: 0
 fetchSize: 0
 varCharIsNull: false
 columnFiller: space
```

sqlCodePointPergeseran

Opsional. Menentukan nilai integer yang mewakili kode SQL point shift yang digunakan pada data. Default-nya adalah 0. Ini berarti bahwa tidak ada pergeseran titik kode yang dilakukan. Sejajarkan pengaturan ini dengan parameter shift titik kode SQL yang digunakan untuk aplikasi modern. Ketika pergeseran titik kode sedang digunakan, nilai yang paling umum untuk parameter ini adalah 384.

```
unload.sqlCodePointShift: 0
```

nbi

Opsional. Menentukan byte indikator null. Ini adalah nilai heksadesimal (sebagai string) yang ditambahkan di sebelah kanan nilai data. Dua nilai yang mungkin adalah sebagai berikut:

- WhenNull: Tambahkan nilai heksadesimal ketika nilai data adalah nol. Default-nya adalah 6 `. Terkadang nilai tinggi FF digunakan sebagai gantinya.

  ```
  unload.nbi.whenNull: "6F"
  ```

- whenNotNull: Tambahkan nilai heksadesimal ketika nilai data tidak null, tetapi kolomnya nullable. Default adalah `00` (nilai rendah).

```
unload.nbi.whenNotNull: "00"
```

useDatabaseConfiguration

Opsional. Menentukan tanggal dan waktu pemformatan properti. Ini digunakan untuk menangani objek tanggal/waktu dalam kueri UNLOAD. Default-nya adalah `false`.

- Jika diatur ke`true`, menggunakan`pgmDateFormat`,`pgmTimeFormat`, dan `pgmTimestampFormat` properti dari file konfigurasi utama (`application-main.yml`).
- Jika disetel ke`false`, gunakan properti pemformatan tanggal dan waktu berikut:
  - `unload.format.date`: Menentukan pola pemformatan tanggal. Default-nya adalah `MM/dd/yyyy`.
  - `unload.format.time`: Menentukan pola pemformatan waktu. Default-nya adalah `HH.mm.ss`.
  - `unload.format.timestamp`: Menentukan pola pemformatan timestamp. Default-nya adalah `yyyy-MM-dd-HH.mm.ss.SSSSSS`.

ChunkSize

Opsional. Menentukan ukuran potongan data yang digunakan untuk membuat set data SYSREC. Kumpulan data ini adalah target operasi pembongkaran kumpulan data, dengan operasi paralel. Defaultnya adalah `0` (tidak ada potongan).

```
unload.chunkSize:0
```

fetchSize

Opsional. Menentukan ukuran pengambilan data. Nilainya adalah jumlah catatan yang akan diambil pada satu waktu ketika strategi potongan data digunakan. Bawaan: `0`.

```
unload.fetchSize:0
```

varCharIsNull

Opsional. Menentukan bagaimana menangani kolom varchar non nullable dengan konten kosong. Default-nya adalah `false`.

Jika Anda menetapkan nilai ini`true`, konten kolom diperlakukan sebagai string kosong untuk tujuan pembongkaran, bukan string spasi tunggal. Tetapkan flag ini `true` untuk kasus mesin database Oracle saja.

```
unload.varCharIsNull: false
```

Pengisi kolom

Opsional. Menentukan nilai yang akan digunakan untuk padding kolom dibongkar di kolom varchar. Nilai yang mungkin adalah ruang atau nilai rendah. Default adalah spasi.

```
unload.columnFiller: space
```

Database memuat properti terkait

Gunakan properti ini untuk mengkonfigurasi utilitas yang memuat data set record ke dalam database target, misalnya, DSNUTILB. Semua properti berikut adalah opsional.

Contoh ini menunjukkan semua properti beban yang mungkin.

```
# Load properties
 # Batch size for DSNUTILB Load Task
 load:
 sqlCodePointShift: 384
 batchSize: 500
 format:
 localDate: dd.MM.yyyy|dd/MM/yyyy|yyyy-MM-dd
 dbDate: yyyy-MM-dd
 localTime: HH:mm:ss|HH.mm.ss
 dbTime: HH:mm:ss

 table-mappings:
 TABLE_1_NAME : LEGACY_TABLE_1_NAME
 TABLE_2_NAME : LEGACY_TABLE_2_NAME
```

sqlCodePointPergeseran

Opsional. Menentukan nilai integer yang mewakili kode SQL point shift yang digunakan pada data. Default ke 0, yang berarti bahwa aplikasi tidak membuat pergeseran titik kode. Sejajarkan pengaturan ini dengan parameter shift titik kode SQL yang digunakan untuk aplikasi modern. Saat Anda menggunakan pergeseran titik kode, nilai yang paling umum untuk parameter ini adalah 384.

```
load.sqlCodePointShift : 384
```

BatchSize

Opsional. Menentukan nilai integer yang mewakili jumlah catatan untuk memperlakukan sebelum Anda mengirim pernyataan batch aktual ke database. Default ke 0.

```
load.batchSize: 500
```

format

Opsional. Menentukan pola pemformatan tanggal dan waktu yang akan digunakan untuk konversi tanggal/waktu selama operasi beban database.

- `load.format.localDate`: Pola pemformatan tanggal lokal. Ini default ke. `dd.MM.yyyy|dd/MM/yyyy|yyyy-MM-dd`
- `load.format.dbDate`: Pola pemformatan tanggal database. Ini default ke. `yyyy-MM-dd`
- `load.format.localTime`: Pola pemformatan waktu lokal. Ini default ke. `HH:mm:ss|HH.mm.ss`
- `load.format.dbTime`: Pola pemformatan waktu database. Ini default ke. `HH:mm:ss`

pemetaan meja

Opsional. Menentukan koleksi pemetaan yang disediakan pelanggan antara warisan dan nama tabel modern. Program utilitas DSNUTILB mengkonsumsi pemetaan ini.

Tentukan nilai dalam format berikut: MODERN_TABLE_NAME: LEGACY_TABLE_NAME

Inilah contohnya:

```
table-mappings:
 TABLE_1_NAME : LEGACY_TABLE_1_NAME
 TABLE_2_NAME : LEGACY_TABLE_2_NAME
 ...
 TABLE_*N*_NAME : LEGACY_TABLE_*N*_NAME
```

> **Note**
>
> Ketika aplikasi utilitas dimulai, secara eksplisit mencatat semua pemetaan yang disediakan.

Tambahkan properti konfigurasi untuk mesin AWS Blu Age

Anda dapat menambahkan file di `config` folder untuk aplikasi refactored Anda yang akan memberi Anda akses ke fitur baru di mesin runtime AWS Blu Age. Anda harus memberi nama file ini`user-properties.yml`. File ini tidak menggantikan definisi aplikasi tetapi memperluasnya. Topik ini menjelaskan properti yang dapat Anda sertakan dalam `user-properties.yml` file.

> **Note**
>
> Anda tidak dapat mengubah beberapa parameter karena mereka dikendalikan baik oleh Modernisasi AWS Mainframe atau oleh definisi aplikasi. Semua parameter yang ditentukan dalam definisi aplikasi untuk aplikasi Anda memiliki prioritas di atas parameter yang Anda tentukan`user-properties.yml`.

Untuk informasi lebih lanjut tentang struktur aplikasi refactored, lihat. Struktur aplikasi terkelola AWS Blu Age

Diagram berikut menunjukkan di mana menemukan `user-properties.yml` file dalam struktur aplikasi sampel AWS Blu Age, PlanetsDemo.

```
PlanetsDemo-v1/
   ## config/
   #  ## application-PlanetsDemo.yml
   #  ## user-properties.yml
   ## jics/
   ## webapps/
```

Referensi properti konfigurasi

Ini adalah daftar properti yang tersedia. Semua parameter bersifat opsional.

Topik

- Properti aplikasi Gapwalk
- Properti batchscript Gapwalk
- Properti Gapwalk Blugen
- Properti perintah Gapwalk CL

- Properti pelari Gapwalk CL
- Properti Gapwalk JHDB
- Properti Gapwalk JICS
- Properti runtime Gapwalk
- Properti program utilitas Gapwalk
- Properti lainnya

Properti aplikasi Gapwalk

bluesam.fileLoading.CommitInterval

Opsional. Interval komit bluesam.

Tipe: angka

Default: 100000

kartu.pengkodean

Opsional. Pengkodean kartu: untuk digunakan dengan`useControlMVariable`.

Tipe: string

Standar: CP1145

checkinputfilesize

Opsional. Menentukan apakah akan merilis cek jika ukuran file adalah kelipatan dari ukuran rekaman.

Tipe: boolean

Bawaan: salah

database.cursor.overflow.allowed

Opsional. Menentukan apakah untuk memungkinkan cursor overflow. Setel `true` untuk melakukan panggilan berikutnya pada kursor apa pun posisinya. Atur `false` untuk memeriksa apakah kursor berada di posisi terakhir sebelum melakukan panggilan berikutnya pada kursor. Hanya aktifkan jika kursor DAPAT DIGULIR (SENSITIF atau TIDAK SENSITIF)

Tipe: boolean

Default: betul

DataSimplifier. onInvalidNumericData

Opsional. Bagaimana bereaksi saat mendekode data numerik yang tidak valid. Nilai yang diizinkan adalah`reject`,`toleratespaces`,`toleratespaceslowvalues`,`toleratemost`.

Tipe: string

Default: tolak

defaultKeepExistingBerkas

Opsional. Menentukan apakah akan mengatur nilai default dataset sebelumnya.

Tipe: boolean

Bawaan: salah

disposisi.checkexistence

Opsional. Menentukan apakah akan merilis cek pada keberadaan file untuk Dataset dengan DISP SHR atau OLD.

Tipe: boolean

Bawaan: salah

ExternalSort.Threshold

Opsional. Ambang batas pengurutan: kapan harus beralih ke pengurutan eksternal (gabungan).

Tipe: string

Default: null

`externalSort.threshold: 12MB`

ForceHR

Opsional. Menentukan apakah akan menggunakan Human Readable SYSPRINT, baik pada konsol atau file output.

Tipe: boolean

Bawaan: salah

ForcedDate

Opsional. Memaksa tanggal dan waktu tertentu dalam database. Gunakan hanya selama pengembangan dan pengujian.

Default: null

```
forcedDate: 2022-08-26T12:59:58.123456+01:57
```

FrozenDate

Opsional. Membekukan tanggal dan waktu dalam database. Gunakan hanya selama pengembangan dan pengujian.

Bawaan: salah

```
frozenDate: false
```

Ims.messages.ExtendedSize

Opsional. Menentukan apakah akan mengatur ExtendedSize pada pesan ims.

Tipe: boolean

Bawaan: salah

LockTimeout

Opsional. Batas waktu dalam milidetik transaksi ketika tidak dapat memperoleh kunci dalam jangka waktu tertentu.

Tipe: angka

Default: 500

MapTransfo.prefix

Opsional. Daftar awalan yang akan digunakan saat mengubah variabel ControlM. Masing-masing dipisahkan dengan koma.

Tipe: string

Default: &, @,%%

kueri. useConcatCondition

Opsional. Menentukan apakah kondisi kunci dibangun oleh penggabungan kunci atau tidak.

Tipe: boolean

Bawaan: salah

RollBackonrte

Opsional. Menentukan apakah untuk mengembalikan transaksi unit run implisit pada pengecualian runtime.

Tipe: boolean

Bawaan: salah

sctThreadLimit

Opsional. Batas utas untuk memicu skrip.

Tipe: angka

Default: 5

sqlCodePointPergeseran

Opsional. Pergeseran titik kode sql. Menggeser titik kode untuk karakter kontrol yang mungkin kita temui saat memigrasikan data rdbms lama ke rdbms modern. Misalnya, Anda dapat menentukan `384` untuk mencocokkan karakter `\u0180` unicode.

Tipe: angka

Default: 0

sqlIntegerOverflowDiizinkan

Opsional. Menentukan apakah untuk memungkinkan SQL integer overflow, yang berarti apakah menempatkan nilai yang lebih besar dalam variabel host diperbolehkan.

Tipe: boolean

Bawaan: salah

stepFailWhenAbend

Opsional. Menentukan apakah akan menaikkan abend jika langkah gagal atau menyelesaikan eksekusi.

Tipe: boolean

Default: betul

stopExecutionWhenProgNotFound

Opsional. Menentukan apakah akan berhenti berjalan jika program tidak ditemukan. Jika diatur `ketrue`, interupsi run jika program tidak ditemukan.

Tipe: boolean

Default: betul

uppercaseUserInput

Opsional. Menentukan apakah input pengguna harus dalam huruf besar.

Tipe: boolean

Default: betul

UseControlmVariable

Opsional. Menentukan apakah akan menggunakan spesifikasi Control-m untuk penggantian variabel.

Tipe: boolean

Bawaan: salah

Properti batchscript Gapwalk

encoding

Opsional. Pengkodean yang digunakan dalam proyek batchscript (bukan dengan groovy). Mengharapkan pengkodean yang valid`CP1047`,,`IBM930`,`ASCII`... `UTF-8`

Tipe: string

Standar: ASCII

Properti Gapwalk Blugen

managers.trancode

Opsional. Pemetaan trankode manajer dialog. Memungkinkan Anda memetakan kode transaksi JICS ke manajer dialog. Format yang diharapkan adalah`trancode1:dialogManager1;trancode2:dialogManager2;`.

Tipe: string

Default: null

```
managers.trancode: OR12:MYDIALOG1
```

Properti perintah Gapwalk CL

perintah-off

Opsional. Daftar perintah untuk dimatikan, dipisahkan dengan koma. Nilai yang diizinkan adalah `PGM_BASICRCVMSG,SNDRCVF,CHGVAR,QCLRDTAQ,,RTVJOBA,ADDLFM,ADDPFM,RCVF,OVRDBF,DLTOVR,CP`. Berguna saat Anda ingin menonaktifkan atau menimpa program yang ada. `PGM_BASIC`adalah program kecepatan khusus yang dirancang untuk tujuan debug.

Tipe: string

Default: null

spring.datasource.primary.jndi-name

Opsional. Sumber data utama Penamaan Java Dan Antarmuka Direktori (jndi).

Tipe: string

Default: jdbc/primer

ZonedMode

Opsional. Mode untuk pengkodean atau decoding tipe data yang dikategorikan. Nilai yang diizinkan adalah`EBCDIC_STRICT`/`EBCDIC_MODIFIED`/`AS400`.

Tipe: string

Standar: EBCDIC_STRICT

Properti pelari Gapwalk CL

cl.configuration.context.encoding

Opsional. Pengkodean file CL. Mengharapkan pengkodean yang validCP1047,,IBM930,ASCII... UTF-8

Tipe: string

Standar: CP297

Cl.ZonedMode

Opsional. Mode untuk perintah encoding atau decoding control language (CL). Nilai yang diizinkan adalahEBCDIC_STRICT/EBCDIC_MODIFIED/AS400.

Tipe: string

Standar: EBCDIC_STRICT

Properti Gapwalk JHDB

ims.program

Opsional. Daftar program IMS yang akan digunakan. Pisahkan setiap parameter dengan titik koma (;) dan setiap transaksi dengan koma (). , Sebagai contoh: ims.programs: PCP008,PCT008;PCP054,PCT054;PCP066,PCT066;PCP068,PCT068;

Tipe: string

Default: null

JHDB.CheckpointPath

Opsional. Jika jhdb.checkpointPersistence tidak none maka parameter ini memungkinkan Anda untuk mengatur jalur persistensi pos pemeriksaan (lokasi penyimpanan file checkpoint.dat), semua data pos pemeriksaan yang terkandung dalam registri diserialkan dan dicadangkan dalam file (checkpoint.dat) yang terletak di folder yang disediakan. Perhatikan bahwa hanya data pos

pemeriksaan (scriptID, StepID, posisi database dan area pos pemeriksaan) yang terkait dengan cadangan ini.

Tipe: string

Default: berkas:. /setup/

JHDB.CheckpointPersistence

Opsional. Mode persistensi pos pemeriksaan. Nilai yang diizinkan adalah`none`/`add`/`end`. Gunakan `add` untuk mempertahankan pos pemeriksaan saat yang baru dibuat dan ditambahkan ke registri. Gunakan `end` untuk mempertahankan pos pemeriksaan saat server shutdown. Nilai lain menonaktifkan persistensi. Perhatikan bahwa setiap kali pos pemeriksaan baru ditambahkan ke registri, semua pos pemeriksaan yang ada akan diserialisasi dan file akan dihapus. Ini bukan tambahan ke data yang ada dalam file. Jadi tergantung pada jumlah pos pemeriksaan, itu dapat memiliki beberapa efek pada kinerja.

Tipe: string

Default: tidak ada

jhdb.configuration.context.encoding

Opsional. Pengkodean JHDB (Java Hierarchical Database). Mengharapkan string pengkodean yang valid`CP1047`,,`IBM930`,`ASCII`... `UTF-8`

Tipe: string

Standar: CP297

jhdb. identificationCardData

Opsional. Digunakan untuk hardcode beberapa “data kartu identifikasi operator” ke bidang MID yang ditunjuk oleh parameter CARD.

Tipe: string

Default: “”

jhdb.lterm

Opsional. Memungkinkan Anda untuk memaksa ID terminal logis umum dalam kasus emulasi IMS. Jika tidak disetel maka sessionId digunakan.

Tipe: string

Default: null

jhdb.metadata.extrapath

Parameter konfigurasi yang menentukan folder root khusus runtime tambahan untuk folder psbs dan dbds.

Tipe: string

Default: berkas:. /setup/

> **Note**
>
> Saat ini, untuk batasan penerapan, Anda harus menyalin direktori dbds dan psbs Anda di direktori konfigurasi aplikasi Anda atau di subdirektori direktori konfigurasi: misalnya, config/setup
>
> ```
> config
> |- setup
>    |- dbds
>    |- psbs
> ```
>
> dan diatur dalam aplikasi-jhdb.yl
> `jhdb.metadata.extrapath: file: ./config/setup/`

jhdb.navigation.cachenexts

Opsional. Durasi cache (dalam milidetik) yang digunakan dalam navigasi hierarkis untuk RDBMS.

Tipe: angka

Default: 5000

jhdb.query. limitJoinUsage

Opsional. Menentukan apakah akan menggunakan batas bergabung parameter penggunaan pada grafik RDBMS.

Tipe: boolean

Default: betul

jhdb. use-db-prefix

Opsional. Menentukan apakah untuk mengaktifkan awalan database dalam navigasi hierarkis untuk RDBMS.

Tipe: boolean

Default: betul

Properti Gapwalk JICS

jics.data. dataJsonInitLokasi

Opsional. Lokasi file json disiapkan oleh Analyzer dari parsing CSD, dan digunakan untuk menginisialisasi database jics,

Tipe: string

Default: “”

jics.db. dataScriptLocation

Opsional. Lokasi skrip initJics.sql, disiapkan oleh Analyzer dari penguraian ekspor CSD dari mainframe.

Tipe: string

Default: “”

jics.db. dataTestQueryLokasi

Opsional. Lokasi skrip sql yang berisi kueri sql tunggal yang diharapkan mengembalikan hitungan objek (misalnya: menghitung jumlah catatan dalam tabel program jics). Jika hitungan sama dengan 0, database akan dimuat menggunakan `jics.db.dataScriptLocation` skrip, jika tidak beban database akan dilewati.

Tipe: string

Default: “”

jics.db. ddlScriptLocation

Opsional. Lokasi skrip ddl Jics. Memungkinkan Anda untuk memulai skema database jics menggunakan skrip.sql.

Tipe: string

Default: “”

```
jics.db.ddlScriptLocation: ./jics/sql/jics.sql
```

jics.db. schemaTestQueryLokasi

Opsional. Lokasi file sql yang harus berisi kueri unik yang mengembalikan jumlah objek dalam skema jics (jika ada).

Tipe: string

Default: “”

kehebohan. runUnitLauncherPool.enable

Opsional. Menentukan apakah akan mengaktifkan run unit launcher pool di JICS.

Tipe: boolean

Bawaan: salah

kehebohan. runUnitLauncherPool.size

Opsional. Ukuran kolam peluncur unit run di JICS.

Tipe: angka

Default: 20

kehebohan. runUnitLauncherPool.validationInterval

Opsional: Interval validasi kumpulan peluncur unit run di JICS, dinyatakan dalam milidetik.

Tipe: angka

Default: 1000

jics.queues.sqs.region

Opsional. Wilayah AWS Untuk Amazon SQS, digunakan dalam JICS. Disarankan untuk mengatur wilayah yang sama dari aplikasi yang digunakan untuk kinerja, tetapi itu tidak wajib.

Tipe: string

Default: eu-west-1

jics.xa.agent.timeout

Opsional. Mendefinisikan durasi maksimum untuk agen xa yang bertanggung jawab untuk mengelola transaksi terdistribusi, untuk menyelesaikan operasinya.

Tipe: angka

Default: null

mq.queues.sqs.region

Opsional. Wilayah AWS Untuk layanan Amazon SQS MQ.

Tipe: string

Default: eu-barat-3

TaskExecutor. allowCoreThreadTimeOut

Opsional. Menentukan apakah untuk memungkinkan thread inti untuk time out di JCIS. Hal ini memungkinkan pertumbuhan dan penyusutan dinamis bahkan dalam kombinasi dengan antrian bukan nol (karena ukuran kolam maksimal hanya akan bertambah setelah antrian penuh).

Tipe: boolean

Bawaan: salah

TaskExecutor. corePoolSize

Opsional. Ketika transaksi di terminal dimulai melalui skrip groovy, thread baru dibuat. Gunakan parameter ini untuk mengatur ukuran kolam inti.

Tipe: angka

Default: 5

TaskExecutor. maxPoolSize

Opsional. Ketika transaksi di terminal dimulai melalui skrip groovy, thead baru dibuat. Gunakan parameter ini untuk mengatur ukuran kolam maks (jumlah maksimum thread paralel).

Tipe: angka

Default: 10

TaskExecutor.QueueCapacity

Opsional. Ketika transaksi di terminal dimulai melalui skrip groovy, thead baru dibuat. Gunakan parameter ini untuk mengatur ukuran antrian. (= jumlah maksimum transaksi yang tertunda saat `taskExecutor.maxPoolSize` tercapai)

Tipe: angka

Default: 50

Properti runtime Gapwalk

Cachemetadata

Opsional. Menentukan apakah untuk cache metadata database.

Tipe: boolean

Default: betul

check-groovy-file

Opsional. Menentukan apakah untuk memeriksa konten file asyik sebelum mendaftar.

Tipe: boolean

Default: betul

DatabaseStatistik

Opsional. Menentukan apakah akan memungkinkan pembangun SQL untuk mengumpulkan dan menampilkan informasi statistik.

Tipe: boolean

Bawaan: salah

dateTimeFormat

Opsional. Ini dateTimeFormat menjelaskan cara menumpahkan tipe stempel waktu tanggal waktu database ke entitas penyederhana data. Nilai yang diizinkan adalah`ISO`/`EUR`/`USA`/`LOCAL`

Tipe: string

Default: ISO

dbDateFormat

Opsional. Format tanggal target database.

Tipe: string

Default: YYYY-MM-DD

dbTimeFormat

Opsional. Format waktu target database.

Tipe: string

Default: HH: mm: SS

dbTimestampFormat

Opsional. Database menargetkan format timestamp.

Tipe: string

Default: YYYY-MM-DD HH: mm: SS.SSSSSS

fetchSize

Opsional. Nilai fetchSize untuk kursor. Gunakan saat mengambil data menggunakan potongan dengan memuat/membongkar utilitas.

Tipe: angka

Default: 10

ForceDisablesQL TrimStringType

Opsional. Menentukan apakah untuk menonaktifkan trim dari semua parameter string sql.

Tipe: boolean

Bawaan: salah

localDateFormat

Opsional. Daftar format tanggal lokal. Pisahkan setiap format dengan|.

Tipe: string

localTimeFormat

Opsional. Daftar format waktu lokal. Pisahkan setiap format dengan|.

Tipe: string

localTimestampFormat

Opsional. Daftar format stempel waktu lokal. Pisahkan setiap format dengan|.

Tipe: string

Default:

pgmDateFormat

Opsional. Format waktu tanggal yang digunakan dalam program.

Tipe: string

Default: YYYY-MM-DD

pgmTimeFormat

Opsional. Format waktu yang digunakan untuk eksekusi pgm (program).

Tipe: string

Default: HH.mm.ss

pgmTimestampFormat

Opsional. Format stempel waktu.

Tipe: string

Standar: YYYY-MM-DD-HH.mm.ss.ssssss

Properti program utilitas Gapwalk

jcl.tipe

Opsional. `.jcl`jenis file. Nilai yang diizinkan adalah`jcl`/`vse`. Perintah IDCAMS utilitas PRINT/REPRO mengembalikan 4 jika file kosong untuk non-vse jcl.

Tipe: string

Default: mvs

listcat.variablelengthpreprocessor.enabled

Opsional. Menentukan apakah akan mengaktifkan preprocessor panjang variabel untuk perintah LISTCAT.

Tipe: boolean

Bawaan: salah

listcat.variablelengthpreprocessor.type

Opsional. Jenis objek yang terkandung dalam file listcat, jika Anda mengaktifkan`listcat.variablelengthpreprocessor.enabled`. Nilai yang diizinkan adalah`rdw/bdw`.

Tipe: string

Default: rdw

Muat.batchSize

Opsional. Ukuran batch utilitas beban.

Tipe: angka

Default: 0

Load.format.dbdate

Opsional. Format database utilitas beban untuk digunakan.

Tipe: string

Default: YYYY-MM-DD

Load.format.dbtime

Opsional. Basis data utilitas beban waktu untuk digunakan.

Tipe: string

Default: HH: mm: SS

Load.format.localdate

Opsional. Format tanggal lokal utilitas muat untuk digunakan.

Tipe: string

Default: dd.mm.yyyy|dd/mm/yyyy|yyyy-mm-dd

Load.format.localTime

Opsional. Format waktu lokal utilitas beban untuk digunakan.

Tipe: string

Default: HH: mm: SS | HH.mm.ss

beban. sqlCodePointPergeseran

Opsional. Kode SQL pointshift untuk utilitas beban. Menjalankan proses pergeseran karakter. Diperlukan ketika database target Anda dari DB2 adalah Postgresql.

Tipe: angka

Default: 0

sysPunchEncoding

Opsional. Set karakter pengkodean syspunch. Nilai yang didukung adalah`Cp1047`/`ASCII`.

Tipe: string

Standar: ASCII

treatLargeNumberAsInteger

Opsional. Menentukan apakah untuk memperlakukan jumlah besar sebagai`Integer`. Mereka diperlakukan sebagai `BigDecimal` default.

Tipe: boolean

Bawaan: salah

Unload.chunkSize

Opsional. Ukuran potongan digunakan untuk utilitas bongkar.

Tipe: angka

Default: 0

Unload.columnfiller

Opsional. Pengisi kolom utilitas bongkar muat.

Tipe: string

Default: ruang

Unload.fetchSize

Opsional. Memungkinkan Anda menyetel ukuran pengambilan saat menangani kursor di utilitas bongkar muat.

Tipe: angka

Default: 0

unload.format.date

Opsional. Jika `unload.useDatabaseConfiguration` diaktifkan, format tanggal yang akan digunakan dalam utilitas bongkar. Untuk informasi selengkapnya, lihat unload.format.date.

Tipe: string

Default: mm/dd/YYYY

unload.format.time

Opsional. Jika `unload.useDatabaseConfiguration` diaktifkan, format waktu untuk digunakan dalam utilitas bongkar.

Tipe: string

Default: HH.mm.ss

unload.format.timestamp

Opsional. Jika `unload.useDatabaseConfiguration` diaktifkan, format stempel waktu untuk digunakan dalam utilitas bongkar.

Tipe: string

Standar: YYYY-MM-DD-HH.mm.ss.ssssss

bongkar.nbi. whenNotNull

Opsional. Nilai Null Byte Indicator (nbi) untuk ditambahkan ketika nilai dari database tidak null.

Jenis: heksadesimal

Default: 00

Unload.nbi.WhenNull

Opsional. Nilai Null Byte Indicator (nbi) untuk ditambahkan ketika nilai dari database adalah null.

Jenis: heksadesimal

Default: 6F

bongkar.nbi. writeNullIndicator

Opsional. Menentukan apakah akan menulis indikator null dalam file output bongkar.

Tipe: boolean

Bawaan: salah

membongkar. sqlCodePointPergeseran

Opsional. Kode SQL pointshift untuk utilitas bongkar. Menjalankan proses pergeseran karakter. Diperlukan ketika database target Anda dari DB2 adalah Postgresql.

Tipe: angka

Default: 0

membongkar. useDatabaseConfiguration

Opsional. Menentukan apakah akan menggunakan konfigurasi tanggal atau waktu dari application-main.ymldalam utilitas bongkar.

Tipe: boolean

Bawaan: salah

membongkar. varCharIsNull

Opsional. Gunakan parameter ini dalam program INFTILB, jika diatur untuk `true` maka semua bidang tidak nullable dengan nilai kosong (spasi) mengembalikan string kosong.

Tipe: boolean

Bawaan: salah

Properti lainnya

qtemp.cleanup.threshold.hours

Opsional. Untuk menentukan kapan `qtemp.dblog` diaktifkan. Masa pakai partisi db (dalam jam).

Tipe: angka

Default: 0

qtemp.dblog

Opsional. Apakah akan mengaktifkan pencatatan Database QTEMP.

Tipe: boolean

Bawaan: salah

qtemp.uuid.length

Opsional. Panjang id unik QTEMP.

Tipe: angka

Default: 9

quartz.scheduler. stand-by-if-error

Opsional. Menentukan apakah akan memicu eksekusi pekerjaan jika penjadwal pekerjaan dalam modus siaga. Jika benar, Ketika diaktifkan eksekusi pekerjaan tidak dipicu.

Tipe: boolean

Bawaan: salah

warmUpCache

Opsional. Menentukan apakah untuk memuat semua data tabel datacom ke cache pemanasan di server mulai.

Tipe: boolean

Bawaan: salah

AWS Referensi definisi aplikasi Modernisasi Mainframe

Dalam Modernisasi AWS Mainframe, Anda mengonfigurasi aplikasi mainframe yang dimigrasi dalam file JSON definisi aplikasi, yang khusus untuk mesin runtime yang Anda pilih. Definisi aplikasi berisi informasi umum dan informasi khusus mesin. Topik ini menjelaskan definisi aplikasi AWS Blu Age dan Micro Focus dan mengidentifikasi semua elemen yang diperlukan dan opsional.

Topik

Bagian header umum

Setiap definisi aplikasi dimulai dengan informasi umum tentang versi template dan lokasi sumber. Versi definisi aplikasi saat ini adalah 2.0. Meskipun versi 1 masih berfungsi, ia berada di jalur penghentian. Kami menyarankan Anda menggunakan versi 2 saat Anda membuat atau memperbarui aplikasi.

Gunakan struktur berikut untuk menentukan versi template dan lokasi sumber.

```
"template-version": "2.0",
    "source-locations": [
        {
            "source-id": "s3-source",
            "source-type": "s3",
            "properties": {
                "s3-bucket": "mainframe-deployment-bucket-aaa",
                "s3-key-prefix": "v1"
            }
        }
```

```
]
```

> **Note**
>
> Anda dapat menggunakan sintaks berikut jika Anda ingin memasukkan S3 ARN sebagai s3-bucket:
>
> ```
> "template-version": "2.0",
>  "source-locations": [
>      {
>          "source-id": "s3-source",
>          "source-type": "s3",
>          "properties": {
>              "s3-bucket": "arn:aws:s3:::mainframe-deployment-bucket-aaa",
>              "s3-key-prefix": "v1"
>          }
>      }
>  ]
> ```

Templat-versi

Wajib. Menentukan versi file definisi aplikasi. Jangan ubah nilai ini. Saat ini, satu-satunya nilai yang diizinkan adalah 2.0. Tentukan `template-version` dengan string.

sumber-lokasi

Menentukan lokasi file dan sumber daya lain yang dibutuhkan aplikasi selama runtime.

properti

Memberikan rincian lokasi sumber. Setiap properti ditentukan dengan string.

- `s3-bucket`- Diperlukan. Menentukan nama bucket Amazon S3 tempat file disimpan.
- `s3-key-prefix`- Diperlukan. Menentukan nama folder di bucket Amazon S3 tempat file disimpan.

> **Note**
>
> Pastikan Anda menentukan nama bucket Amazon S3, bukan bucket ARN. Jangan tentukan jalur absolut ke sumber daya di bucket.

Ikhtisar bagian definisi

Menentukan definisi sumber daya dari layanan, pengaturan, data, dan sumber daya khas lainnya yang perlu dijalankan aplikasi. Saat Anda memperbarui definisi aplikasi, Modernisasi AWS Mainframe mendeteksi perubahan dengan membandingkan `source-locations` dan `definition` daftar dari versi file JSON definisi aplikasi sebelumnya dan saat ini.

Bagian definisi khusus mesin dan dapat berubah. Bagian berikut menunjukkan contoh definisi aplikasi khusus mesin untuk kedua mesin.

AWS Sampel definisi aplikasi Blu Age

```
{
 "template-version": "2.0",
 "source-locations": [
        {
            "source-id": "s3-source",
            "source-type": "s3",
            "properties": {
                "s3-bucket": "mainframe-deployment-bucket-aaa",
                "s3-key-prefix": "v1"
            }
        }
    ],
    "definition" : {
        "listeners": [{
            "port": 8194,
            "type": "http"
        }],
        "ba-application": {
            "app-location": "${s3-source}/murachs-v6/"
        },
        "blusam": {
            "db": {
                "nb-threads": 8,
                "batch-size": 10000,
                "name": "blusam",
                "secret-manager-arn": "arn:aws:secretsmanager:us-
west-2:111122223333:secret:blusam-FfmXLG"
            },
            "redis": {
                "hostname": "blusam.c3geul.ng.0001.usw2.cache.amazonaws.com",
```

```
                "port": 6379,
                "useSsl": true,
                "secret-manager-arn": "arn:aws:secretsmanager:us-
west-2:111122223333:secret:bluesamredis-nioefm"
            }
        }
    }
}
```

AWS Detail definisi Blu Age

Pendengar (s) - diperlukan

Tentukan port yang akan Anda gunakan untuk mengakses aplikasi melalui Elastic Load Balancing yang dibuat Modernisasi AWS Mainframe. Gunakan struktur berikut:

```
"listeners": [{
            "port": 8194,
            "type": "http"
        }],
```

port

Wajib. Anda dapat menggunakan port apa pun yang tersedia kecuali port terkenal 0 hingga 1023. Kami merekomendasikan menggunakan kisaran dari 8192 hingga 8199. Pastikan tidak ada pendengar atau aplikasi lain yang beroperasi di port ini.

tipe

Wajib. Saat ini, hanya `http` didukung.

AWS Aplikasi Blu Age - diperlukan

Tentukan lokasi di mana mesin mengambil file gambar aplikasi menggunakan struktur berikut.

```
"ba-application": {
            "app-location": "${s3-source}/murachs-v6/",
            "files-directory": "/m2/mount/myfolder",
            "enable-jics": <true|false>,
            "shared-app-location": "${s3-source}/shared/"
        },
```

lokasi aplikasi

Lokasi spesifik di Amazon S3 tempat file gambar aplikasi disimpan.

file-direktori

Opsional. Lokasi file input/output untuk batch. Harus berupa subfolder dari pengaturan titik pemasangan Amazon EFS atau Amazon FSx di tingkat lingkungan.

aktifkan-jics

Opsional. Menentukan apakah akan mengaktifkan JICS. Default ke true. Menyetel ini ke false mencegah database JICS muncul.

shared-app-location

Opsional. Lokasi lebih lanjut di Amazon S3 tempat elemen aplikasi bersama disimpan. Ini dapat berisi jenis struktur aplikasi yang sama dengan lokasi aplikasi.

BluSam - opsional

Tentukan database BluSam dan cache Redis menggunakan struktur berikut.

```
"blusam": {
            "db": {
                "nb-threads": 8,
                "batch-size": 10000,
                "name": "blusam",
                "secret-manager-arn": "arn:aws:secretsmanager:us-
west-2:111122223333:secret:blusam-FfmXLG"
            },
            "redis": {
                "hostname": "blusam.c3geul.ng.0001.usw2.cache.amazonaws.com",
                "port": 6379,
                "useSsl": true,
                "secret-manager-arn": "arn:aws:secretsmanager:us-
west-2:111122223333:secret:bluesamredis-nioefm"
            }
        }
```

db

Menentukan properti database yang digunakan dengan aplikasi. Database harus berupa database Aurora PostgreSQL. Anda dapat menentukan properti berikut:

- `nb-threads`- Opsional. Menentukan berapa banyak thread khusus yang digunakan untuk mekanisme write-behind yang diandalkan oleh mesin blusam. Defaultnya adalah 8.
- `batch-size`- Opsional. Menentukan ambang batas yang digunakan mekanisme write-behind untuk memulai operasi penyimpanan batch. Ambang batas mewakili jumlah catatan yang dimodifikasi yang akan memulai operasi penyimpanan batch untuk memastikan bahwa catatan yang dimodifikasi tetap ada. Pemicunya sendiri didasarkan pada kombinasi ukuran batch dan waktu berlalu satu detik, mana yang dicapai terlebih dahulu. Defaultnya adalah 10000.
- `name`- Opsional. Menentukan nama database.
- `secret-manager-arn`- Menentukan Amazon Resource Name (ARN) dari rahasia yang berisi kredensi database. Untuk informasi selengkapnya, lihat Langkah 4: Buat dan konfigurasikan rahasia AWS Secrets Manager database.

redis

Menentukan properti cache Redis yang digunakan aplikasi untuk menyimpan data sementara yang dibutuhkan di lokasi pusat untuk meningkatkan kinerja. Kami menyarankan Anda mengenkripsi dan melindungi cache Redis dengan kata sandi.

- `hostname`- Menentukan lokasi cache Redis.
- `port`- Menentukan port, biasanya 6379, di mana cache Redis mengirim dan menerima komunikasi.
- `useSsl`- Menentukan apakah cache Redis dienkripsi. Jika cache tidak dienkripsi, atur `useSsl` ke false.
- `secret-manager-arn`- Menentukan Nama Sumber Daya Amazon (ARN) dari rahasia yang berisi kata sandi cache Redis. Jika cache Redis tidak dilindungi kata sandi, jangan tentukan. `secret-manager-arn` Untuk informasi selengkapnya, lihat Langkah 4: Buat dan konfigurasikan rahasia AWS Secrets Manager database.

AWS Antrian pesan Blu Age - opsional

Tentukan detail koneksi JMS-MQ untuk AWS aplikasi Blu Age.

```
"message-queues": [
  {
      "product-type": "JMS-MQ",
      "queue-manager": "QMgr1",
      "channel": "mqChannel1",
      "hostname": "mqserver-host1",
```

```
      "port": 1414,
      "user-id": "app-user1",
      "secret-manager-arn": "arn:aws:secretsmanager:us-
west-2:123456789012:secret:sample/mq/test-279PTa"
  },
  {
      "product-type": "JMS-MQ",
      "queue-manager": "QMgr2",
      "channel": "mqChannel2",
      "hostname": "mqserver-host2",
      "port": 1412,
      "user-id": "app-user2",
      "secret-manager-arn": "arn:aws:secretsmanager:us-
west-2:123456789012:secret:sample/mq/test-279PTa"
  }
]
```

jenis produk

Wajib. Menentukan jenis produk. Saat ini, ini hanya bisa “JMS-MQ” untuk AWS aplikasi Blu Age.

manajer antrian

Wajib. Menentukan nama manajer antrian.

saluran

Wajib. Menentukan nama saluran server-koneksi.

hostname

Wajib. Menentukan nama host dari server antrian pesan.

port

Wajib. Menentukan nomor port listener yang sedang didengarkan server.

id pengguna

Opsional. Menentukan ID akun pengguna yang diizinkan untuk melakukan operasi antrian pesan pada saluran yang ditentukan.

secret-manager-arn

Opsional. Menentukan Nama Sumber Daya Amazon (ARN) Secrets Manager yang menyediakan kata sandi pengguna yang ditentukan.

Definisi aplikasi Fokus Mikro

Bagian definisi sampel berikut adalah untuk mesin runtime Micro Focus, dan berisi elemen wajib dan opsional.

```
{
 "template-version": "2.0",
 "source-locations": [
        {
            "source-id": "s3-source",
            "source-type": "s3",
            "properties": {
                "s3-bucket": "mainframe-deployment-bucket-aaa",
                "s3-key-prefix": "v1"
            }
        }
    ],
    "definition" : {
        "listeners": [{
            "port": 5101,
            "type": "tn3270"
        }],
        "dataset-location": {
            "db-locations": [{
                "name": "Database1",
                "secret-manager-arn": "arn:aws:secrets:1234:us-east-1:secret:123456"
            }]
        },
        "cognito-auth-handler": {
            "user-pool-id": "cognito-idp.us-west-2.amazonaws.com/us-west-2_rvYFnQIxL",
            "client-id": "58k05jb8grukjjsudm5hhn1v87",
            "identity-pool-id": "us-west-2:64464b12-0bfb-4dea-ab35-5c22c6c245f6"
        },
        "ldap-ad-auth-handler": {
            "ldap-ad-connection-secrets": [LIST OF AD-SECRETS]
        },
        "batch-settings": {
            "initiators": [{
                "classes": ["A", "B"],
                "description": "initiator...."
            }],
            "jcl-file-location": "${s3-source}/batch/jcl"
        },
```

```
        "cics-settings": {
            "binary-file-location": "${s3-source}/cics/binaries",
            "csd-file-location": "${s3-source}/cics/def",
            "system-initialization-table": "BNKCICV"
        },
        "xa-resources" : [{
            "name": "XASQL",
            "secret-manager-arn": "arn:aws:secrets:1234:us-east-1:secret:123456",
            "module": "${s3-source}/xa/ESPGSQLXA64.so"
        }]
    }
}
```

Detail definisi Fokus Mikro

Konten di bagian definisi file definisi aplikasi Micro Focus bervariasi, tergantung pada sumber daya yang dibutuhkan aplikasi mainframe yang dimigrasi saat runtime.

Pendengar (s) - diperlukan

Tentukan pendengar menggunakan struktur berikut:

```
"listeners": [{
    "port": 5101,
    "type": "tn3270"
}],
```

port

Untuk tn3270, defaultnya adalah 5101. Untuk jenis pendengar layanan lainnya, port bervariasi. Anda dapat menggunakan port apa pun yang tersedia kecuali port terkenal 0 hingga 1023. Setiap pendengar harus memiliki port yang berbeda. Pendengar tidak boleh berbagi port. Untuk informasi selengkapnya, lihat Kontrol Pendengar dalam dokumentasi Micro Focus Enterprise Server.

tipe

Menentukan jenis pendengar layanan. Untuk informasi selengkapnya, lihat Pendengar dalam dokumentasi Micro Focus Enterprise Server.

Lokasi kumpulan data - diperlukan

Tentukan lokasi kumpulan data menggunakan struktur berikut.

```
"dataset-location": {
            "db-locations": [{
                "name": "Database1",
                "secret-manager-arn": "arn:aws:secrets:1234:us-east-1:secret:123456"
            }],
 }
```

db-lokasi

Menentukan lokasi kumpulan data yang dibuat oleh aplikasi yang dimigrasi. Saat ini, Modernisasi AWS Mainframe hanya mendukung kumpulan data dari satu database VSAM.

- `name`- Menentukan nama instance database yang berisi kumpulan data yang dibuat oleh aplikasi yang dimigrasi.
- `secret-manager-arn`- Menentukan Amazon Resource Name (ARN) dari rahasia yang berisi kredensi database.

Otentikasi Amazon Cognito dan penangan otorisasi - opsional

AWS Modernisasi Mainframe menggunakan Amazon Cognito untuk otentikasi dan otorisasi untuk aplikasi yang dimigrasi. Tentukan handler otentikasi Amazon Cognito menggunakan struktur berikut.

```
"cognito-auth-handler": {
            "user-pool-id": "cognito-idp.Region.amazonaws.com/Region_rvYFnQIxL",
            "client-id": "58k05jb8grukjjsudm5hhn1v87",
            "identity-pool-id": "Region:64464b12-0bfb-4dea-ab35-5c22c6c245f6"
}
```

user-pool-id

Menentukan kumpulan pengguna Amazon Cognito AWS yang digunakan Modernisasi Mainframe untuk mengautentikasi pengguna aplikasi yang dimigrasi. Wilayah AWS Untuk kumpulan pengguna harus cocok dengan Wilayah AWS untuk aplikasi Modernisasi AWS Mainframe.

id klien

Menentukan aplikasi yang dimigrasi yang dapat diakses oleh pengguna yang diautentikasi.

identity-pool-id

Menentukan kumpulan identitas Amazon Cognito tempat pengguna yang diautentikasi menukar token kumpulan pengguna dengan kredensil yang memungkinkan pengguna mengakses Modernisasi Mainframe. AWS Wilayah AWS Untuk kumpulan identitas harus cocok dengan Wilayah AWS untuk aplikasi Modernisasi AWS Mainframe.

LDAP dan pengendali Direktori Aktif - opsional

Anda dapat mengintegrasikan aplikasi Anda dengan Active Directory (AD) atau semua jenis server LDAP untuk memungkinkan pengguna aplikasi menggunakan kredensi LDAP/AD mereka untuk otorisasi dan otentikasi.

Untuk mengintegrasikan aplikasi Anda dengan AD

1. Ikuti langkah-langkah yang dijelaskan dalam Mengkonfigurasi Active Directory for Enterprise Server Security dalam dokumentasi Micro Focus Enterprise Server.
2. Buat AWS Secrets Manager rahasia dengan detail AD/LDAP Anda untuk setiap server AD/LDAP yang ingin Anda gunakan dengan aplikasi Anda. Untuk informasi tentang cara membuat rahasia, lihat Membuat rahasia AWS Secrets Manager di Panduan AWS Secrets Manager Pengguna. Untuk tipe rahasia, pilih Jenis rahasia lainnya dan sertakan pasangan kunci-nilai berikut.

```
{
    "connectionPath"       : "<HOST-ADDRESS>:<PORT>",
    "authorizedId"         : "<USER-FULL-DN>",
    "password"             : "<PASSWORD>",
    "baseDn"               : "<BASE-FULL-DN>",
    "userClassDn"          : "<USER-TYPE>",
    "userContainerDn"      : "<USER-CONTAINER-DN>",
    "groupContainerDn"     : "<GROUP-CONTAINER-DN>",
    "resourceContainerDn"  : "<RESOURCE-CONTAINER-DN>"
}
```

⚠ Rekomendasi keamanan

- Untuk`connectionPath`, Modernisasi AWS Mainframe mendukung protokol LDAP dan LDAP over SSL (LDAPS). Kami merekomendasikan penggunaan LDAPS karena lebih aman dan mencegah kredensil muncul dalam transmisi jaringan.
- Untuk `authorizedId` dan`password`, kami menyarankan Anda menentukan kredensil pengguna tanpa izin lebih dari izin baca dan verifikasi yang paling ketat yang diperlukan untuk menjalankan aplikasi Anda.
- Kami merekomendasikan untuk memutar kredensyal AD/LDAP secara teratur.
- Jangan membuat pengguna AD dengan nama pengguna `awsuser` ataum`fuser`. Kedua nama pengguna ini dicadangkan untuk AWS digunakan.

Berikut sebuah contoh.

```
{
    "connectionPath" : "ldaps://msad4.m2.example.people.aws.dev:636",
    "authorizedId" :
 "CN=LDAPUser,OU=Users,OU=msad4,DC=msad4,DC=m2,DC=example,DC=people,DC=aws,DC=dev",
    "password" : "ADPassword",
    "userContainerDn" : "CN=Enterprise Server Users,CN=Micro Focus,CN=Program
 Data,OU=msad4,DC=msad4,DC=m2,DC=example,DC=people,DC=aws,DC=dev",
    "groupContainerDn" : "CN=Enterprise Server Groups,CN=Micro Focus,CN=Program
 Data,OU=msad4,DC=msad4,DC=m2,DC=example,DC=people,DC=aws,DC=dev",
    "resourceContainerDn" : "CN=Enterprise Server Resources,CN=Micro
 Focus,CN=Program Data,OU=msad4,DC=msad4,DC=m2,DC=example,DC=people,DC=aws,DC=dev"
}
```

Buat rahasia dengan kunci KMS yang dikelola pelanggan. Anda harus memberikan Modernisasi AWS Mainframe `GetSecretValue` dan `DescribeSecret` izin pada rahasia, dan `Decrypt` dan `DescribeKey` izin pada kunci KMS. Untuk informasi selengkapnya, lihat Izin untuk kunci KMS di AWS Secrets Manager Panduan Pengguna.

3. Tambahkan yang berikut ini ke definisi aplikasi Anda.

```
"ldap-ad-auth-handler": {
    "ldap-ad-connection-secrets": [LIST OF AD/LDAP SECRETS]
```

```
}
```

Berikut sebuah contoh.

```
"ldap-ad-auth-handler": {
    "ldap-ad-connection-secrets": ["arn:aws:secrets:1234:us-east-1:secret:123456"]
}
```

Penangan otentikasi LDAP/AD tersedia untuk Micro Focus 8.0.11 dan versi yang lebih baru.

Pengaturan Batch - diperlukan

Tentukan detail yang diperlukan oleh pekerjaan batch yang dijalankan sebagai bagian dari aplikasi menggunakan struktur berikut.

```
"batch-settings": {
            "initiators": [{
                "classes": ["A","B"],
                "description": "initiator...."
            }],
            "jcl-file-location": "${s3-source}/batch/jcls"
 }
```

pemrakarsa

Menentukan inisiator batch yang dimulai ketika aplikasi yang dimigrasi mulai berhasil dan terus berjalan sampai aplikasi berhenti. Anda dapat menentukan satu atau beberapa kelas per inisiator. Anda juga dapat menentukan beberapa inisiator. Sebagai contoh:

```
"batch-settings": {
            "initiators": [
                {
                  "classes": ["A", "B"],
                  "description": "initiator...."
                },
                {
                  "classes": ["C", "D"],
                  "description": "initiator...."
                }
            ],
```

```
            "jcl-file-location": "${s3-source}/batch/jcls"
}
```

Untuk informasi selengkapnya, lihat Untuk menentukan pemrakarsa batch atau SEP printer dalam dokumentasi Micro Focus Enterprise Server.

- `classes`- Menentukan kelas pekerjaan yang inisiator dapat menjalankan. Anda dapat menggunakan hingga 36 karakter. Anda dapat menggunakan karakter berikut: A-Z atau 0-9.
- `description`- Menjelaskan untuk apa inisiator.
- `jcl-file-location`- Menentukan lokasi file JCL yang diperlukan oleh pekerjaan batch aplikasi yang dimigrasi berjalan.

Pengaturan CICS - diperlukan

Tentukan detail yang diperlukan untuk transaksi CICS yang berjalan sebagai bagian dari aplikasi menggunakan struktur berikut.

```
"cics-settings": {
            "binary-file-location": "${s3-source}/cics/binaries",
            "csd-file-location": "${s3-source}/cics/def",
            "system-initialization-table": "BNKCICV"
        }
```

binary-file-location

Menentukan lokasi file program transaksi CICS.

csd-file-location

Menentukan lokasi file definisi sumber daya CICS (CSD) untuk aplikasi ini. Untuk informasi selengkapnya, lihat Definisi Sumber Daya CICS dalam dokumentasi Micro Focus Enterprise Server.

system-initialization-table

Menentukan tabel inisialisasi sistem (SIT) yang digunakan aplikasi yang dimigrasi. Nama tabel SIT bisa sampai 8 karakter. Anda dapat menggunakan A-Z, 0-9, $, @, dan #. Untuk informasi selengkapnya, lihat Definisi Sumber Daya CICS dalam dokumentasi Micro Focus Enterprise Server.

Sumber daya XA - diperlukan

Tentukan detail yang diperlukan untuk sumber daya XA yang dibutuhkan aplikasi menggunakan struktur berikut.

```
"xa-resources" : [{
            "name": "XASQL",
            "secret-manager-arn": "arn:aws:secrets:1234:us-east-1:secret:123456",
            "module": "${s3-source}/xa/ESPGSQLXA64.so"
        }]
```

name

Wajib. Menentukan nama sumber daya XA.

secret-manager-arn

Menentukan Amazon Resource Name (ARN) untuk rahasia yang berisi kredensil untuk menghubungkan ke database.

modul

Menentukan lokasi file executable modul switch RM. Untuk informasi selengkapnya, lihat Merencanakan dan Merancang XAR dalam dokumentasi Micro Focus Enterprise Server.

AWSReferensi definisi kumpulan data modernisasi mainframe

Jika aplikasi Anda memerlukan lebih dari beberapa set data untuk diproses, memasukkannya satu per satu di konsol Modernisasi AWS Mainframe tidak efisien. Sebagai gantinya, kami menyarankan Anda membuat file JSON untuk menentukan setiap kumpulan data. Tipe kumpulan data yang berbeda ditentukan secara berbeda di JSON, meskipun banyak parameter yang umum. Dokumen ini menjelaskan rincian JSON yang diperlukan untuk mengimpor berbagai jenis kumpulan data.

> **Note**
>
> Sebelum Anda mengimpor kumpulan data apa pun, Anda harus mentransfer kumpulan data dari mainframe keAWS. Maka Anda harus memastikan bahwa kumpulan data dikonversi dari format mainframe ke format yang AWS dapat digunakan. Jika perlu, ubah data sesuai kebutuhan dan simpan kumpulan data yang diubah di Amazon S3. Tentukan nama bucket dan folder dalam file JSON definisi kumpulan data.

Jika Anda menggunakan mesin runtime Micro Focus, Anda dapat menggunakan `DFCONV` utilitas untuk mengonversi kumpulan data. Kami menyertakan utilitas ini dalam gambar Pengembang Perusahaan Fokus Mikro dan Server Perusahaan kami. Untuk informasi selengkapnya, lihat Konversi File Batch DFCONV dalam dokumentasi Pengembang Perusahaan Fokus Mikro.

Topik

Properti umum

Beberapa parameter umum untuk semua kumpulan data. Parameter ini mencakup bidang-bidang berikut:

- Informasi tentang kumpulan data (`datasetName`,`datasetOrg`,`recordLength`,`encoding`)
- Informasi tentang lokasi tempat Anda mengimpor; yaitu, lokasi sumber kumpulan data. Ini bukan lokasi di mainframe. Ini adalah jalur ke lokasi Amazon S3 tempat Anda mengunggah kumpulan data (). `externalLocation`
- Informasi tentang lokasi yang Anda impor; yaitu, lokasi target kumpulan data. Lokasi ini adalah database atau sistem file, tergantung pada mesin runtime Anda. (`storageTypedanrelativePath`).
- Informasi tentang tipe kumpulan data (tipe kumpulan data tertentu, format, pengkodean, dan sebagainya).

Setiap definisi kumpulan data memiliki struktur JSON yang sama. Contoh berikut JSON menunjukkan semua parameter umum ini.

```
{
    "dataSet": {
        "storageType": "Database",
```

```
        "datasetName": "MFI01V.MFIDEMO.BNKACC",
        "relativePath": "DATA",
        "datasetOrg": {
            "type": {
                type-specific properties
                    ...
            },
        },
    },
}
```

Properti berikut ini umum untuk semua kumpulan data.

storageType

Wajib. Berlaku untuk lokasi target. Menentukan apakah kumpulan data disimpan dalam database atau sistem file. Nilai yang mungkin adalah `Database` atau`FileSystem`.

- AWSMesin runtime Blu Age: sistem file tidak didukung. Anda harus menggunakan database.
- Micro Focus runtime engine: database dan sistem file keduanya didukung. Anda dapat menggunakan Amazon Relational Database Service atau Amazon Aurora untuk database, dan Amazon Elastic File System atau Amazon FSx for Lustre untuk sistem file.

DatasetName

Wajib. Menentukan nama yang sepenuhnya memenuhi syarat dari kumpulan data seperti yang muncul di mainframe.

RelativePath

Wajib. Berlaku untuk lokasi target. Menentukan lokasi relatif dari kumpulan data dalam database atau sistem file.

DatasetOrg

Wajib. Menentukan jenis kumpulan data. Kemungkinan nilai adalah `vsam`, `gdg`, `ps`, `po`, atau `unknown`.

- AWSMesin runtime Blu Age: hanya kumpulan data tipe VSAM yang didukung.
- Mesin runtime Micro Focus: VSAM, GDG, PS, PO, atau kumpulan data tipe Tidak Dikenal didukung.

> **Note**
>
> Jika aplikasi Anda memerlukan file yang bukan file data COBOL tetapi PDF atau file biner lainnya, Anda dapat menentukannya sebagai berikut:

```
"datasetOrg": {
            "type": PS {
                "format": U
            },
```

Contoh format permintaan kumpulan data untuk VSAM

- AWSMesin runtime Blu Age: didukung.
- Mesin runtime Micro Focus: didukung.

Jika Anda mengimpor kumpulan data VSAM, tentukan `vsam` sebagai. `datasetOrg` JSON Anda harus menyerupai contoh berikut:

```
{
    "storageType": "Database",
    "datasetName": "AWS.M2.VSAM.KSDS",
    "relativePath": "DATA",
    "datasetOrg": {
        "vsam": {

            "encoding": "A",
            "format": "KS",
            "primaryKey": {
                "length": 11,
                "offset": 0
            }
        }
    },
    "recordLength": {
        "min": 300,
        "max": 300
    }
```

```
},
"externalLocation": {
    "s3Location": "s3://$M2_DATA_STORE/catalog/data/AWS.M2.VSAM.KSDS.DAT"
}
```

Properti berikut didukung untuk kumpulan data VSAM.

encoding

Wajib. Menentukan pengkodean set karakter dari kumpulan data. Nilai yang mungkin adalah ASCII (`A`), EBCDIC (`E`), dan Unknown (). `?`

format

Wajib. Menentukan jenis set data VSAM dan format rekaman.

- AWSMesin runtime Blu Age: nilai yang mungkin adalah ESDS (`ES`), KSDS (), dan RRDS (`KS`). `RR` Format rekaman dapat diperbaiki atau variabel.
- Micro Focus runtime engine: nilai yang mungkin adalah ESDS (`ES`), KSDS (`KS`), dan RRDS (). `RR` Definisi VSAM menyertakan format rekaman, jadi Anda tidak perlu menentukannya secara terpisah.

PrimaryKey

Hanya berlaku untuk kumpulan data VSAM KSDS. Menentukan kunci utama. Terdiri dari nama kunci primer, offset kunci, dan panjang kunci. `name`Itu opsional; `offset` dan `length` diperlukan.

RecordLength

Wajib. Menentukan panjang catatan. Untuk format rekaman panjang tetap, nilai-nilai ini harus cocok.

- AWSMesin runtime Blu Age: untuk VSAM ESDS, KSDS, dan RRDS, bersifat opsional dan diperlukan. `min` `max`
- Mesin runtime Micro Focus: `min` dan `max` diperlukan.

Lokasi Eksternal

Wajib. Menentukan lokasi sumber: yaitu bucket Amazon S3 tempat Anda mengunggah kumpulan data.

Properti khusus mesin Blu Age

Mesin runtime AWS Blu Age mendukung kompresi untuk kumpulan data VSAM. Contoh berikut menunjukkan bagaimana Anda dapat menentukan properti ini di JSON.

```
{
    common properties
        ...
        "datasetOrg": {
            "vsam": {
                common properties
                ...
                "compressed": boolean,
                common properties
                ...
            }
        }
}
```

Tentukan properti kompresi sebagai berikut:

Kompresi

Opsional. Menentukan apakah indeks untuk kumpulan data ini disimpan sebagai nilai terkompresi. Jika Anda memiliki kumpulan data besar (biasanya > 100 Mb), pertimbangkan untuk menyetel flag ini`true`.

Contoh format permintaan kumpulan data untuk GDG Base

- AWSMesin runtime Blu Age: tidak didukung.
- Mesin runtime Micro Focus: didukung.

Jika Anda mengimpor kumpulan data dasar GDG, tentukan `gdg` sebagai. `datasetOrg` JSON Anda harus menyerupai contoh berikut:

```
{
    "storageType": "Database",
    "datasetName": "AWS.M2.GDG",
    "relativePath": "DATA",
    "datasetOrg": {
```

```
        "gdg": {
            "limit": "3",
            "rollDisposition": "Scratch and No Empty"
        }
    }
}
```

Properti berikut didukung untuk kumpulan data dasar GDG.

batasan

Wajib. Menentukan jumlah generasi aktif, atau bias. Untuk cluster dasar GDG, maksimumnya adalah 255.

RollDisposisi

Opsional. Menentukan bagaimana menangani set data generasi ketika maksimum tercapai atau terlampaui. Kemungkinan nilainya adalah `No Scratch and No Empty`, `Scratch and No Empty`, `Scratch and Empty`, atau `No Scratch and Empty`. Default-nya adalah `Scratch and No Empty`.

Contoh format permintaan kumpulan data untuk Generasi PS atau GDG

- AWSMesin runtime Blu Age: tidak didukung.
- Mesin runtime Micro Focus: didukung.

Jika Anda mengimpor kumpulan data generasi PS atau GDG, tentukan `ps` sebagai. `datasetOrg` JSON Anda harus menyerupai contoh berikut:

```
{
    "storageType": "Database",
    "datasetName": "AWS.M2.PS.FB",
    "relativePath": "DATA",
    "datasetOrg": {
        "ps": {
            "format": "FB",
            "encoding": "A"
        }
    },
    "recordLength": {
        "min": 300,
```

```
        "max": 300
    }
},
"externalLocation": {
    "s3Location": "s3://$M2_DATA_STORE/catalog/data/AWS.M2.PS.LSEQ"
}
}
```

Properti berikut didukung untuk kumpulan data generasi PS atau GDG.

format

Wajib. Menentukan format data set catatan. Nilai yang mungkin adalah `FFA`,`FB`,`FBA`,`FBM`,`FBS`,`FM`,`FS`,`LSEQ`,`U`,`V`,`VA`,`VB`,`VBA`,`VBM`,`VBS`,`VM`, dan`VS`.

encoding

Wajib. Menentukan pengkodean set karakter dari kumpulan data. Nilai yang mungkin adalah ASCII (`A`), EBCDIC (`E`), dan Unknown () ?

RecordLength

Wajib. Menentukan panjang catatan. Anda harus menentukan panjang minimum (`min`) dan maksimum (`max`) catatan. Untuk format rekaman panjang tetap, nilai-nilai ini harus cocok.

Lokasi Eksternal

Wajib. Menentukan lokasi sumber: yaitu bucket Amazon S3 tempat Anda mengunggah kumpulan data.

Contoh format permintaan kumpulan data untuk PO

Jika Anda mengimpor kumpulan data PO, tentukan `po` sebagai. `datasetOrg` JSON Anda harus menyerupai contoh berikut:

```
{
    "storageType": "Database",
    "datasetName": "AWS.M2.PO.PROC",
    "relativePath": "DATA",
    "datasetOrg": {
        "po": {
            "format": "LSEQ",
            "encoding": "A",
            "memberFileExtensions": ["PRC"]
```

```
        }
    },
    "recordLength": {
        "min": 80,
        "max": 80
    }
},
"externalLocation": {
    "s3Location": "s3://$M2_DATA_STORE/source/proc/"
}
}
```

Properti berikut didukung untuk kumpulan data PO.

format

Wajib. Menentukan format data set catatan. Nilai yang mungkin adalah `FFA`,`FB`,`FBA`,`FBM`,`FBS`,`FM`,`FS`,`LSEQ`,`U`,`V`,`VA`,`VB`,`VBA`,`VBM`,`VBS`,`VM`, dan`VS`.

encoding

Wajib. Menentukan pengkodean set karakter dari kumpulan data. Nilai yang mungkin adalah ASCII (`A`), EBCDIC (`E`), dan Unknown (). `?`

memberFileExtensions

Wajib. Menentukan array yang berisi satu atau lebih ekstensi nama file, memungkinkan Anda untuk menentukan file yang akan dimasukkan sebagai anggota PDS.

RecordLength

Opsional. Menentukan panjang catatan. Panjang minimum (`min`) dan maksimum (`max`) catatan adalah opsional. Untuk format rekaman panjang tetap, nilai-nilai ini harus cocok.

Lokasi Eksternal

Wajib. Menentukan lokasi sumber: yaitu bucket Amazon S3 tempat Anda mengunggah kumpulan data.

> **Note**
>
> Implementasi saat ini untuk mesin runtime Micro Focus menambahkan entri PDS sebagai kumpulan data dinamis.

Lingkungan Runtime Terkelola dalam Modernisasi AWS Mainframe

Jika Anda baru mengenal Modernisasi AWS Mainframe, lihat topik berikut untuk memulai:

- Apa itu Modernisasi AWS Mainframe?
- Menyiapkan AWS Modernisasi Mainframe
- Memulai Modernisasi AWS Mainframe
- Tutorial: Runtime Terkelola untuk AWS Blu Age
- Tutorial: Runtime terkelola untuk Micro Focus

Lingkungan runtime di Modernisasi AWS Mainframe adalah kombinasi bernama sumber daya AWS komputasi, mesin runtime, dan detail konfigurasi yang Anda tentukan. Lingkungan runtime menghosting satu atau lebih aplikasi. Aplikasi dalam Modernisasi AWS Mainframe berisi beban kerja mainframe yang dimigrasi. Anda dapat memilih mesin runtime untuk lingkungan yang Anda buat. Pilih AWS Blu Age jika Anda menggunakan pola refactoring otomatis, dan Micro Focus jika Anda menggunakan pola replatforming. Anda juga dapat memilih jumlah sumber daya komputasi yang tepat untuk aplikasi Anda dan secara opsional melampirkan penyimpanan ke lingkungan runtime. AWS Modernisasi Mainframe memungkinkan CloudWatch metrik Amazon dan pencatatan untuk Anda sehingga Anda dapat memantau lingkungan runtime Anda.

Topik

- Buat lingkungan runtime Modernisasi AWS Mainframe
- Perbarui lingkungan AWS runtime Modernisasi Mainframe
- Hentikan lingkungan AWS runtime Modernisasi Mainframe
- Mulai ulang lingkungan AWS runtime Modernisasi Mainframe
- Menghapus lingkungan AWS runtime Modernisasi Mainframe

Buat lingkungan runtime Modernisasi AWS Mainframe

Gunakan konsol Modernisasi AWS Mainframe untuk membuat lingkungan Modernisasi AWS Mainframe.

Instruksi ini mengasumsikan bahwa Anda telah menyelesaikan langkah-langkahnyaMenyiapkan AWS Modernisasi Mainframe.

Buat lingkungan runtime

Untuk membuat lingkungan runtime

1. Buka konsol Modernisasi AWS Mainframe di https://console.aws.amazon.com/m2/.
2. Di Wilayah AWS pemilih, pilih Wilayah tempat Anda ingin membuat lingkungan.
3. Pada halaman Environments, pilih Create environment.
4. Pada halaman Tentukan informasi dasar, berikan informasi berikut:
    a. Di bagian Nama dan deskripsi, masukkan nama untuk lingkungan.
    b. (Opsional) Di bidang deskripsi Lingkungan, masukkan deskripsi untuk lingkungan. Deskripsi ini dapat membantu Anda dan pengguna lain mengidentifikasi tujuan lingkungan runtime.
    c. Di bagian opsi Engine, pilih Blu Age untuk refactoring otomatis, atau Micro Focus untuk replatforming.
    d. Pilih versi untuk mesin yang Anda pilih.
    e. (Opsional) Di bagian Tag, pilih Tambahkan tag baru untuk menambahkan satu atau beberapa tag lingkungan ke lingkungan Anda. Tag lingkungan adalah label atribut khusus yang membantu Anda mengatur dan mengelola AWS sumber daya Anda.
    f. Pilih Berikutnya.
5. Pada halaman Tentukan konfigurasi, berikan informasi berikut:
    a. Di bagian Ketersediaan, pilih Lingkungan runtime mandiri atau Kluster ketersediaan tinggi.

        Pola ketersediaan menentukan seberapa tersedia aplikasi Anda saat dijalankan. Standalone baik-baik saja untuk tujuan pengembangan. Ketersediaan tinggi adalah untuk aplikasi yang harus tersedia setiap saat.
    b. Di Sumber Daya, pilih jenis instans dan kapasitas yang diinginkan.

        Sumber daya ini adalah instans Amazon EC2 yang dikelola Modernisasi AWS Mainframe yang akan menghosting lingkungan runtime Anda. Lingkungan runtime mandiri menawarkan dua pilihan untuk jenis instance dan hanya mengizinkan satu instance. Lingkungan runtime ketersediaan tinggi menawarkan dua pilihan untuk jenis instans dan mengizinkan hingga dua instance.

Untuk informasi selengkapnya, lihat Jenis Instans Amazon EC2, dan hubungi spesialis AWS mainframe untuk panduan.

6. Di bagian Keamanan dan jaringan, lakukan hal berikut:

   a. Jika Anda ingin aplikasi dapat diakses publik, pilih Izinkan aplikasi yang disebarkan ke lingkungan ini agar dapat diakses publik.

   b. Pilih Virtual Private Cloud (VPC)

   c. Jika Anda menggunakan pola ketersediaan tinggi, pilih dua atau lebih subnet. Jika Anda menggunakan pola mandiri dengan mesin AWS Blu Age, pilih dua atau lebih subnet. Jika Anda menggunakan pola mandiri dengan mesin Micro Focus, Anda dapat menentukan satu subnet.

   d. Pilih grup keamanan untuk VPC yang Anda pilih.

      > **Note**
      >
      > AWSModernisasi Mainframe menciptakan Network Load Balancer bagi Anda untuk mendistribusikan koneksi ke lingkungan runtime Anda. Pastikan aturan masuk grup keamanan Anda mengizinkan akses dari alamat IP ke port yang Anda tentukan di `listener` properti definisi aplikasi. Untuk informasi selengkapnya, lihat Mendaftarkan target di Panduan Pengguna untuk Penyeimbang Beban Jaringan.

   e. Di bidang kunci KMS, pilih Sesuaikan pengaturan enkripsi jika Anda ingin menggunakan pelanggan yang dikelolaAWS KMS key. Untuk informasi selengkapnya, lihat Enkripsi data saat istirahat untuk layanan Modernisasi AWS Mainframe.

      > **Note**
      >
      > Secara default, Modernisasi AWS Mainframe mengenkripsi data Anda dengan Modernisasi AWS Mainframe AWS KMS key yang dimiliki dan dikelola untuk Anda. Namun, Anda dapat memilih untuk menggunakan pelanggan yang dikelolaAWS KMS key.

   f. (Opsional) Pilih AWS KMS key berdasarkan nama atau Nama Sumber Daya Amazon (ARN). Bergantian, pilih Buat AWS KMS key untuk pergi ke AWS KMS konsol dan buat yang baru. AWS KMS key

   g. Pilih Berikutnya.

7. (Opsional) Pada halaman Lampirkan penyimpanan, pilih satu atau beberapa sistem file Amazon EFS atau Amazon FSx, lalu pilih Berikutnya.
8. Di bagian jendela Pemeliharaan, pilih kapan Anda ingin menerapkan perubahan yang tertunda ke lingkungan.

   - Jika Anda memilih Tidak ada preferensi, Modernisasi AWS Mainframe memilih jendela pemeliharaan yang dioptimalkan untuk Anda.
   - Jika Anda ingin menentukan jendela pemeliharaan tertentu, pilih Pilih jendela pemeliharaan baru. Kemudian pilih hari dalam seminggu, waktu mulai, dan durasi untuk jendela pemeliharaan.

   Untuk informasi selengkapnya tentang jendela pemeliharaan, lihatAWSJendela pemeliharaan Modernisasi Mainframe.

   Pilih Berikutnya.
9. Pada halaman Tinjau dan buat, tinjau informasi yang Anda masukkan, lalu pilih Buat lingkungan.

Perbarui lingkungan AWS runtime Modernisasi Mainframe

Gunakan konsol Modernisasi AWS Mainframe untuk memperbarui lingkungan runtime Modernisasi AWS Mainframe. Anda dapat memperbarui versi minor dari mesin runtime atau jenis instans yang menghosting lingkungan runtime. Anda dapat memilih apakah Anda ingin menerapkan pembaruan segera atau selama jendela pemeliharaan yang diinginkan.

Petunjuk ini berasumsi bahwa Anda telah menyelesaikan langkah-langkah di Menyiapkan AWS Modernisasi Mainframe.

Perbarui lingkungan runtime

Untuk memperbarui lingkungan runtime

1. Buka konsol Modernisasi AWS Mainframe di https://console.aws.amazon.com/m2/.
2. Di Wilayah AWS pemilih, pilih Wilayah tempat lingkungan yang ingin Anda perbarui dibuat.
3. Pada halaman Lingkungan, pilih lingkungan yang ingin Anda perbarui.
4. Pada halaman detail untuk lingkungan, pilih Tindakan, lalu pilih Edit lingkungan.
5. Lakukan salah satu dari perubahan berikut:

- Di bagian Opsi mesin, pilih versi mesin yang Anda inginkan.
- Di bagian Sumber Daya, pilih jenis instance yang Anda inginkan.
- Di bagian jendela Pemeliharaan, pilih hari, waktu, dan durasi yang Anda inginkan.

> **Note**
>
> Satu-satunya perubahan yang dapat Anda pilih untuk diterapkan selama jendela pemeliharaan adalah perubahan pada versi mesin. Anda harus segera menerapkan semua perubahan lainnya.

6. Pilih Berikutnya.
7. Di Kapan menerapkan perubahan ini, pilih Segera atau Selama jendela pemeliharaan berikutnya. Kemudian pilih Perbarui lingkungan.

Jika Anda memilih Segera, Anda akan melihat pesan ketika lingkungan telah selesai diperbarui.

AWSJendela pemeliharaan Modernisasi Mainframe

Setiap lingkungan runtime memiliki jendela pemeliharaan satu jam mingguan. Setiap perubahan sistem diterapkan selama waktu ini. Jendela pemeliharaan adalah kesempatan Anda untuk mengontrol kapan modifikasi dan penambalan perangkat lunak terjadi. Jika acara pemeliharaan dijadwalkan untuk minggu tertentu, itu dimulai selama jendela pemeliharaan satu jam itu. Sebagian besar acara pemeliharaan juga selesai selama jendela pemeliharaan satu jam, meskipun acara pemeliharaan yang lebih besar mungkin memakan waktu lebih dari satu jam untuk diselesaikan.

Jendela pemeliharaan satu jam dipilih secara acak dari blok waktu 8 jam per Wilayah. Jika Anda tidak menentukan jendela pemeliharaan saat membuat lingkungan runtime, Modernisasi AWS Mainframe menetapkan jendela pemeliharaan 1 jam pada hari yang dipilih secara acak dalam seminggu.

AWSModernisasi Mainframe menghabiskan beberapa sumber daya di instans lingkungan Anda saat pemeliharaan sedang diterapkan. Anda mungkin mengamati efek minimal pada kinerja atau beberapa gangguan dalam aplikasi selama pemeliharaan.

Tabel berikut menunjukkan blok waktu default ketika jendela pemeliharaan ditetapkan untuk setiap Wilayah.

Nama Wilayah	Wilayah	Blok Waktu
US East (Northern Virginia)	us-east-1	03:00–11:00 UTC
AS Barat (Oregon)	us-west-2	06.00–14.00 UTC
Asia Pacific (Mumbai)	ap-south-1	06.00–14.00 UTC
Asia Pasifik (Singapura)	ap-southeast-1	14.00–22.00 UTC
Asia Pasifik (Sydney)	ap-southeast-2	12.00–20.00 UTC
Asia Pasifik (Tokyo)	ap-northeast-1	13.00–21.00 UTC
Kanada (Pusat)	ca-central-1	03:00–11:00 UTC
Eropa (Frankfurt)	eu-central-1	21.00–05.00 UTC
Eropa (Irlandia)	eu-west-1	22.00–06.00 UTC
Eropa (London)	eu-west-2	22:00–06:00 UTC
Europe (Paris)	eu-west-3	23.59–07.29 UTC
Amerika Selatan (Sao Paulo)	sa-east-1	00.00–08.00 UTC

Hentikan lingkungan AWS runtime Modernisasi Mainframe

Gunakan konsol Modernisasi AWS Mainframe untuk menghentikan lingkungan runtime Modernisasi AWS Mainframe. Saat Anda menghentikan lingkungan, penerapan aplikasi saat ini dipertahankan dan Anda tidak akan dikenakan biaya untuk lingkungan tersebut hingga lingkungan dimulai ulang.

Petunjuk ini berasumsi bahwa Anda telah menyelesaikan langkah-langkah di Menyiapkan AWS Modernisasi Mainframe.

Hentikan lingkungan runtime

Jika Anda perlu menghentikan lingkungan runtime Modernisasi AWS Mainframe, Anda mengikuti langkah serupa seperti bagian lingkungan pembaruan.

Gunakan konsol Modernisasi AWS Mainframe untuk menghentikan lingkungan runtime Modernisasi AWS Mainframe. Ketika Anda menghentikan lingkungan, penerapan aplikasi saat ini dipertahankan dan Anda tidak akan dikenakan biaya untuk lingkungan sampai lingkungan dimulai ulang.

Hentikan lingkungan runtime

Untuk menghentikan lingkungan runtime Modernisasi AWS Mainframe, Anda mengikuti langkah serupa seperti bagian lingkungan pembaruan.

> **Note**
>
> Anda harus menghentikan semua aplikasi sebelum menghentikan lingkungan.

Untuk menghentikan lingkungan runtime

1. Buka konsol Modernisasi AWS Mainframe di https://console.aws.amazon.com/m2/.
2. Di Wilayah AWS pemilih, pilih Wilayah tempat lingkungan yang ingin Anda hentikan dibuat.
3. Pada halaman Lingkungan, pilih lingkungan yang ingin Anda hentikan.
4. Pada halaman detail untuk lingkungan, pilih Tindakan, lalu pilih Edit lingkungan.
5. Pada halaman Edit lingkungan, temukan bagian Sumber Daya, dan perbarui kapasitas yang diinginkan ke nol.

   > **Note**
   >
   > Untuk menghentikan lingkungan, Anda hanya dapat memilih untuk segera berhenti.

6. Pilih Berikutnya.
7. Di Kapan menerapkan perubahan ini, pilih Segera. Kemudian pilih Perbarui lingkungan.

Anda melihat pesan saat kapasitas lingkungan diperbarui.

Mulai ulang lingkungan AWS runtime Modernisasi Mainframe

Gunakan konsol Modernisasi AWS Mainframe untuk memulai ulang lingkungan runtime Modernisasi AWS Mainframe. Saat Anda memulai ulang lingkungan runtime, penagihan untuk lingkungan akan dilanjutkan.

Mulai ulang lingkungan runtime

Untuk memulai ulang lingkungan runtime Modernisasi AWS Mainframe, Anda mengikuti langkah serupa seperti bagian stop environment.

Untuk memulai ulang lingkungan runtime

1. Buka konsol Modernisasi AWS Mainframe di https://console.aws.amazon.com/m2/.
2. Di Wilayah AWS pemilih, pilih Wilayah tempat lingkungan yang ingin Anda restart dibuat.
3. Pada halaman Environments, pilih lingkungan yang ingin Anda restart.
4. Pada halaman detail untuk lingkungan, pilih Tindakan, lalu pilih Edit lingkungan.

> **Note**
>
> Kapasitas yang diinginkan untuk lingkungan mandiri hanya dapat diperbarui ke 1. Untuk memulai ulang lingkungan runtime, Anda hanya dapat memilih untuk memulai ulang segera.

5. Pada halaman Edit lingkungan, temukan bagian Sumber Daya, dan perbarui kapasitas yang diinginkan dari nol ke kapasitas yang diperlukan.
6. Pilih Berikutnya.
7. Di Kapan menerapkan perubahan ini, pilih Segera. Kemudian pilih Perbarui lingkungan.

Anda melihat pesan ketika kapasitas lingkungan diperbarui dan lingkungan dimulai ulang.

Menghapus lingkungan AWS runtime Modernisasi Mainframe

Gunakan konsol Modernisasi AWS Mainframe untuk menghapus lingkungan runtime Modernisasi AWS Mainframe.

Petunjuk ini berasumsi bahwa Anda telah menyelesaikan langkah-langkah di Menyiapkan AWS Modernisasi Mainframe.

Menghapus lingkungan runtime

Jika Anda perlu menghapus lingkungan runtime Modernisasi AWS Mainframe, pastikan Anda menghapus aplikasi yang diterapkan dari lingkungan terlebih dahulu. Anda tidak dapat menghapus lingkungan runtime tempat aplikasi digunakan.

Untuk menghapus lingkungan

1. Buka konsol Modernisasi AWS Mainframe di https://console.aws.amazon.com/m2/.
2. Di Wilayah AWS pemilih, pilih Wilayah tempat lingkungan yang ingin Anda hapus dibuat.
3. Pada halaman Environments, pilih lingkungan yang ingin Anda hapus, lalu pilih Actions and Delete environment.
4. Di jendela Hapus lingkungan, masukkan `delete` untuk mengonfirmasi bahwa Anda ingin menghapus lingkungan runtime, lalu pilih Hapus.

Pengujian Aplikasi dalam Modernisasi AWS Mainframe

AWS Pengujian Aplikasi dalam rilis pratinjau untuk Modernisasi AWS Mainframe dan dapat berubah sewaktu-waktu. Kami menyarankan Anda menggunakan fitur ini hanya dengan data pengujian dan aplikasi, dan bukan di lingkungan produksi.

AWS Pengujian Aplikasi Modernisasi Mainframe menyediakan pengujian kesetaraan fungsional otomatis untuk proyek migrasi Anda.

Topik

Apa itu Pengujian Aplikasi Modernisasi AWS Mainframe?

AWSPengujian Aplikasi dalam rilis pratinjau untuk Modernisasi AWS Mainframe dan dapat berubah sewaktu-waktu. Kami menyarankan Anda menggunakan fitur ini hanya dengan data pengujian dan aplikasi, dan bukan di lingkungan produksi.

Pengujian berdampak pada proyek migrasi secara signifikan. Ini dapat menghabiskan hingga 70% dari migrasi, modernisasi, atau waktu dan upaya proyek augmentasi Anda. AWS Pengujian Aplikasi, fitur Modernisasi AWS Mainframe, menyediakan pengujian kesetaraan fungsional otomatis untuk aplikasi yang dimigrasikan. Pengujian kesetaraan fungsional membantu Anda memvalidasi bahwa aplikasi Anda setara dengan aplikasi AWS Cloud Anda di mainframe Anda. AWS Pengujian Aplikasi secara otomatis membandingkan perubahan pada kumpulan data, catatan basis data, dan layar 3270 online antara mainframe Anda dan. AWS Selain itu, Pengujian Aplikasi memungkinkan pengujian berulang, sehingga Anda dapat menjalankan skenario pengujian berkali-kali saat memperbarui arsitektur target, menyelesaikan masalah, dan maju menuju aplikasi yang sepenuhnya dimigrasi.

Setelah migrasi, Anda dapat terus menggunakan Pengujian Aplikasi untuk pengujian regresi, untuk memastikan bahwa pembaruan ke mesin runtime atau komponen lain tidak menyebabkan regresi. Pengujian Aplikasi hemat biaya: lingkungan pengujian target dibuat menggunakan CloudFormation templat yang disediakan pengguna, memanfaatkan konsep Infrastructure-as-Code (IAc). Pengujian Aplikasi mempercepat proyek migrasi menggunakan elastisitas cloud. Anda dapat menjalankan skenario pengujian independen pada lingkungan paralel sebanyak yang diperlukan, mengurangi jadwal pengujian.

Topik

Apakah Anda pengguna Pengujian Aplikasi pertama kali?

Jika Anda adalah pengguna pertama kali Pengujian Aplikasi, kami sarankan Anda mulai dengan membaca bagian berikut:

- Konsep Pengujian Aplikasi
- Tutorial: Mengatur CardDemo

Manfaat Pengujian Aplikasi

Pengujian Aplikasi memberikan beberapa manfaat untuk membantu Anda dalam proses migrasi:

- Menguji akselerasi, kelincahan, dan fleksibilitas
- Konsep pengujian “Rekam sekali di mainframe, putar ulang beberapa kali di AWS”
- Pembuatan lingkungan target IAC melalui templat yang disediakan pengguna CloudFormation
- Tingkat pengulangan pengujian yang tinggi
- Dibangun untuk cloud, dengan skalabilitas dan elastisitas dalam pikiran

- Pengujian skala besar dengan otomatisasi tingkat tinggi
- Efisiensi biaya

Integrasi dengan AWS CloudFormation

Pengujian Aplikasi menggunakan infrastruktur sebagai kode denganAWS CloudFormation. Pilihan desain ini menyederhanakan dan meningkatkan pengalaman pengujian Anda. AWS CloudFormationmemberi Anda otonomi dan kemandirian untuk menentukan infrastruktur yang lebih baik untuk kebutuhan Anda. Anda dapat memilih atau menentukan untuk banyak parameter (ukuran instans, contoh RDS, grup keamanan optimal) secara independen. Anda dapat menambahkan sumber daya, seperti antrean Amazon SQS yang Anda perlukan agar aplikasi berfungsi dengan baik dalam kondisi pengujian.

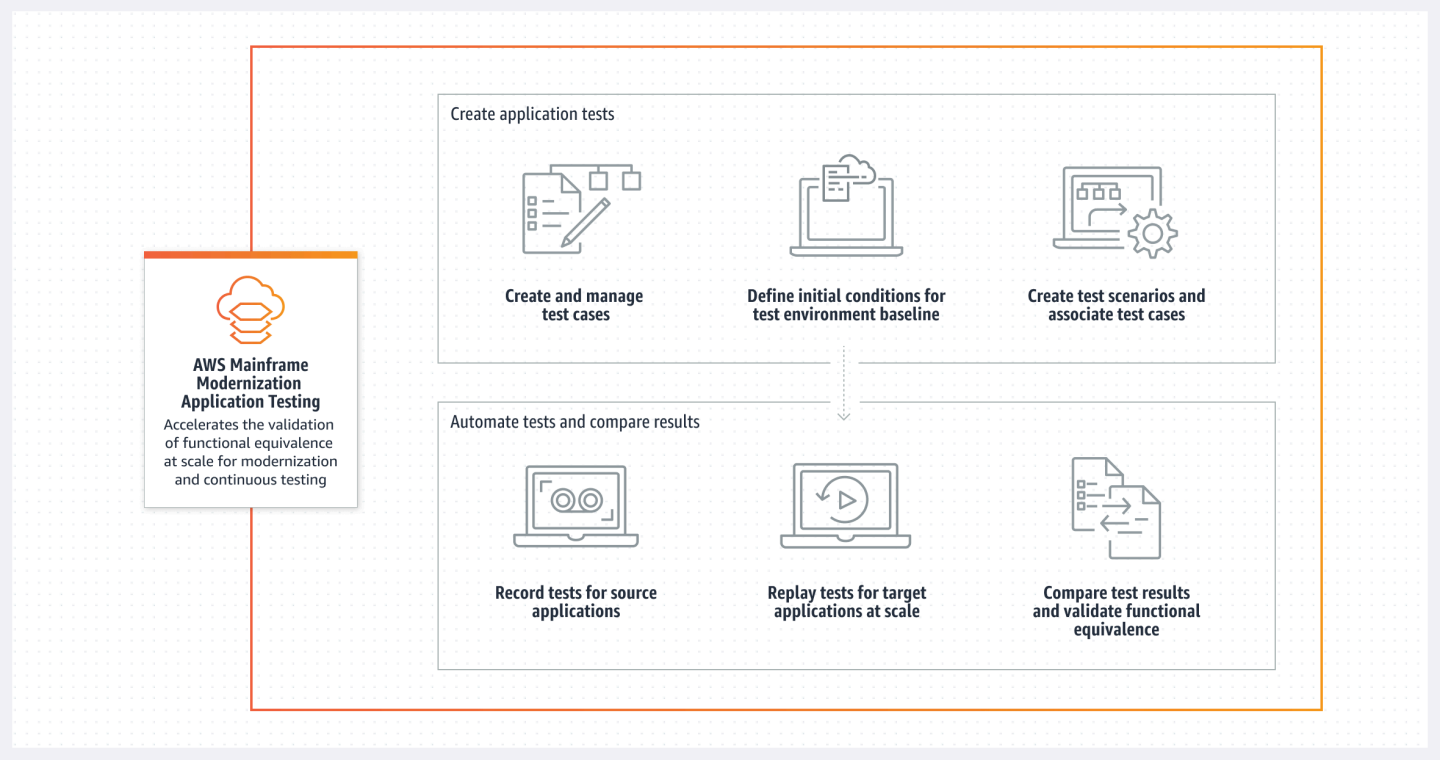
Dalam AWS CloudFormation templat yang disediakan untuk diunduh, Anda akan melihat beberapa fitur umum:

- Pengujian Aplikasi menciptakan tumpukan yang sepenuhnya terisolasi, termasuk lingkungan dan aplikasi runtime Modernisasi AWS Mainframe, dengan definisi jaringan dan keamanannya sendiri. Tumpukan terisolasi ini memberikan ketahanan, karena aktor lain dalam hal yang sama Akun AWS tidak dapat mengganggu aktivitas pengujian. Ini juga menghindari situasi di mana operator sistem memodifikasi VPC default atau grup keamanan, yang dapat menyebabkan kegagalan aktivitas pengujian.
- Grup keamanan juga memungkinkan Anda untuk mengontrol akses eksternal ke sumber daya yang digunakan dalam pengujian. Misalnya, database mungkin berisi data rahasia.
- Isolasi penuh mencegah aktor lain yang berbagi VPC mengintip lalu lintas.
- Ini meningkatkan kinerja. Misalnya, komunikasi antara aplikasi Modernisasi AWS Mainframe yang dibuat template dan database Amazon RDS terjadi pada jaringan terpisah (VPC pribadi), yang menghindari aktor lain memperlambat lalu lintas.

Kami menyarankan Anda menerapkan fitur-fitur ini di AWS CloudFormation template yang Anda buat juga.

Bagaimana Pengujian Aplikasi bekerja

Gambar berikut adalah ikhtisar tentang bagaimana pengujian kesetaraan fungsional dalam Pengujian Aplikasi bekerja.



- - Anda dapat mentransfer data input dari sumber ke AWS menggunakan Transfer File Modernisasi AWS Mainframe atau alat pilihan Anda untuk transfer data mainframe.
- Anda menjalankan logika bisnis yang sama pada sumber dan target.
- Pengujian Aplikasi secara otomatis membandingkan data output (dataset, perubahan database relasional, 3270 layar dan interaksi pengguna) dari sumber dan target. Setelah Anda menjalankan skenario pengujian di mainframe, Anda menangkap data keluaran dan mentransfernyaAWS, lalu memutar ulang skenario pengujian pada target. Pengujian Aplikasi secara otomatis membandingkan data keluaran dari pengujian yang dijalankan AWS dengan data keluaran dari sumbernya. Anda dapat melihat sekilas catatan mana yang identik, setara, berbeda, atau hilang. Selain itu, Anda dapat menentukan aturan kesetaraan, sehingga catatan yang tidak identik tetapi memiliki arti bisnis yang sama dipahami setara.

Alur kerja yang Anda ikuti dalam Pengujian Aplikasi terdiri dari langkah-langkah berikut:

1. Buat kasus uji. Kasus uji adalah unit terkecil dari tindakan pengujian. Saat Anda membuat kasus uji, Anda juga mengidentifikasi tipe data yang akan dibandingkan yang paling mewakili kesetaraan fungsional antara sumber dan target.
2. Buat skenario pengujian. Skenario pengujian mengelompokkan kasus uji terkait ke dalam urutan tertentu untuk dijalankan.

3. Buat kondisi awal. Kondisi awal menjelaskan cara merekam tes pada mainframe dan digunakan AWS CloudFormation untuk secara otomatis membuat status yang sama. AWS
4. Rekam pada sumber dan putar ulang pada target. Tangkap kumpulan data input dan output pada mainframe, dan unggah ke. AWS Kemudian putar ulang skenario pengujian padaAWS.
5. Bandingkan kumpulan data sumber dan target. Pengujian Aplikasi secara otomatis membandingkan kumpulan data keluaran dari sumber dan target, sehingga Anda dapat melihat sekilas apa yang benar dan apa yang tidak.

Baik tindakan akhir dari skenario pengujian dan tujuan dari seluruh proses adalah untuk mengidentifikasi perbedaan antara sumber dan uji target yang dijalankan. Pengujian Aplikasi membandingkan versi sumber dan versi target untuk data yang diambil pada semua saluran interaksi selama uji coba. Ini juga membandingkan keadaan akhir dari data yang relevan (seperti yang didefinisikan dalam kasus uji).

Layanan-layanan terkait

Pengujian Aplikasi adalah fitur Modernisasi AWS Mainframe. Ini juga menggunakan infrastruktur sebagai kode AWS CloudFormation untuk memastikan pengujian pengulangan, otomatisasi, dan efisiensi biaya. Untuk informasi selengkapnya, lihat:

- AWSModernisasi Mainframe
- AWS CloudFormation

Mengakses Pengujian Aplikasi

Anda dapat mengakses Pengujian Aplikasi dari konsol Modernisasi AWS Mainframe dengan memilih Pengujian Aplikasi di navigasi kiri.

Harga untuk Pengujian Aplikasi

Harga untuk Pengujian Aplikasi dapat ditemukan di Harga Modernisasi AWS Mainframe.

AWS Konsep Pengujian Aplikasi Modernisasi Mainframe

AWS Pengujian Aplikasi dalam rilis pratinjau untuk Modernisasi AWS Mainframe dan dapat berubah sewaktu-waktu. Kami menyarankan Anda menggunakan fitur ini hanya dengan data pengujian dan aplikasi, dan bukan di lingkungan produksi.

AWS Pengujian Aplikasi menggunakan istilah yang mungkin digunakan oleh layanan pengujian atau paket perangkat lunak lain dengan arti yang sedikit berbeda. Bagian berikut menjelaskan bagaimana Pengujian Aplikasi Modernisasi AWS Mainframe menggunakan terminologi ini.

Topik

- Perbedaan
- Kesetaraan
- Aplikasi sumber
- Aplikasi target

Kasus uji

Kasus uji adalah unit aksi paling atom individual dalam alur kerja pengujian Anda. Biasanya, kasus uji digunakan untuk mewakili unit independen logika bisnis yang memodifikasi data. Perbandingan akan dilakukan untuk setiap kasus uji. Kasus uji ditambahkan ke skenario pengujian. Kasus uji berisi metadata tentang artefak data (kumpulan data, database) yang dimodifikasi oleh kasus uji dan tentang fungsi bisnis yang dipicu selama eksekusi kasus uji: pekerjaan batch, 3270 dialog interaktif, dan lainnya. Misalnya, nama dan halaman kode kumpulan data.

Masukan data → Kasus uji → Data keluaran

Kasus uji dapat berupa jenis online atau batch:

- Kasus uji online adalah kasus uji di mana pengguna menjalankan dialog layar interaktif (3270) untuk membaca, memodifikasi, atau menghasilkan data bisnis baru (database dan/atau catatan dataset).
- Kasus uji Batch adalah kasus uji yang memerlukan pengiriman batch untuk membaca, memproses, dan memodifikasi atau menghasilkan data bisnis baru (kumpulan data dan/atau catatan basis data).

Skenario uji

Skenario pengujian adalah serangkaian kasus uji yang dijalankan dalam urutan berurutan, satu per satu. Putar ulang dilakukan pada tingkat skenario pengujian. Semua kasus pengujian dalam skenario pengujian dijalankan di lingkungan pengujian target saat skenario pengujian diputar ulang. Jika ada perbedaan setelah membandingkan artefak pengujian referensi dan replay, perbedaan akan ditampilkan pada tingkat kasus uji.

Misalnya, Skenario Uji A:

Test Case 1, Test Case 2, Test Case 3, dan sebagainya.

Proyek uji

Proyek pengujian mewakili kumpulan skenario pengujian untuk mencapai tonggak pengujian yang diinginkan. Misalnya, migrasi aplikasi tertentu dapat dianggap sebagai proyek uji tunggal. Mengelompokkan skenario pengujian ke dalam proyek pengujian memungkinkan manajer pengujian untuk melacak status proyek pengujian, termasuk pengujian yang lulus/gagal.

Kondisi awal

Kondisi awal berisi sekumpulan sumber daya (komputasi, datastore, dan lainnya) yang harus Anda buat, dan data aplikasi yang harus Anda pulihkan pada sumber daya yang dibuat sebelum Anda dapat menjalankan skenario pengujian. Ini menciptakan baseline lingkungan pengujian target. Hal ini memungkinkan Anda untuk menyediakan AWS CloudFormation template. Anda menggunakan template untuk membuat lingkungan pengujian target dan, secara opsional, ekstrak DDL dari database sumber jika skenario pengujian Anda memodifikasi catatan database. Setiap skenario pengujian akan dikaitkan dengan kondisi awal. Kondisi awal dapat dikaitkan dengan beberapa skenario pengujian. Untuk memastikan pengulangan dengan konsistensi hasil dan untuk menghindari positif palsu karena data yang sudah berubah, Anda harus memulihkan kondisi awal sebelum setiap skenario pengujian dijalankan.

Untuk skenario pengujian yang berisi kasus uji yang memodifikasi catatan database, kondisi awal juga mereferensikan ekspor DDL dari skema dan tabel database sumber.

Rekam (tangkap)

Catatan dilakukan pada tingkat skenario pengujian. Selama pencatatan, Anda harus menyediakan lokasi Amazon S3 yang berisi artefak, kumpulan data, dan jurnal CDC database relasional dari mainframe sumber untuk dibandingkan. Ini akan dianggap sebagai data referensi dari mainframe sumber. Selama pemutaran ulang, data replay yang dihasilkan akan dibandingkan dengan data referensi yang direkam untuk memastikan kesetaraan aplikasi.

Memutar ulang

Pemutaran ulang dilakukan pada tingkat skenario pengujian. Selama pemutaran ulang, Pengujian Aplikasi Modernisasi AWS Mainframe menggunakan CloudFormation skrip yang direferensikan dalam kondisi awal terkait untuk membuat lingkungan pengujian target dan menjalankan aplikasi. Dataset dan catatan database yang dimodifikasi selama pemutaran ulang ditangkap dan dibandingkan dengan data referensi dari mainframe. Biasanya, Anda akan merekam pada mainframe sekali dan kemudian memutar ulang beberapa kali, sampai kesetaraan fungsional telah tercapai.

Bandingkan

Perbandingan dilakukan secara otomatis setelah pemutaran ulang selesai dengan sukses. Selama perbandingan, data yang direferensikan yang Anda unggah dan tangkap selama fase perekaman dibandingkan dengan data pemutaran ulang yang dihasilkan selama fase pemutaran ulang. Perbandingan terjadi pada tingkat kasus uji individu untuk kumpulan data, catatan basis data, dan layar online secara terpisah.

Perbandingan basis data

Pengujian Aplikasi menggunakan fungsionalitas pencocokan state-progress saat membandingkan perubahan dalam catatan database antara aplikasi sumber dan target. Pencocokan state-progress membandingkan perbedaan di setiap individu menjalankan pernyataan INSERT, UPDATE, dan DELETE, tidak seperti membandingkan baris tabel di akhir proses. Pencocokan kemajuan negara lebih efisien daripada alternatif, memberikan perbandingan yang lebih cepat dan lebih akurat dengan hanya membandingkan data yang diubah dan mendeteksi kesalahan koreksi diri dalam aliran transaksi. Dengan menggunakan teknologi CDC (Changed Data Capture), Pengujian Aplikasi dapat mendeteksi perubahan database relasi individu dan membandingkannya antara sumber dan target.

Perubahan database relasi dihasilkan pada sumber dan target oleh kode aplikasi yang diuji menggunakan pernyataan DHTML (Data Modification Language) seperti SQL INSERT, UPDATE, atau DELETE, tetapi juga secara tidak langsung ketika aplikasi menggunakan prosedur tersimpan, atau ketika pemicu database diatur pada beberapa tabel, atau ketika CASCADE DELETE digunakan untuk menjamin integritas referensial, memicu penghapusan tambahan secara otomatis.

Perbandingan dataset

Pengujian Aplikasi secara otomatis membandingkan kumpulan data referensi dan pemutaran ulang yang dihasilkan pada sistem sumber (perekaman) dan pemutaran ulang target).

Untuk membandingkan kumpulan data:

1. Mulailah dengan data input yang sama (dataset, database) pada sumber dan target.
2. Jalankan kasus pengujian Anda pada sistem sumber (mainframe).
3. Tangkap kumpulan data yang dihasilkan dan unggah ke bucket Amazon S3. Anda dapat mentransfer kumpulan data input dari sumber ke AWS menggunakan jurnal, layar, dan kumpulan data CDC.

4. Tentukan lokasi bucket Amazon S3 tempat kumpulan data mainframe diunggah saat Anda merekam kasus uji.

Setelah pemutaran ulang selesai, Pengujian Aplikasi secara otomatis membandingkan referensi output dan kumpulan data target, menunjukkan apakah catatan identik, setara, berbeda, atau hilang. Misalnya, bidang tanggal yang relatif terhadap momen eksekusi beban kerja (hari + 1, akhir bulan berjalan, dll.) Secara otomatis dianggap setara. Selain itu, Anda dapat secara opsional menentukan aturan kesetaraan, sehingga catatan yang tidak identik masih memiliki arti bisnis yang sama, dan ditandai sebagai setara.

Status perbandingan

Pengujian Aplikasi menggunakan status perbandingan berikut: IDENTIK, SETARA, dan BERBEDA.

IDENTIK

Sumber dan data target persis sama.

SETARA

Sumber dan data target mengandung perbedaan palsu yang dianggap sebagai kesetaraan, seperti tanggal atau stempel waktu yang tidak mempengaruhi kesetaraan fungsional ketika relatif terhadap momen eksekusi beban kerja. Anda dapat menentukan aturan kesetaraan untuk mengidentifikasi apa perbedaan ini. Ketika semua skenario pengujian yang diputar ulang dibandingkan dengan skenario pengujian referensinya menunjukkan status IDENTIK atau EQUIVALENT, skenario pengujian Anda membuktikan kesetaraan fungsional.

BERBEDA

Sumber dan data target berisi perbedaan, seperti jumlah catatan yang berbeda dalam kumpulan data, atau nilai yang berbeda dalam catatan yang sama.

Aturan kesetaraan

Seperangkat aturan untuk mengidentifikasi perbedaan palsu yang dapat dianggap sebagai hasil yang setara. Pengujian kesetaraan fungsional offline (OFET) pasti menyebabkan perbedaan untuk beberapa hasil antara sumber dan sistem target. Misalnya, stempel waktu pembaruan berbeda menurut desain. Aturan kesetaraan menjelaskan bagaimana menyesuaikan perbedaan tersebut dan menghindari positif palsu pada waktu perbandingan. Misalnya, jika tanggal adalah runtime + 2 hari

di kolom data tertentu, aturan kesetaraan menjelaskannya dan menerima waktu pada sistem target yang merupakan runtime pada target+2 hari alih-alih nilai yang sama secara ketat sama dengan kolom yang sama dalam rekaman referensi.

Perbandingan dataset keadaan akhir

Status akhir kumpulan data yang telah dibuat atau dimodifikasi, termasuk semua perubahan atau pembaruan yang dilakukan pada kumpulan data dari keadaan awalnya. Untuk kumpulan data, Pengujian Aplikasi melihat catatan dalam kumpulan data tersebut di akhir kasus uji, dan membandingkan hasilnya.

Perbandingan basis data kemajuan negara

Perbandingan perubahan yang dilakukan pada catatan database sebagai urutan pernyataan DML/Delete (Delete, Update, Insert) individu. Pengujian Aplikasi membandingkan perubahan individual (menyisipkan, memperbarui, atau menghapus baris tabel) dari database sumber ke database target, dan akan mengidentifikasi perbedaan untuk setiap perubahan individu. Misalnya, pernyataan INSERT individu dapat digunakan untuk menyisipkan dalam tabel baris dengan nilai yang berbeda pada database sumber dibandingkan dengan database target.

Kesetaraan fungsional (FE)

Dua sistem dianggap setara secara fungsional jika menghasilkan hasil yang sama pada semua operasi yang dapat diamati, mengingat data input yang sama. Misalnya, dua aplikasi dianggap setara secara fungsional jika data input yang sama menghasilkan data keluaran yang identik (melalui layar, perubahan dataset atau perubahan database).

Perbandingan layar 3270 online

Membandingkan output layar mainframe 3270 dengan output layar web aplikasi modern saat sistem target berjalan di bawah runtime AWS Blu Age di file. AWS Cloud Dan itu membandingkan output dari layar mainframe 3270 dengan layar 3270 dari aplikasi rehosted ketika sistem target berjalan di bawah runtime Micro Focus di. AWS Cloud

Merekam

Tindakan memulihkan keadaan data yang terkenal, kemudian menangkap, atau merekam data referensi dari skenario pengujian referensi (untuk satu atau beberapa kasus uji secara berurutan) pada sistem sumber.

Putar ulang data

Data replay digunakan untuk menggambarkan data yang dihasilkan dengan memutar ulang skenario pengujian pada lingkungan pengujian target. Misalnya, data replay dihasilkan ketika skenario pengujian berjalan pada aplikasi layanan Modernisasi AWS Mainframe. Data replay kemudian dibandingkan dengan data referensi yang diambil dari sumbernya. Setiap kali Anda memutar ulang beban kerja di lingkungan target, generasi baru data replay dihasilkan.

Data referensi

Data referensi digunakan untuk menggambarkan data yang diambil pada mainframe sumber. Ini adalah referensi di mana data yang dihasilkan replay (target) akan dibandingkan. Biasanya, untuk setiap catatan pada mainframe yang membuat data referensi, akan ada banyak tayangan ulang. Ini karena pengguna biasanya menangkap status aplikasi yang benar pada mainframe, dan memutar ulang kasus uji pada aplikasi target yang dimodernisasi untuk memvalidasi kesetaraan. Jika bug ditemukan, mereka diperbaiki dan kasus uji diputar ulang lagi. Seringkali, beberapa siklus pemutaran ulang, memperbaiki bug, dan memutar ulang lagi untuk memvalidasi kejadian. Ini disebut capture once, replay multiple times paradigma pengujian.

Rekam, Putar Ulang, dan Bandingkan

Pengujian Aplikasi beroperasi dalam tiga langkah:

- Rekam: menangkap data referensi yang dibuat pada mainframe untuk setiap kasus uji skenario pengujian. Ini dapat mencakup 3270 layar online, kumpulan data, dan catatan basis data.
  - Untuk layar 3270 online, Anda harus menggunakan emulator terminal Blu Insights untuk menangkap beban kerja sumber Anda. Untuk informasi lebih lanjut lihat, dokumentasi Blu Insights.
  - Untuk kumpulan data, Anda perlu menangkap kumpulan data yang dihasilkan oleh setiap kasus uji pada mainframe dengan menggunakan alat umum, seperti FTP atau bagian layanan transfer dataset dari AWS Modernisasi Mainframe.
  - Untuk perubahan basis data, Anda menggunakan dokumentasi AWS Mainframe Modernization Data Replication with Acceply untuk menangkap dan membuat jurnal CDC yang berisi perubahan.
- Putar ulang: Skenario pengujian diputar ulang di lingkungan target. Semua kasus uji yang ditentukan dalam skenario pengujian dijalankan. Tipe data tertentu yang dibuat oleh kasus uji individual, seperti kumpulan data, perubahan basis data relasional, atau layar 3270, akan

ditangkap dengan otomatisasi. Data ini dikenal sebagai data replay, dan akan dibandingkan dengan data referensi yang ditangkap selama fase pencatatan.

> **Note**
>
> Perubahan database relasional akan memerlukan opsi konfigurasi khusus DMS di templat kondisi awal Anda. CloudFormation

- Bandingkan: data referensi pengujian sumber, dan data pemutaran ulang target dibandingkan, dan hasilnya akan ditampilkan kepada Anda sebagai data yang identik, berbeda, setara, atau hilang.

Perbedaan

Menunjukkan perbedaan telah terdeteksi antara referensi dan replay dataset dengan perbandingan data. Misalnya, bidang di layar 3270 online yang menunjukkan nilai berbeda dari sudut pandang logika bisnis antara mainframe sumber dan aplikasi yang dimodernisasi target akan dianggap sebagai perbedaan. Contoh lain adalah catatan dalam kumpulan data yang tidak identik antara aplikasi sumber dan target.

Kesetaraan

Catatan ekivalen adalah catatan yang berbeda antara kumpulan data referensi dan pemutaran ulang, tetapi tidak boleh diperlakukan berbeda dari sudut pandang logika bisnis. Misalnya, catatan yang berisi stempel waktu kapan dataset diproduksi (waktu eksekusi beban kerja). Dengan menggunakan aturan kesetaraan yang dapat disesuaikan, Anda dapat menginstruksikan Pengujian Aplikasi untuk memperlakukan perbedaan positif palsu tersebut sebagai kesetaraan, bahkan jika itu menunjukkan nilai yang berbeda antara data referensi dan replay.

Aplikasi sumber

Aplikasi mainframe sumber untuk dibandingkan dengan.

Aplikasi target

Aplikasi baru atau yang dimodifikasi di mana pengujian dilakukan dan yang akan dibandingkan dengan aplikasi sumber untuk mendeteksi cacat dan untuk mencapai kesetaraan fungsional antara aplikasi sumber dan target. Aplikasi target biasanya berjalan di AWS Cloud.

Tutorial: Mengatur aplikasi CardDemo sampel

AWSPengujian Aplikasi dalam rilis pratinjau untuk Modernisasi AWS Mainframe dan dapat berubah sewaktu-waktu. Kami menyarankan Anda menggunakan fitur ini hanya dengan data pengujian dan aplikasi, dan bukan di lingkungan produksi.

Untuk tutorial ini, Anda membuat AWS CloudFormation tumpukan yang membantu Anda mengatur aplikasi CardDemo sampel untuk replatforming dengan Micro Focus pada layanan terkelola Modernisasi AWS Mainframe, dan fitur termasuk AWS Pengujian Aplikasi Modernisasi Mainframe. Tutorial ini menjelaskan contoh AWS CloudFormation template yang dapat Anda gunakan untuk membuat tumpukan. Kami juga menyediakan file zip dari artefak aplikasi yang diperlukan. Contoh template menyediakan database, lingkungan runtime, aplikasi, dan lingkungan jaringan yang sepenuhnya terisolasi.

Template ini menciptakan beberapa AWS sumber daya. Anda akan ditagih untuk mereka jika Anda membuat tumpukan dari template ini.

Prasyarat

- Unduh dan unzip `IC3-card-demo-zip` dan`datasets_Mainframe_ebcdic.zip`. File-file ini berisi CardDemo sampel dan kumpulan data sampel untuk digunakan dengan Pengujian AWS Aplikasi.
- Buat bucket Amazon S3 untuk menyimpan CardDemo file dan artefak lainnya. Sebagai contoh, `my-carddemo-bucket`.

Langkah 1: Bersiaplah untuk mengatur CardDemo

Unggah file CardDemo sampel dan edit AWS CloudFormation template yang akan membuat CardDemo aplikasi.

1. Unggah `datasets_Mainframe_ebcdic` dan `IC3-card-demo` folder yang Anda buka ritsleting sebelumnya ke bucket Anda.
2. Unduh `aws-m2-math-mf-carddemo.yaml` AWS CloudFormation template dari ember Anda. Itu ada di `IC3-card-demo` folder.
3. Edit `aws-m2-math-mf-carddemo.yaml` AWS CloudFormation template sebagai berikut:

- Ubah `BucketName` parameter menjadi nama bucket yang Anda tentukan sebelumnya, sepertimy`-carddemo-bucket`.
- Ubah `ImportJsonPath` ke lokasi di ember `mf-carddemo-datasets-import.json` file Anda. Misalnya, `s3://my-carddemo-bucket/IC3-card-demo/mf-carddemo-datasets-import.json` Memperbarui nilai ini memastikan bahwa output `M2ImportJson` memiliki nilai yang benar.
- (Opsional) Sesuaikan `EngineVersion` dan `InstanceType` parameter agar sesuai dengan standar Anda.

> **Note**
>
> Jangan memodifikasi `M2EnvironmentId` dan `M2ApplicationId` output. Pengujian Aplikasi menggunakan nilai-nilai tersebut untuk menemukan sumber daya yang akan berinteraksi dengannya.

Langkah 2: Buat semua sumber daya yang diperlukan

Jalankan AWS CloudFormation template khusus Anda untuk membuat semua sumber daya yang Anda butuhkan untuk menyelesaikan tutorial ini dengan sukses. Template ini mengatur CardDemo aplikasi sehingga Anda dapat menggunakannya dalam pengujian.

1. Masuk ke AWS CloudFormation konsol dan pilih Buat tumpukan, lalu pilih Dengan sumber daya baru (standar).
2. Dalam Prasyarat - Siapkan template, pilih Template sudah siap.
3. Di Tentukan templat, pilih Unggah file templat, lalu pilih Pilih file.
4. Arahkan ke tempat Anda mengunduh `aws-m2-math-mf-carddemo.yaml` dan memilih file itu, lalu pilih Berikutnya.
5. Di Tentukan detail tumpukan berikan nama untuk tumpukan sehingga Anda dapat dengan mudah menemukannya dalam daftar dan kemudian memilih Berikutnya.
6. Di Configure stack options, pertahankan nilai default dan pilih Next.
7. Di Tinjauan, periksa AWS CloudFormation apa yang dibuat untuk Anda, lalu pilih Kirim.

Dibutuhkan sekitar 10-15 menit AWS CloudFormation untuk membuat tumpukan.

> **Note**
>
> Template diatur untuk menambahkan akhiran unik ke nama sumber daya yang dibuatnya. Ini berarti Anda dapat membuat beberapa contoh template tumpukan ini secara paralel, fitur utama untuk Pengujian Aplikasi yang memungkinkan Anda menjalankan beberapa skenario pengujian secara bersamaan.

Langkah 3: Menyebarkan dan memulai aplikasi

Terapkan CardDemo aplikasi yang AWS CloudFormation dibuat untuk Anda dan pastikan itu berjalan.

1. Buka konsol Modernisasi AWS Mainframe dan pilih Aplikasi dari navigasi kiri.
2. Pilih CardDemo aplikasi, yang dinamai sesuatu seperti`aws-m2-math-mf-carddemo-abc1d2e3`.
3. Pilih Tindakan, lalu pilih Deploy aplikasi.
4. Di Lingkungan, pilih lingkungan runtime yang sesuai dengan aplikasi. Ini akan memiliki pengenal unik yang sama ditambahkan ke akhir nama. Sebagai contoh, `aws-m2-math-mf-carddemo-abc1d2e3`.
5. Pilih Deploy. Tunggu hingga aplikasi berhasil digunakan dan dalam keadaan Siap.
6. Pilih aplikasi, lalu pilih Actions and Start Application. Tunggu hingga aplikasi dalam status Running.
7. Di halaman detail aplikasi, salin Port dan DNS Hostname, yang Anda butuhkan untuk terhubung ke aplikasi yang sedang berjalan.

Langkah 4: Impor data awal

Untuk menggunakan aplikasi CardDemo sampel, Anda harus mengimpor kumpulan data awal. Selesaikan langkah-langkah berikut:

1. Unduh `mf-carddemo-datasets-import.json` filenya.
2. Edit file di editor teks pilihan Anda.
3. Temukan `s3Location` parameter dan perbarui nilainya untuk menunjuk ke bucket Amazon S3 yang Anda buat.

4. Buat perubahan yang sama untuk semua kejadian`s3Location`, lalu simpan file.
5. Masuk ke konsol Amazon S3 dan navigasikan ke bucket yang Anda buat sebelumnya.
6. Unggah `mf-carddemo-datasets-import.json` file yang disesuaikan.
7. Buka konsol Modernisasi AWS Mainframe dan pilih Aplikasi dari navigasi kiri.
8. Pilih CardDemo aplikasinya.
9. Pilih Kumpulan data dan kemudian pilih Impor.
10. Arahkan ke lokasi di Amazon S3 tempat Anda mengunggah file JSON yang disesuaikan dan pilih Kirim.

Pekerjaan ini mengimpor 23 kumpulan data. Untuk memantau hasil pekerjaan impor, periksa konsol. Ketika semua kumpulan data berhasil diimpor, sambungkan ke aplikasi.

> **Note**
>
> Bila Anda menggunakan template ini dalam Pengujian Aplikasi, Output `M2ImportJson` secara otomatis menangani proses impor.

Langkah 5: Connect ke CardDemo aplikasi

Connect ke aplikasi CardDemo sampel menggunakan emulator 3270 pilihan Anda.

- Saat aplikasi berjalan, gunakan emulator 3270 Anda untuk terhubung ke aplikasi, tentukan nama host DNS dan nama port, jika perlu.

  Misalnya, jika Anda menggunakan emulator open source c3270, perintah Anda terlihat seperti ini:

  ```
  c3270 -port port-number DNS-hostname
  ```

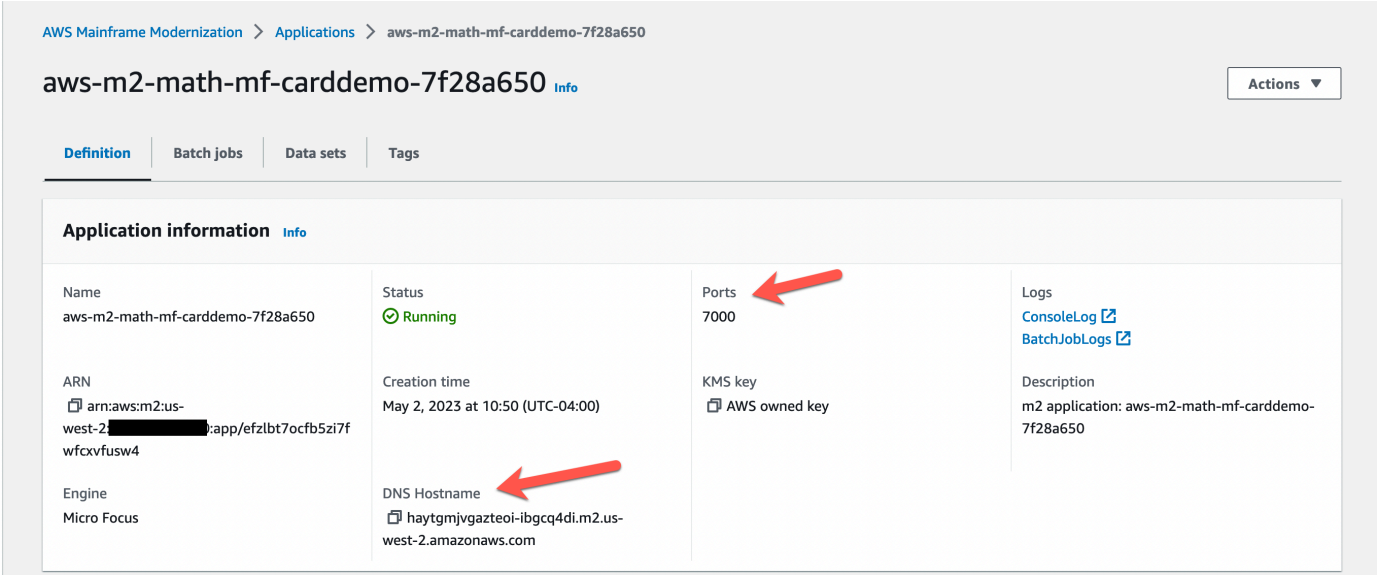
  port

  Port yang ditentukan pada halaman detail aplikasi. Misalnya, 6000.

Nama host

Nama Host DNS ditentukan pada halaman detail aplikasi.

Gambar berikut menunjukkan di mana menemukan port dan DSN Hostname.



Tutorial: Pengujian Aplikasi Modernisasi AWS Mainframe memutar ulang dan membandingkan penggunaan CardDemo untuk AWS Blu Age yang diterapkan di Amazon EC2

AWSPengujian Aplikasi dalam rilis pratinjau untuk Modernisasi AWS Mainframe dan dapat berubah sewaktu-waktu. Kami menyarankan Anda menggunakan fitur ini hanya dengan data pengujian dan aplikasi, dan bukan di lingkungan produksi.

Dalam tutorial ini, Anda akan menyelesaikan langkah-langkah yang diperlukan untuk memutar ulang dan membandingkan beban kerja pengujian dengan CardDemo aplikasi yang berjalan pada AWS Blu Age yang digunakan di Amazon EC2.

Langkah 1: Dapatkan Gambar Mesin Amazon EC2 Amazon AWS Blu Age (AMI)

Ikuti petunjuk dalam tutorial Pengaturan AWSAWSBlu Age Runtime (di Amazon EC2) untuk langkah-langkah orientasi yang diperlukan untuk mendapatkan akses AWS ke Blu Age di Amazon EC2 AMI.

Langkah 2: Mulai instans Amazon EC2 menggunakan AWS Blu Age AMI

1. Siapkan AWS kredensyal Anda.
2. Identifikasi lokasi file biner 3.5.0 Amazon EC2 AMI (hanya CLI/versi Blu AWS Age) dari bucket Amazon S3:

```
aws s3 ls s3://aws-bluage-runtime-artifacts-xxxxxxx-eu-west-1/
aws s3 ls s3://aws-bluage-runtime-artifacts-xxxxxxx-eu-west-1/3.5.0/AMI/
```

> **Note**
>
> Fitur Pengujian Aplikasi hanya tersedia untuk digunakan di 4 wilayah di prod (us-east-1, sa-east-1, eu-central-1 dan ap-southeast-2).

3. Kembalikan AMI di akun Anda dengan perintah berikut:

```
aws ec2 create-restore-image-task --object-key 3.5.0/AMI/ami-0182ffe3b9d63925b.bin
 --bucket aws-bluage-runtime-artifacts-xxxxxxx-eu-west-1 --region eu-west-1 --name
 "AWS BLUAGE RUNTIME AMI"
```

> **Note**
>
> Ganti nama file bin AMI dan Wilayah tempat Anda ingin membuat AMI.

4. Setelah membuat instans Amazon EC2, Anda dapat menemukan ID AMI yang benar yang dipulihkan AMI dari bucket Amazon S3 di katalog gambar Amazon EC2.

> **Note**
>
> Dalam tutorial ini, ID AMI adalah ami-0d0fafcc636fd1e6d, dan Anda harus mengubah ID ini di file konfigurasi yang berbeda dengan yang diberikan kepada Anda.

1. Jika aws ec2 create-restore-image-task gagal, periksa versi Python dan CLI Anda menggunakan perintah berikut:

```
aws --version
```

> **Note**
>
> Versi Python harus> = 3 dan versi CLI harus> = 2.

2. Jika versi ini sudah usang, CLI harus diperbarui. Untuk memperbarui CLI:
   a. Ikuti petunjuk di Instal atau perbarui AWS CLI versi terbaru.
   b. Hapus CLI v1 dengan perintah berikut:

```
sudo yum remove awscli
```

   c. Dan instal CLI v2 dengan perintah berikut:

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
 "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

   d. Terakhir, periksa versi Python dan CLI dengan perintah berikut:

```
aws --version
```

3. Anda kemudian dapat mengulang aws create-restore-image-task ec2.

Langkah 3: Unggah file CardDemo dependen ke S3

Salin konten folder database, file-system, dan userdata. Unduh dan unzip CardDemo aplikasi. Ketiga folder ini harus disalin ke salah satu bucket Anda yang disebut your-s3-bucket dalam dokumentasi ini.

Langkah 4: Muat database dan inisialisasi aplikasi CardDemo

Buat instans Amazon EC2 sementara yang akan Anda gunakan sebagai sumber daya komputasi untuk menghasilkan snapshot database yang diperlukan untuk aplikasi. CardDemo Instans EC2 ini

tidak akan menjalankan CardDemo aplikasi itu sendiri, melainkan menghasilkan snapshot database yang akan digunakan nanti.

Mulailah dengan mengedit CloudFormation template yang disediakan bernama 'load-and-create-ba-snapshots.yml'. Ini adalah CloudFormation template yang digunakan untuk membuat instans Amazon EC2 yang digunakan untuk menghasilkan snapshot database.

1. Buat dan berikan key pair EC2 Anda yang akan digunakan untuk instans EC2. Untuk informasi selengkapnya, lihat Membuat pasangan kunci.

   Contoh:

   ```
   Ec2KeyPair:
       Description: 'ec2 key pair'
       Default: 'm2-tests-us-west-2'
       Type: String
   ```

2. Tentukan jalur Amazon S3 dari folder Anda tempat Anda meletakkan folder database dari langkah sebelumnya:

   ```
   S3DBScriptsPath:
       Description: 'S3 DB scripts folder path'
       Type: String
       Default: 's3://your-s3-bucket/databases'
   ```

3. Tentukan jalur Amazon S3 dari folder Anda tempat Anda meletakkan folder sistem file dari langkah sebelumnya:

   ```
   S3ApplicationFilesPath:
       Description: 'S3 application files folder path'
       Type: String
       Default: 's3://your-s3-bucket/file-system'
   ```

4. Tentukan jalur Amazon S3 dari folder Anda tempat Anda meletakkan folder userdata dari langkah sebelumnya:

   ```
   S3UserDataPath:
       Description: 'S3 userdata folder path'
       Type: String
       Default: 's3://your-s3-bucket/userdata'
   ```

5. Juga tentukan jalur Amazon S3 tempat Anda akan menyimpan file hasil yang akan digunakan pada langkah berikutnya.

```
S3SaveProducedFilesPath:
    Description: 'S3 path folder to save produced files'
    Type: String
    Default: 's3://your-s3-bucket/post-produced-files'
```

6. Ubah ID AMI dengan yang benar diperoleh sebelumnya dalam tutorial ini menggunakan template berikut:

```
  BaaAmiId:
    Description: 'ami id (AL2) for ba anywhere'
    Default: 'ami-0bd41245734fd20d9'
    Type: String
```

- Anda dapat secara opsional mengubah nama tiga snapshot yang akan dibuat oleh menjalankan database beban. withCloudFormation Ini akan terlihat di CloudFormation tumpukan saat sedang dibuat dan akan digunakan nanti dalam tutorial ini. Ingatlah untuk mencatat nama yang digunakan untuk snapshot database.

```
SnapshotPrimary:
    Description: 'Snapshot Name DB BA Primary'
    Type: String
    Default: 'snapshot-primary'

  SnapshotBluesam:
    Description: 'Snapshot Name DB BA Bluesam'
    Type: String
    Default: 'snapshot-bluesam'

  SnapshotJics:
    Description: 'Snapshot Name DB BA Jics'
    Type: String
    Default: 'snapshot-jics'
```

> **Note**
>
> Dalam dokumen ini, kami berasumsi bahwa nama snapshot tetap konsisten.

7. Jalankan CloudFormation dengan CLI atau AWS konsol menggunakan tombol Create Stack dan wizard. Di akhir proses, Anda akan melihat tiga snapshot di konsol RDS dengan nama yang Anda pilih diikuti oleh ID unik. Anda akan membutuhkan nama-nama ini di langkah berikutnya.

   > **Note**
   >
   > RDS akan menambahkan postfix ke nama snapshot yang ditentukan dalam template. AWS CloudFormation Pastikan untuk mendapatkan nama snapshot lengkap dari RDS sebelum melanjutkan ke langkah berikutnya.

   Contoh perintah CLI-

   ```
   aws cloudformation create-stack --stack-name load-and-create-ba-snapshots --
   template-url https://your-apptest-bucket.s3.us-west-2.amazonaws.com/load-and-
   create-ba-snapshots.yml --capabilities CAPABILITY_NAMED_IAM
   ```

   Anda juga dapat memeriksa di jalur Amazon S3 yang Anda berikan untuk S3 SaveProducedFilesPath bahwa kumpulan data telah dibuat dengan benar.

Langkah 5: Luncurkan runtime AWS Blu Age CloudFormation

Gunakan CloudFormation untuk menjalankan instans Amazon EC2 dengan aplikasi CardDemo AWS Blu Age. Anda harus mengganti beberapa variabel dalam CloudFormation nama `m2-with-ba-using-snapshots-https-authentication.yml` dengan mengedit file YAMM atau dengan memodifikasi nilai di konsol selama peluncuran CFN.

1. Ubah `AllowedVpcEndpointPrincipals` untuk menentukan akun mana yang akan mencapai titik akhir VPC untuk mengakses runtime AWS Blu Age, menggunakan perintah berikut:

   ```
   AllowedVpcEndpointPrincipals:
       Description: 'comma-separated list of IAM users, IAM roles, or AWS accounts'
       Default: 'apptest.amazonaws.com'
       Type: String
   ```

2. Ubah nilai variabel SnapshotPrimaryDb, SnapshotBlusamDb, dan SnapshotJicsDb ke nama snapshot. Dapatkan juga nama snapshot dari RDS setelah dibuat pada langkah sebelumnya.

```
SnapshotPrimary:
    Description: 'Snapshot DB cluster for DB Primary'
    Type: String
    Default: 'snapshot-primary87d067b0'

  SnapshotBluesam:
    Description: 'Snapshot DB cluster for DB Bluesam'
    Type: String
    Default: 'snapshot-bluesam87d067b0'

  SnapshotJics:
    Description: 'Snapshot DB cluster for DB Jics'
    Type: String
    Default: 'snapshot-jics87d067b0'
```

> **Note**
>
> RDS akan menambahkan postfix sendiri ke nama snapshot.

3. Berikan key pair Amazon EC2 Anda untuk instans EC2, menggunakan perintah ini:

```
Ec2KeyPair:
    Description: 'ec2 key pair'
    Default: 'm2-tests-us-west-2'
    Type: String
```

4. Berikan ID AMI yang telah Anda peroleh selama proses registrasi AMI untuk variabel tersebut BaaAmiId, dengan menggunakan:

```
BaaAmiId:
    Description: 'ami id (AL2) for ba anywhere'
    Default: 'ami-0d0fafcc636fd1e6d'
    Type: String
```

5. Berikan jalur folder Amazon S3 yang Anda gunakan pada langkah sebelumnya untuk menyimpan file yang dihasilkan, menggunakan perintah berikut:

```
S3ApplicationFilesPath:
    Description: 'bucket name'
    Type: String
```

```
    Default: 's3://your-s3-bucket/post-produced-files'
```

6. Terakhir, berikan jalur folder s3-: userdata-folder-path

```
S3UserDataPath:
    Description: 'S3 userdata folder path'
    Type: String
    Default: 's3://your-s3-bucket/userdata'
```

- (Opsional) Anda dapat mengaktifkan mode HTTPS dan otentikasi HTTP dasar untuk tomcat. Meskipun pengaturan default juga akan berfungsi.

> **Note**
>
> Secara default, mode HTTPS dinonaktifkan dan diatur ke mode HTTP dalam parameter BacHttpsMode:

Sebagai contoh:

```
BacHttpsMode:
    Description: 'http or https for Blue Age Runtime connection mode '
    Default: 'http'
    Type: String
    AllowedValues: [http, https]
```

- (Opsional) Untuk mengaktifkan mode HTTPS, Anda harus mengubah nilainya menjadi HTTPS dan memberikan ARN sertifikat ACM Anda dengan mengubah nilai variabel ACM: CertArn

```
ACMCertArn:
    Type: String
    Description: 'ACM certificate ARN'
    Default: 'your arn certificate'
```

- (Opsional) Otentikasi dasar dinonaktifkan secara default dengan parameter WithBacBasicAuthenticationdisetel ke false. Anda dapat mengaktifkannya dengan menyetel nilai ke true.

```
WithBacBasicAuthentication:
    Description: 'false or true for Blue Age Runtime Basic Authentication '
```

```
    Default: false
    Type: String
    AllowedValues: [true, false]
```

7. Ketika Anda telah menyelesaikan konfigurasi, Anda dapat membuat tumpukan dengan menggunakan CloudFormation template yang diedit.

Langkah 6: Menguji instans Amazon EC2 AWS Blu Age

Jalankan CloudFormation template secara manual untuk membuat instance AWS Blu Age Amazon EC2 untuk CardDemo aplikasi untuk memastikan bahwa itu dimulai tanpa kesalahan. Hal ini dilakukan untuk memverifikasi bahwa CloudFormation template dan semua prasyarat valid, sebelum menggunakan CloudFormation template dengan fitur Pengujian Aplikasi. Anda kemudian dapat menggunakan Pengujian Aplikasi untuk secara otomatis membuat instans AWS Blu Age Amazon EC2 target selama pemutaran ulang dan membandingkan melalui kondisi awal.

1. Jalankan perintah CloudFormation create stack untuk membuat instance AWS Blu Age Amazon EC2, dengan menyediakan template m2 with-ba-using-snapshots - CloudFormation -https-authentication.yl yang Anda edit pada langkah sebelumnya:

```
aws cloudformation create-stack --stack-name load-and-create-ba-snapshots —-
template-url https://apptest-ba-demo.s3.us-west-2.amazonaws.com/m2-with-ba-using-
snapshots-https-authentication.yml --capabilities CAPABILITY_NAMED_IAM --region us-
west-2
```

> **Note**
>
> Ingatlah untuk menentukan Wilayah yang benar di mana AMI Zaman AWS Blu dipulihkan.

2. Pastikan semuanya berfungsi dengan benar dengan melihat di konsol untuk menemukan instans Amazon EC2 yang sedang berjalan. Connect ke sana menggunakan Session Manager.
3. Setelah Anda terhubung ke instans Amazon EC2, gunakan perintah berikut:

```
sudo su
cd /m2-anywhere/tomcat.gapwalk/velocity/logs
cat catalina.log
```

4. Pastikan tidak ada pengecualian atau kesalahan dalam log.

5. Selanjutnya, periksa apakah aplikasi merespons dengan menggunakan perintah ini:

```
curl http://localhost:8080/gapwalk-application/
```

Anda akan melihat pesan, “Aplikasi Jics sedang berjalan.”

Langkah 7: Validasi langkah-langkah sebelumnya telah diselesaikan dengan benar

Dalam beberapa langkah berikutnya, kita akan menggunakan Pengujian Aplikasi Modernisasi AWS Mainframe untuk memutar ulang dan membandingkan kumpulan data yang dibuat oleh aplikasi. CardDemo Langkah-langkah ini bergantung pada keberhasilan penyelesaian semua langkah sebelumnya dalam tutorial ini. Validasi hal-hal berikut sebelum melanjutkan:

1. Anda telah berhasil membuat instance AWS Blu Age di Amazon EC2 melalui AWS CloudFormation template.
2. Layanan Tomcat pada AWS Blu Age di Amazon EC2 aktif dan berjalan, tanpa pengecualian.

Saat Anda menjalankan instans EC2 dengan CardDemo aplikasi, selesaikan langkah-langkah berikut di konsol Pengujian Aplikasi untuk melakukan pemutaran ulang dan membandingkan kumpulan data batch.

Langkah 8. Buat kondisi awal

Pada langkah ini, Anda membuat kondisi awal dengan menyediakan CloudFormation template yang Anda gunakan untuk menyebarkan CardDemo aplikasi AWS Blu Age di Amazon EC2.

1. Buka konsol Modernisasi AWS Mainframe di https://console.aws.amazon.com/m2/.
2. Di panel navigasi kiri, pilih Pengujian Aplikasi.
3. Dalam Pengujian Aplikasi, pilih Buat Kondisi Awal.
4. Gunakan salinan lokal Anda dengan jalur Amazon S3 yang dimodifikasi dan ID snapshot yang mengarah ke sumber daya Anda.

Langkah 9: Buat kasus uji

Pada langkah ini, Anda membuat test case yang akan digunakan untuk membandingkan dataset yang dibuat dalam aplikasi Card Demo.

1. Buat kasus uji baru. Berikan nama dan deskripsi.
2. Tentukan `CREASTMT.JCL` sebagai nama JCL.
3. Tambahkan kumpulan data berikut ke definisi kasus Uji:

Nama	CCSID	RecordFormat	RecordLength
AWS.M2.CARDDEMO.STATEMNT.PS	“037"	FB	80
AWS.M2.CARDDEMO.STATEMNT.HTML	“037"	FB	100

> **Note**
>
> Nama JCL dan detail dataset Anda harus cocok.

Langkah 10: Buat skenario pengujian

1. Buat skenario pengujian baru, dan berikan nama dan deskripsi untuknya.
2. Tambahkan kasus uji yang Anda buat pada langkah sebelumnya ke skenario pengujian Anda.
3. Setelah skenario pengujian dibuat, pilih kondisi awal yang dibuat pada langkah 1 di halaman ikhtisar skenario pengujian.

Langkah 11: Rekam skenario pengujian Anda

Pada langkah ini, Anda menjalankan kasus uji pada sumber. Untuk melakukan itu:

1. Download dan jalankan dataset yang berasal dari mainframe run aplikasi. CardDemo

2. Unggah folder yang tidak di-zip ke bucket Amazon S3 Anda. Bucket Amazon S3 ini harus berada di Wilayah yang sama dengan sumber daya Pengujian Aplikasi Anda yang lain.

> **Note**
>
> Harus ada dua file dengan nama yang cocok dengan nama kumpulan data yang diteruskan dalam kasus uji sebelumnya.

3. Pada halaman ikhtisar skenario Uji, pilih tombol Rekam.
4. Pada halaman catatan skenario Uji, tentukan lokasi Amazon S3 tempat Anda mengunggah kumpulan data yang diperoleh dari mainframe sumber.
5. Klik Kirim untuk memulai proses pencatatan.

> **Note**
>
> Tunggu hingga rekaman selesai sebelum Anda melakukan pemutaran ulang dan membandingkan.

Langkah 12: Putar ulang dan bandingkan

Jalankan skenario pengujian dan kasus uji di lingkungan AWS AWS Blu Age target di Amazon EC2. Pengujian Aplikasi akan menangkap kumpulan data yang dihasilkan replay, dan membandingkannya dengan kumpulan data referensi yang direkam pada mainframe.

1. Pilih Putar Ulang dan Bandingkan. Diperlukan waktu sekitar tiga menit untuk membuat CloudFormation tumpukan, melakukan perbandingan, dan menghapus tumpukan.

Setelah semuanya selesai, Anda harus memiliki hasil perbandingan dengan beberapa perbedaan yang sengaja dibuat untuk tujuan demo ini.

AWS Modernisasi Mainframe Pengujian Aplikasi set data yang didukung halaman kode

AWS Pengujian Aplikasi dalam rilis pratinjau untuk Modernisasi AWS Mainframe dan dapat berubah sewaktu-waktu. Kami menyarankan Anda menggunakan fitur ini hanya dengan data pengujian dan aplikasi, dan bukan di lingkungan produksi.

Gunakan tabel berikut untuk menentukan apakah pengenal set karakter berkode (CCSID) untuk data Anda didukung pada AWS Pengujian Aplikasi. Jika data Anda menggunakan CCSID yang tidak didukung, kami sarankan Anda mengonversinya menjadi CCSID yang didukung atau hubungi kami untuk mendapatkan bantuan.

CCSID	Set karakter	Deskripsi
37	IBM037, IBM-037, Cp037	Tuan rumah: AS, Kanada (ESA), Belanda, Portugal, Brasil, Australia, Selandia Baru
273	IBM273, IBM-273, Cp273	Tuan rumah: Austria, Jerman
277	IBM277, IBM-277, Cp277	Tuan rumah: Denmark, Norwegia
278	IBM278, IBM-278, Cp278	Tuan rumah: Finlandia, Swedia
280	IBM280, IBM-280, Cp280	Tuan rumah: Italia
284	IBM284, IBM-284, Cp284	Tuan rumah: Spanyol, Amerika Latin (Spanyol)
285	IBM285, IBM-285, Cp285	Tuan rumah: Inggris
297	IBM297, IBM-297, Cp297	Tuan rumah: Prancis
300	IBM-300	JEPANG DB EBCDIC

CCSID	Set karakter	Deskripsi
301	IBM-301	Data PC: Jepang DB
437	IBM437, IBM-437, ASCII ASCII, ASCII, Cp437, ASCII ASCII	Data PC: PC Base USA, banyak negara lain
500	IBM500, IBM-500, Cp500	Tuan rumah: Belgia, Kanada (AS/400), Swiss, Internasional Latin-1
720	IBM-720	MSDOS ARAB
737	IBM-737, x-IBM737	MSDOS YUNANI
775	IBM775, IBM-775	MSDOS BALTIK
808	IBM-808	Data PC: Cyrillic, Rusia, dengan euro
813	ISO-8859-7, ISO8859_7	ISO 8859-7: Yunani
819	ISO-8859-1, ISO8859_1	ISO 8859-1: Negara-negara Latin-1
833	IBM-833	EBCDIC KOREA
834	IBM-834, X-IBM834	DB KOREA EBCDIC
835	IBM-835	T-CHINA DB EBCD
836	IBM-836	S-CHINA EBCDIC
837	IBM-837	S-CHINA EBCDIC
850	IBM850, IBM-850, Cp850	Data PC: Negara-negara Latin-1
855	IBM855, IBM-855, Cp855	Data PC: Sirilik

CCSID	Set karakter	Deskripsi
856	IBM-856, X-ibm856, Cp856	Data PC: Ibrani
858	IBM00858, IBM-858, Cp858	Data PC: Negara-negara Latin-1, dengan euro
859	IBM-859	Data PC: LATIN-9
860	IBM860, IBM-860	Data PC: Portugis
861	IBM861, IBM-861	Data PC: Islandia
862	IBM862, IBM-862, Cp862	Data PC: Ibrani (migrasi)
863	IBM863, IBM-863	Data PC: Kanada
865	IBM865, IBM-865, Cp865	Data PC: Den/Norwegia
866	IBM866, IBM-866, Cp866	Data PC: Sirilik, Rusia
867	IBM-867	Data PC: Ibrani dengan euro
870	IBM870, IBM-870, Cp870	Tuan rumah: Latin-2 multibahasa
871	IBM871, IBM-871, Cp871	Tuan rumah: Islandia
874	x-IBM874	Data PC: Thailand
875	IBM-875, X-ibm875, Cp875	Tuan rumah: Yunani
897	IBM-897	Data PC: Jepang SB
912	ISO-8859-2, ISO8859_2	ISO 8859-2: Bahasa Latin-2 multibahasa
915	ISO-8859-5, ISO8859_5	ISO 8859-5: Sirilik
916	ISO-8859-8, ISO8859_8	ISO 8859-8: Ibrani
918	IBM918, IBM-918, Cp918	Tuan rumah: Urdu

CCSID	Set karakter	Deskripsi
920	ISO-8859-9, ISO8859_9	ISO 8859-9: Latin-5 (ECMA-128, Turki TS-5881)
921	IBM-921, X-ibm921, Cp921	Data PC: Latvia, Lituania
922	IBM-922, X-ibm922, Cp922	Data PC: Estonia
923	ISO-8859-15, Cp923, ISO8859_15_FDIS	ISO 8859-15: Latin-9
924	IBM-924	ISO 8859-15: Latin-9
927	IBM-927	Data PC: T-Chinese
930	IBM-930, X-ibm930, Cp930	Host Katakana: SBCS diperpanjang. Kanji Host: DBCS termasuk 4370 karakter yang ditentukan pengguna
932	IBM-932	Data PC: Campuran Jepang
933	IBM-933, X-ibm933, Cp933	Tuan rumah: SBCS yang Diperpanjang. Host: DBCS termasuk 1880 karakter yang ditentukan pengguna dan 11172 karakter Hangul penuh
935	IBM-935, X-ibm935, Cp935	Tuan rumah: SBCS yang Diperpanjang. Host: DBCS termasuk 1880 karakter yang ditentukan pengguna.
937	IBM-937, X-ibm937, Cp937	Tuan rumah: SBCS yang Diperpanjang. Host: DBCS termasuk 6204 karakter yang ditentukan pengguna

CCSID	Set karakter	Deskripsi
939	IBM-939, X-ibm939, Cp939	Tuan Rumah Latin: SBCS yang diperluas. Kanji Host: DBCS termasuk 4370 karakter yang ditentukan pengguna.
942	IBM-942, IBM-942C, X-IBM942, X-ibm942c, Cp942, Cp942C	Data PC: SBCS yang Diperpanjang. Data PC: DBCS termasuk 1880 karakter yang ditentukan pengguna
943	IBM-943, IBM-943C, Shift_JIS, jendela-31j, jendela-932, x-IBM943, X-IBM943C, Cp943, Cp943C, MS932	Data PC: SBCS. Data PC: DBCS untuk lingkungan Terbuka termasuk 1880 IBMkarakter yang ditentukan pengguna
947	IBM-947	T-CHINESE BESAR-5
948	IBM-948, X-ibm948, Cp948	Data PC: SBCS yang Diperpanjang. Data PC: DBCS termasuk 6204 karakter yang ditentukan pengguna
949	IBM-949, IBM-949C, X-IBM949, X-IBM949C, Cp949, Cp949C	Kode IBM KS - Data PC: SBCS. Kode IBM KS - Data PC: DBCS termasuk 1880 karakter yang ditentukan pengguna
950	Big5, IBM-950, X-ibm950, Cp950	Data PC: SBCS (IBM BIG5). Data PC: DBCS termasuk 13493 SSP, 566 IBM dipilih, 6204 karakter yang ditentukan pengguna
951	IBM-951	Data PC: IBM KS

CCSID	Set karakter	Deskripsi
954	EUC-JP, IBM-954, IBM-954C	G0: JIS X201 Romawi. G1: JIS X208-1990. G1: JIS X201 Katakana. G1: JIS X212
964	EUC-TW, IBM-964, X-IBM964, Cp964	G0: ASCII. G1: CNS 11643 pesawat 1. G1: CNS 11643 pesawat 2.
970	EUC-KR, X-IBM970, Cp970	G0: ASCII. G1: KSC X5601-1989 termasuk 1880 karakter yang ditentukan pengguna
971	IBM-971	EUC KOREA
1006	IBM-1006, X-IBM1006, Cp1006	ISO-8: Urdu
1025	IBM-1025, X-ibm1025, Cp1025	Tuan rumah: Sirilik multibahasa
1026	IBM1026, IBM-1026, Cp1026	Tuan rumah: Latin-5 (Turki)
1027	IBM-1027	JEPANG LATIN EBCD
1041	IBM-1041	Data PC: Jepang
1043	IBM-1043	Data PC: T-Chinese
1046	IBM-1046, IBM-1046S, X-IBM1046	ARAB - PC
1047	IBM1047, IBM-1047	Tuan rumah: Latin-1
1051	hp-roman8	EMULASI HP
1088	IBM-1088	Data PC: Korea KS

CCSID	Set karakter	Deskripsi
1089	ISO-8859-6, ISO8859_6	ISO 8859-6: Bahasa Arab
1097	IBM-1097, X-ibm1097, Cp1097	Tuan rumah: Farsi
1098	IBM-1098, X-ibm1098, Cp1098	Data PC: Farsi
1112	IBM-1112, X-ibm1112, Cp1112	Tuan rumah: Latvia, Lituania
1114	IBM-1114	Data PC: T-CH SB
1115	IBM-1115	Data PC: S-CH GB
1122	IBM-1122, x-IBM1122, Cp1122	Tuan rumah: Estonia
1123	IBM-1123, x-IBM1123, Cp1123	Tuan rumah: Sirilik Ukraina
1124	IBM-1124, x-IBM1124, Cp1124	8-bit: Sirilik, Belarus
1140	IBM01140, IBM-1140, Cp1140	Tuan rumah: AS, Kanada (ESA), Belanda, Portugal, Brasil, Australia, Selandia Baru, dengan euro
1141	IBM01141, IBM-1141, Cp1141	Tuan rumah: Austria, Jerman, dengan euro
1142	IBM01142, IBM-1142, Cp1142	Tuan rumah: Denmark, Norwegia, dengan euro
1143	IBM01143, IBM-1143, Cp1143	Tuan rumah: Finlandia, Swedia, dengan euro

CCSID	Set karakter	Deskripsi
1144	IBM01144, IBM-1144, Cp1144	Tuan rumah: Italia, dengan euro
1145	IBM01145, IBM-1145, Cp1145	Tuan rumah: Spanyol, Amerika Latin (Spanyol), dengan euro
1146	IBM01146, IBM-1146, Cp1146	Tuan rumah: Inggris Raya, dengan euro
1147	IBM01147, IBM-1147, Cp1147	Tuan rumah: Prancis, dengan euro
1148	IBM01148, IBM-1148, Cp1148	Tuan rumah: Belgia, Kanada (AS/400), Swiss, Internasional Latin-1, dengan euro
1149	IBM01149, IBM-1149, Cp1149	Tuan rumah: Islandia, dengan euro
1200	UTF-16BE	Unicode dengan set karakter 65535. Dengan tidak adanya tanda urutan byte (BOM), diasumsikan UTF-16 BE (big-endian).
1202	UTF-16LE	UTF-16 LE dengan IBM PUA
1204	UTF-16	UTF-16 dengan IBM PUA
1208	UTF-8, UTF-8J, UTF8	Unicode dengan set karakter 65535. UTF-8.
1232	UTF-32BE	UTF-32 BE dengan IBM PUA
1234	UTF-32LE	UTF-32 LE dengan IBM PUA
1236	UTF-32	UTF-32 dengan IBM PUA

CCSID	Set karakter	Deskripsi
1351	IBM-1351	JEPANG TERBUKA
1362	IBM-1362	MS-WIN KOREA
1363	IBM-1363, IBM-1363C, jendela-949, MS949	Data PC: MS Windows SBCS Korea. Data PC: MS Windows Quran DBCS termasuk 11172 Hangul penuh
1364	IBM-1364	Tuan rumah: SBCS yang Diperpanjang. Host: DBCS termasuk 1880 karakter yang ditentukan pengguna dan 11172 karakter Hangul penuh
1370	IBM-1370	Data PC: SBCS yang diperluas, dengan euro. Data PC: DBCS termasuk 6204 karakter yang ditentukan pengguna, dengan euro
1371	IBM-1371	Tuan rumah: SBCS yang diperpanjang, dengan euro. Host: DBCS termasuk 6204 karakter yang ditentukan pengguna, dengan euro
1375	Big5-HKSCS	Campuran Big-5 Ext untuk HKSCS
1380	IBM-1380	Data PC: S-CH GB

CCSID	Set karakter	Deskripsi
1381	IBM-1381, x-IBM1381, Cp1381	Data PC: SBCS yang Diperpanjang (IBM GB). Data PC: DBCS (IBM GB) termasuk 31 karakter yang dipilih IBM, 1880 yang ditentukan pengguna
1382	IBM-1382	S-CHINA EUC
1383	EUC-CN, GB2312, IBM-1383, X-IBM1383, Cp1383	G0: ASCII. G1: GB 2312-80 set
1385	IBM-1385	Data PC: GBK S-CH
1386	GBK, IBM-1386, jendela-936, MS936	Data PC: S-Chinese GBK dan T-Chinese IBM BIG-5. Data PC: GBK S-China
1388	IBM-1388	Tuan rumah: SBCS yang Diperpanjang. Host: DBCS termasuk 1880 karakter yang ditentukan pengguna
1390	IBM-1390	Katakana Host: SBCS diperpanjang, dengan euro. Kanji Host: DBCS termasuk 6205 karakter yang ditentukan pengguna
1399	IBM-1399	Tuan Rumah Latin: SBCS diperpanjang, dengan euro. Kanji Host: DBCS termasuk 4370 karakter yang ditentukan pengguna, dengan euro

CCSID	Set karakter	Deskripsi
5050	JIS0201, JIS0208, JIS0212, JIS0201, JIS0208, JIS0212	G0: JIS X201 Romawi. G1: JIS X208-1990. G1: JIS X201 Katakana. G1: JIS X212
5054	ISO-2022-JP	TCP JEPANG
5346	jendela-1250, Cp1250	MS Windows: Latin-2, versi 2 dengan euro
5347	jendela-1251, Cp1251	MS Windows: Cyrillic, versi 2 dengan euro
5348	jendela-1252, Cp1252	MS Windows: Negara-negara Latin-1, versi 2 dengan euro
5349	jendela-1253, Cp1253	MS Windows: Yunani, versi 2 dengan euro
5350	jendela-1254, Cp1254	MS Windows: Turki, versi 2 dengan euro
5351	jendela-1255, Cp1255	MS Windows: Ibrani, versi 2 dengan euro
5352	jendela-1256, Windows-1 256s, Cp1256	MS Windows: Arab, versi 2 dengan euro
5353	jendela-1257, Cp1257	MS Windows: Baltic Rim, versi 2 dengan euro
5354	jendela-1258, Cp1258	MS Windows: Vietnam, versi 2 dengan euro
5488	GB18030	GB18030, data 1-byte GB18030, data 2-byte GB18030, data 4-byte

CCSID	Set karakter	Deskripsi
9030	IBM-838, Cp838	Tuan rumah: SBCS Thailand yang diperpanjang
9066	IBM-874, Cp874	Data PC: SBCS Thailand yang diperluas
9400	CESU-8	CESU-8 dengan IBM PUA
25546	ISO-2022	TCP KOREA
33722	IBM-33722, IBM-33722C	IBMeUCJP

Transfer File dalam Modernisasi AWS Mainframe

AWS Mainframe Modernization File Transfer memungkinkan Anda mentransfer dan mengonversi set data mainframe ke Amazon S3 untuk kasus penggunaan modernisasi, migrasi, dan augmentasi mainframe.

Topik

Apa itu Transfer File Modernisasi AWS Mainframe?

Dengan AWS Mainframe Modernization File Transfer, Anda dapat mentransfer dan mengonversi kumpulan data dan file dengan layanan yang dikelola sepenuhnya untuk mempercepat dan menyederhanakan kasus penggunaan modernisasi, migrasi, dan augmentasi ke layanan Modernisasi Mainframe dan Amazon S3. AWS

Topik

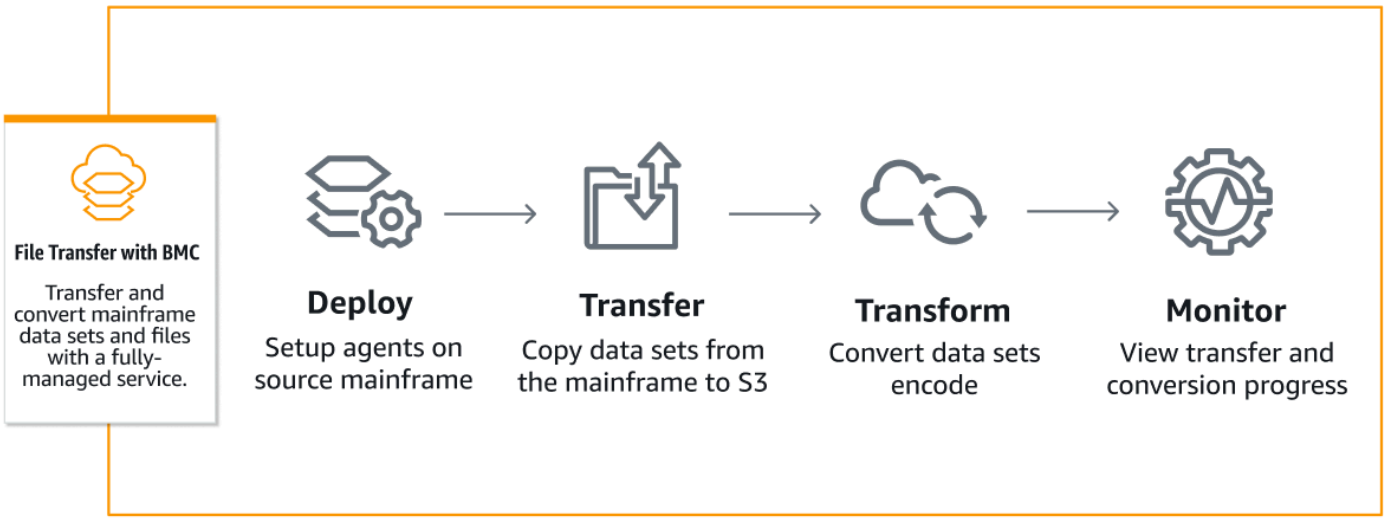
Manfaat Transfer File Modernisasi AWS Mainframe

AWS Mainframe Modernisasi File Transfer membantu Anda mentransfer kumpulan data dari mainframe ke Amazon S3. Beberapa manfaat meliputi:

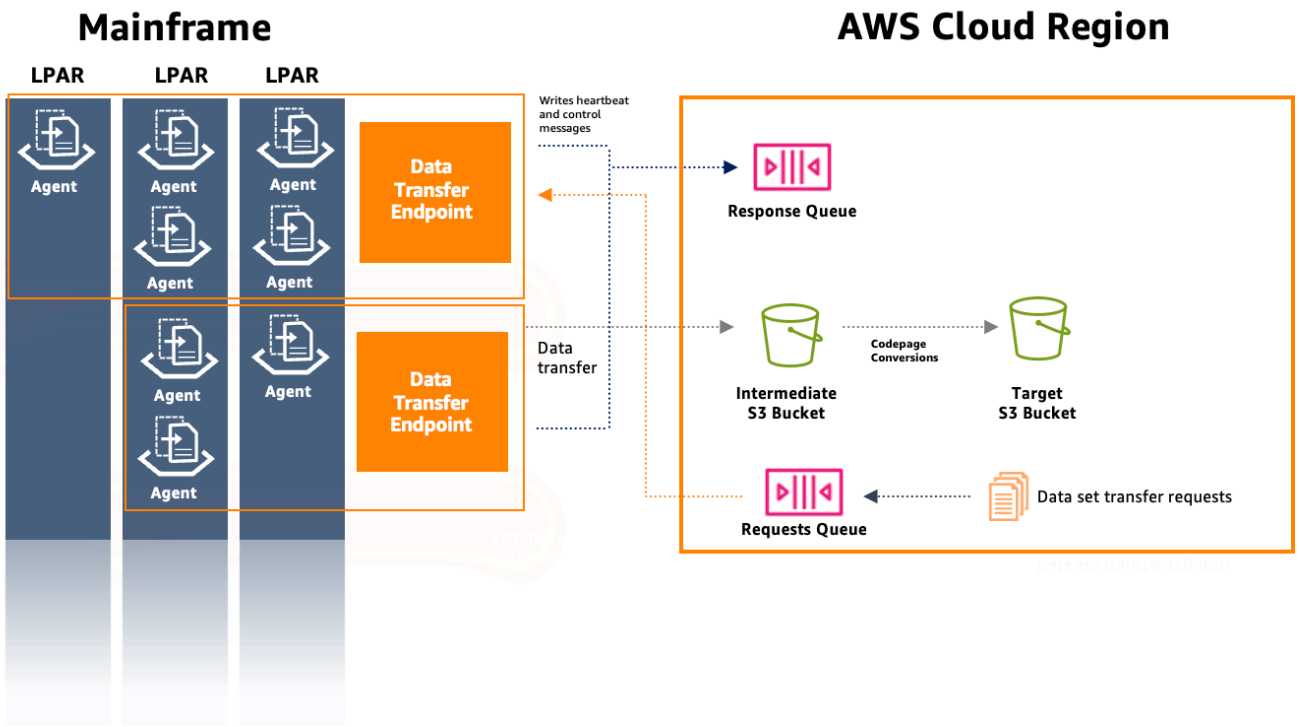
- Penemuan kumpulan data dan artefak mainframe sumber
- Transfer otomatis dan konversi dataset
- Skalabilitas, efisiensi, dan kecepatan untuk mencapai transfer dataset yang lebih cepat ke AWS

Cara kerja Transfer File Modernisasi AWS Mainframe

Gambar berikut adalah ikhtisar tentang cara kerja AWS Mainframe Modernization File Transfer pada tingkat konseptual.



Gambar berikut adalah ikhtisar arsitektur fitur AWS Mainframe Modernization File Transfer.



Instal agen Transfer File

Ikuti panduan ini step-by-step untuk menyelesaikan prasyarat untuk menginstal agen pada mainframe sumber dan untuk mengkonfigurasi agen.

Topik

Langkah 1: Masuk ke ISPF

Masuk ke sesi ISPF (Interactive System Productivity Facility) Anda. Ini biasanya dilakukan melalui emulator terminal 3270.

Langkah 2: Alokasikan kumpulan data untuk z/FS

Menggunakan utilitas dataset ISPF, alokasikan dataset baru untuk z/FS. Biasanya, ini dilakukan dalam utilitas daftar dataset di langkah 1.

1. Pergi ke opsi 3.4 (Utilities -> Dataset).

2. Tekan tombol 'C' untuk membuat dataset baru.
3. Masukkan nama dataset (misalnya, 'YourHLQ.m2Agent.zfs').
4. Tentukan jenis dataset sebagai 'Format besar' dengan ukuran primer 1000 silinder dan ukuran sekunder 200.
5. Tetapkan organisasi kumpulan data (DSORG) sebagai PS dan format rekam (RECFM) sebagai 'U' (Tidak ditentukan).
6. Selesaikan proses pembuatan.

Langkah 3: Format dataset sebagai z/FS

Setelah membuat kumpulan data, formatlah sebagai sistem file z/FS.

Salah satu caranya adalah dengan menggunakan Job Control Language (JCL) berikut ini:

```
//FORMAT  EXEC PGM=IOEAGFMT,PARM='AGGRNAME(yourhlq.M2AGENT.ZFS),FORMAT,AGGRSIZE(1200)'
//SYSPRINT DD  SYSOUT=A
```

Kirimkan pekerjaan ini dan periksa apakah berhasil diselesaikan.

Langkah 4: Tentukan sistem file ke z/OS

Tentukan z/FS ke z/OS menggunakan `DEF FILESYSTEM` perintah atau melalui pekerjaan batch. Gunakan perintah berikut ini.

```
DEF FILESYSTEM('yourhlq.M2AGENT.ZFS') TYPE(ZFS) MODE(R/W) MOUNTPOINT('/usr/lpp/aws/m2-
agent')
```

> **Note**
>
> Langkah ini membutuhkan otoritas tingkat sistem.

Langkah 5: Pasang sistem file

Untuk me-mount sistem file, gunakan perintah. `MOUNT` Anda dapat memasang sistem file di baris perintah di ISPF atau dalam batch.

Sebagai contoh:

```
MOUNT FILESYSTEM('yourhlq.M2AGENT.ZFS') TYPE(ZFS) MODE(RDWR) MOUNTPOINT('/usr/lpp/aws/
m2-agent')
```

Langkah 6: Verifikasi mount

Verifikasi bahwa sistem file dipasang dengan benar menggunakan `D OMVS,A` perintah atau dengan memeriksa dalam Unix System Service (USS).

Langkah 7: Masukkan OMV

Gunakan perintah berikut untuk memasukkan OMV:

```
TSO OMVS
```

Langkah 8: Mengatur variabel lingkungan direktori instalasi agen

Gunakan perintah berikut untuk mengatur lingkungan direktori instalasi agen:

```
export AGENT_DIR=/usr/lpp/aws/m2-agent
```

Langkah 9: Mengatur variabel lingkungan direktori kerja

Gunakan perintah berikut untuk mengatur variabel lingkungan direktori kerja:

```
export WORK_DIR=$AGENT_DIR/tmp
```

Langkah 10: Buat direktori kerja

Gunakan perintah berikut untuk mengatur lingkungan direktori kerja:

```
mkdir -p $WORK_DIR
```

Langkah 11: Salin paket tar Modernisasi AWS Mainframe ke direktori kerja di z/OS

Saat menggunakan FTP atau metode transfer apa pun, pastikan file tar ditransfer dalam mode biner.

Langkah 12: Asumsikan pengguna root

Gunakan perintah berikut untuk mengasumsikan pengguna root:

```
su
```

Ikuti langkah-langkah ini untuk menyelesaikan instalasi agen:

> **Note**
>
> Anda harus mengasumsikan pengguna root sebelum melanjutkan dengan langkah-langkah ini.

1. Setel variabel lingkungan versi m2-agent ke versi yang sedang diinstal menggunakan perintah berikut:

   ```
   export M2_AGENT_VERSION=1.0.0
   ```

2. Ekstrak paket agen tar menggunakan perintah berikut:

   ```
   tar -xpf m2-agent-package-$M2_AGENT_VERSION.tar -C $AGENT_DIR
   ```

3. Buat tautan `current-version` simbolis ke direktori instalasi agen saat ini dengan perintah berikut:

   ```
   ln -s $AGENT_DIR/m2-agent-v$M2_AGENT_VERSION $AGENT_DIR/current-version
   ```

4. Perbarui dan kirimkan `CPY#PDS` untuk membuat kumpulan data agen Transfer File.

   > **Note**
   >
   > JCL menggunakan SYS2.AWS.M2 HLQ.

   Untuk membuat agen Transfer File, atur baris parameter 000006-000012. Juga, perbarui tiga variabel simbolik`HLQ`,`VOLSER`, dan `AGNTPATH` untuk digunakan nanti di JCL:

   ```
   oedit $AGENT_DIR/current-version/installation/CPY#PDS
   ```

```
submit $AGENT_DIR/current-version/installation/CPY#PDS
```

> **Note**
>
> JCL ini dirancang untuk menyiapkan aspek-aspek tertentu dari instalasi agen pada mainframe. Ini mengalokasikan kumpulan data yang diperlukan dan kemudian menyalin file tertentu dari sistem file Unix ke kumpulan data ini.

Konfigurasikan izin dan STC

1. Perbarui dan kirimkan salah satu `SYS2.AWS.M2.SAMPLIB(SEC#RACF)` (untuk menyiapkan izin RACF) atau `SYS2.AWS.M2.SAMPLIB(SEC#TSS)` (untuk menyiapkan izin TSS) sesuai dengan instruksi mereka. Anggota ini diciptakan oleh `CPY#PDS` langkah sebelumnya.
2. Perbarui ekspor PWD di `SYS2.AWS.M2.SAMPLIB(M2AGENT)` STC JCL, jika jalur direktori agen Transfer File default () `/usr/lpp/aws/m2-agent` diubah.
3. Perbarui dan salin `SYS2.AWS.M2.SAMPLIB(M2AGENT)` JCL ke`SYS1.PROCLIB`.
4. Tambahkan `SYS2.AWS.M2.LOADLIB` ke daftar APF menggunakan perintah berikut:

   ```
   SETPROG APF ADD DSNAME(SYS2.AWS.M2.LOADLIB) SMS
   ```

5. Setel grup dan pemilik agen `logs` dan `diag` folder ke pengguna/grup agen (M2USER/M2GROUP). Gunakan perintah berikut ini.

   ```
   chown -R M2USER:M2GROUP $AGENT_DIR/current-version/logs
   chown -R M2USER:M2GROUP $AGENT_DIR/current-version/diag
   ```

Buat pengguna IAM dengan kredensyal akses jangka panjang

Anda perlu membuat agen mainframe yang diperlukan oleh pengguna IAM untuk menggunakan respons dan antrian permintaan dan menyimpan kumpulan data ke bucket Amazon S3.

Saat membuat pengguna ini:

1. Pilih Lampirkan kebijakan secara langsung di opsi Izin.

2. Setelah pengguna dibuat, buka tab Security credentials, dan buat kunci akses. Untuk informasi selengkapnya tentang membuat kunci akses IAM, lihat Mengelola kunci akses untuk pengguna IAM.
3. Di bagian tombol Akses, pilih Lainnya saat diminta untuk Kasus penggunaan.

> **Note**
>
> Simpan tombol Akses dan kunci akses Rahasia yang ditampilkan di halaman terakhir panduan pembuatan kunci akses, sebelum memilih Selesai. Tombol ini digunakan untuk mengkonfigurasi agen mainframe.

> **Note**
>
> Simpan ARN pengguna IAM yang digunakan untuk mengatur hubungan kepercayaan dengan peran IAM.

Buat peran IAM untuk diasumsikan oleh agen

Anda membuat peran IAM baru dengan kebijakan kepercayaan khusus untuk jenis entitas Tepercaya. Kebijakan ini akan menggunakan template berikut:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DataTransferEndpointAgentSqsReceive",
            "Effect": "Allow",
            "Action": [
                "sqs:DeleteMessage",
                "sqs:ReceiveMessage"
            ],
            "Resource": "<data-transfer-endpoint-request-queue-arn>"
        },
        {
            "Sid": "DataTransferEndpointS3",
            "Effect": "Allow",
            "Action": "s3:PutObject",
```

```
            "Resource": "<data-transfer-endpoint-intermediate-bucket-arn>/*"
        },
        {
            "Sid": "DataTransferEndpointAgentSqsSend",
            "Effect": "Allow",
            "Action": "sqs:SendMessage",
            "Resource": "<data-transfer-endpoint-response-queue-arn>"
        },
        {
            "Sid": "DataTransferEndpointAgentKmsDecrypt",
            "Effect": "Allow",
            "Action": "kms:Decrypt",
            "Resource": "<kms-key-id>"
        }
    ]
}
```

Di mana:

- `request-queue-arn`dan `response-queue-arn` merupakan ARN dari antrean permintaan Amazon SQS yang dibuat selama inisialisasi titik akhir transfer data.
- `transfer-bucket-arn`adalah ARN dari ember transfer yang dibuat sebelumnya.

> **Note**
>
> Anda dapat mencari semua nilai ini menggunakan konsol AWS.

> **Note**
>
> Simpan nama peran, yang akan Anda gunakan nanti untuk mengonfigurasi agen mainframe.

Konfigurasi agen

Untuk mengkonfigurasi agen Transfer File:

1. Navigasi ke `$AGENT_DIR/current-version/config`.

2. Edit file konfigurasi agen `appication.properties` untuk menambahkan konfigurasi lingkungan menggunakan perintah berikut:

```
oedit $AGENT_DIR/current-version/config/application.properties
```

Sebagai contoh:

```
agent.environments[0].account-id=<AWS_ACCOUNT_ID>
agent.environments[0].agent-role-name=<AWS_IAM_ROLE_NAME>
agent.environments[0].access-key-id=<AWS_IAM_ROLE_ACCESS_KEY>
agent.environments[0].secret-access-id=<AWS_IAM_ROLE_SECRET_KEY>
agent.environments[0].bucket-name=<AWS_S3_BUCKET_NAME>
agent.environments[0].environment-name=<AWS_REGION>
agent.environments[0].region=<AWS_REGION>
```

Di mana:

- `AWS_ACCOUNT_ID`adalah ID dari akun pelanggan.
- `AWS_IAM_ROLE_NAME`adalah nama peran IAM yang dibuat di. the section called “Buat peran IAM untuk diasumsikan oleh agen”
- `AWS_IAM_ROLE_ACCESS_KEY`adalah kunci akses pengguna IAM yang dibuat dithe section called “Buat pengguna IAM dengan kredensyal akses jangka panjang”.
- `AWS_IAM_ROLE_SECRET_KEY`adalah kunci rahasia akses untuk pengguna IAM yang dibuat dithe section called “Buat pengguna IAM dengan kredensyal akses jangka panjang”.
- `AWS_S3_BUCKET_NAME`adalah nama bucket transfer yang dibuat dengan titik akhir transfer data.
- `AWS_REGION`adalah wilayah di mana Anda mengkonfigurasi agen Transfer File.

> **Note**
>
> Mungkin ada beberapa bagian seperti itu, selama indeks dalam tanda kurung — `[0]` — bertambah untuk masing-masing.

Anda harus me-restart agen agar perubahan diterapkan.

Persyaratan

1. Ketika parameter ditambahkan atau dihapus, agen harus dihentikan dan dimulai. Mulai agen transfer File menggunakan perintah berikut di CLI:

   ```
   /S M2AGENT
   ```

   Untuk menghentikan agen M2, gunakan perintah berikut di CLI:

   ```
   /P M2AGENT
   ```

2. Anda dapat meminta transfer agen Transfer File ke beberapa wilayah dan akun AWS dengan mendefinisikan beberapa lingkungan.

   ```
   #Region 1
   agent.environments[0].account-id=AWS_ACCOUNT_ID
   agent.environments[0].agent-role-name=AWS_IAM_ROLE_NAME
   agent.environments[0].access-key-id=AWS_IAM_ROLE_ACCESS_KEY
   agent.environments[0].secret-access-id=AWS_IAM_ROLE_SECRET_KEY
   agent.environments[0].bucket-name=AWS_S3_BUCKET_NAME
   agent.environments[0].environment-name=AWS_REGION
   agent.environments[0].region=AWS_REGION

   #Region 2
   agent.environments[1].account-id=AWS_ACCOUNT_ID
   agent.environments[1].agent-role-name=AWS_IAM_ROLE_NAME
   agent.environments[1].access-key-id=AWS_IAM_ROLE_ACCESS_KEY
   agent.environments[1].secret-access-id=AWS_IAM_ROLE_SECRET_KEY
   agent.environments[1].bucket-name=AWS_S3_BUCKET_NAME
   agent.environments[1].environment-name=AWS_REGION
   agent.environments[1].region=AWS_REGION
   ```

Titik akhir transfer data

Titik akhir transfer data memungkinkan ketersediaan tinggi, skalabilitas, dan manajemen agen yang efisien pada mainframe sumber. Agen individu diinstal pada LPAR mainframe dan dapat dikelompokkan bersama ke dalam titik akhir transfer data. Ketika permintaan dibuat untuk mentransfer dataset, satu agen di titik akhir transfer data akan menangani transfer. Untuk memulai transfer data, setidaknya satu agen pada titik akhir transfer data harus online.

Prosedur ini mengasumsikan bahwa Anda telah menyelesaikan langkah-langkah Menyiapkan AWS Modernisasi Mainframe dan Konfigurasi agen Transfer File pada mainframe sumber.

Buat titik akhir transfer data

Untuk membuat titik akhir transfer data untuk Transfer File, Anda harus mengikuti langkah-langkah ini di konsol Modernisasi AWS Mainframe.

Untuk membuat endpoint transfer data

1. Buka konsol Modernisasi AWS Mainframe di https://console.aws.amazon.com/m2/.
2. Di Wilayah AWS pemilih, pilih wilayah tempat Anda ingin mentransfer file dari mainframe ke bucket Amazon S3.
3. Pada halaman Endpoint transfer data, di bawah Transfer File, pilih Buat titik akhir transfer data.
4. Pada halaman prasyarat titik akhir transfer data, baca semua instruksi untuk memastikan Anda telah menyelesaikan langkah-langkah ini. Setelah dikonfirmasi, pilih Berikutnya.
5. Pada halaman Konfigurasi titik akhir transfer data, tambahkan informasi dasar untuk titik akhir transfer data Anda.

    1. Di bagian informasi dasar, masukkan nama titik akhir transfer data, deskripsi, dan kunci KMS Anda. Untuk informasi selengkapnya tentang kunci KMS, lihat Buat kunci.

    > **Note**
    >
    > Nama titik akhir transfer data harus sesuai dengan nama yang ditentukan saat mengonfigurasi agen transfer file pada mainframe sumber.

    > **Note**
    >
    > Anda harus menambahkan kebijakan berbasis sumber daya berikut untuk KMS sehingga layanan Modernisasi AWS Mainframe dapat membaca dan menggunakan kunci ini untuk enkripsi/dekripsi:

```
{
  "Sid" : "Enable AWS M2 Permissions",
  "Effect" : "Allow",
  "Principal" : {
```

```
      "Service" : "m2.amazonaws.com"
            },
   "Action" : [
       "kms:Encrypt",
       "kms:Decrypt"
             ],
   "Resource" : "*"
}
```

2. Tentukan lokasi S3 untuk data perantara, yang merupakan lokasi S3 perantara tempat kumpulan data yang ditransfer dari mainframe disimpan.

> **Note**
>
> Disarankan agar Anda membuat bucket Amazon S3 baru untuk tugas transfer Anda. Untuk informasi tambahan, lihat Membuat bucket. Anda juga dapat menelusuri bucket Amazon S3 yang ada dengan memilih opsi Browse S3.

3. Setelah memasukkan bidang wajib, pilih Berikutnya.

6. Pada halaman Tinjau dan buat titik akhir transfer data, periksa apakah Anda telah menyelesaikan prasyarat, dan tinjau informasi dasar. Setelah dikonfirmasi, pilih Buat dan sambungkan.
7. Pada halaman Periksa konektivitas, Anda akan melihat semua agen ditampilkan dengan ID dan detak jantung mereka setelah konektivitas agen dibuat. Setelah konektivitas dibuat, Anda mendapatkan pesan “Titik akhir transfer data berhasil dibuat.”

> **Note**
>
> Jika Anda melihat pesan “Konektivitas agen gagal dibuat”, kunjungi kembali langkah 4 untuk memastikan bahwa Anda telah menyelesaikan semua prasyarat yang diperlukan.

8. Pilih Selesai.

Anda akan diarahkan ke halaman ikhtisar titik akhir transfer data di mana Anda dapat melihat daftar semua titik akhir transfer data. Anda juga akan dapat melihat titik akhir transfer data yang tersedia atau gagal.

Anda juga dapat mencari titik akhir transfer data berdasarkan nama dan mengakses informasi tambahan untuk setiap agen yang tersedia.

Transfer tugas

Tugas transfer digunakan untuk menentukan pengkodean sumber dan pengkodean target dari kumpulan data. Pengkodean sumber adalah format kumpulan data sumber, dan pengkodean target adalah format kumpulan data ini akan disimpan di bucket Amazon S3 target. Bucket target ini ditentukan oleh tugas transfer.

Prosedur ini mengasumsikan bahwa Anda telah menyelesaikan langkah-langkah Menyiapkan AWS Modernisasi Mainframe dan mengaturnyathe section called “Titik akhir transfer data”.

Topik

- Buat tugas transfer
- Lihat tugas transfer

Buat tugas transfer

Untuk membuat tugas transfer untuk Transfer File, Anda harus mengikuti langkah-langkah ini di konsol Modernisasi AWS Mainframe.

Untuk membuat tugas transfer

> **Note**
>
> Anda harus memiliki setidaknya satu titik akhir transfer data untuk membuat tugas transfer baru.

1. Buka konsol Modernisasi AWS Mainframe di https://console.aws.amazon.com/m2/.
2. Di Wilayah AWS pemilih, pilih Wilayah tempat Anda ingin mentransfer file dari mainframe ke bucket Amazon S3.
3. Pada halaman Transfer tugas, di bawah Transfer file, pilih titik akhir transfer data untuk membuat tugas transfer.
4. Jika titik akhir transfer data Anda tidak memiliki tugas transfer, Anda dapat membuat tugas baru dengan memilih Buat tugas transfer.

5. Pada halaman Buat tugas transfer, atur properti untuk tugas transfer.

   - Di halaman ini, masukkan informasi dasar tugas transfer Anda termasuk nama tugas transfer, deskripsi, kunci rahasia, dan kriteria pencarian kumpulan data.

   > **Note**
   >
   > - Enkripsi rahasia menggunakan kunci KMS yang ditentukan dengan titik akhir transfer data. Rahasianya juga harus berisi kredenal mainframe yang diperlukan untuk mengakses dataset pada mainframe menggunakan kunci and. `userId` `password` Untuk informasi selengkapnya, lihat Rahasia AWS Secrets Manager.
   > - Anda harus mengkonfigurasi kunci rahasia dengan kebijakan berbasis sumber daya berikut sehingga layanan Modernisasi AWS Mainframe dapat mengaksesnya untuk melakukan tugas transfer data:
   >
   > ```
   > {
   >   "Version" : "2012-10-17",
   >   "Statement" : [ {
   >     "Effect" : "Allow",
   >     "Principal" : {
   >         "Service" : "m2.amazonaws.com"
   >     },
   >     "Action" : [ "secretsmanager:GetSecretValue",
   >                  "secretsmanager:DescribeSecret" ],
   >     "Resource" : "*"
   >   } ]
   > }
   > ```

   > **Note**
   >
   > Ukuran dataset maksimum yang didukung saat ini adalah 90 GiB.

   - Masuk atau telusuri lokasi bucket Amazon S3 target untuk file Anda.
   - Titik akhir transfer data default akan dipilih. Anda juga dapat memilih untuk mengubah titik akhir dari titik akhir yang tersedia.

6. Pilih Berikutnya.

7. Pada halaman Pilih kumpulan data, Anda harus memperbarui pengkodean sumber dan pengkodean target secara manual untuk setiap kumpulan data yang Anda pilih. Pengkodean sumber adalah format kumpulan data sumber dan pengkodean target adalah format kumpulan data target, dan digunakan untuk mengonversi kumpulan data.
8. Setelah mengonfirmasi pengkodean sumber dan target, pilih Berikutnya.
9. Pada halaman Tinjau dan buat, Anda dapat meninjau atau mengedit informasi untuk tugas transfer Anda.
10. Pilih Buat tugas transfer.

Anda melihat pesan “Transfer tugas berhasil dibuat.”

Lihat tugas transfer

Untuk melihat tugas transfer untuk Transfer File, Anda harus mengikuti langkah-langkah ini di konsol Modernisasi AWS Mainframe.

Untuk melihat tugas transfer

1. Buka konsol Modernisasi AWS Mainframe di https://console.aws.amazon.com/m2/.
2. Di Wilayah AWS pemilih, pilih Wilayah tempat Anda ingin mentransfer file dari mainframe ke bucket Amazon S3.
3. Pada halaman Transfer tugas, di bawah Transfer file, pilih titik akhir transfer data untuk melihat tugas transfer Anda.
4. Untuk titik akhir yang memiliki tugas transfer yang sudah ada sebelumnya, ini akan terisi di bawah bagian Tugas Transfer. Anda dapat memilih untuk melihat detail tugas transfer apa pun dari daftar ini.

Tutorial: Memulai dengan AWS Mainframe Modernization File Transfer

AWS Mainframe Modernization File Transfer memungkinkan Anda mentransfer dan mengonversi kumpulan data mainframe untuk kasus penggunaan modernisasi, migrasi, dan augmentasi mainframe.

Ikuti langkah-langkah dalam tutorial ini untuk memahami cara kerja AWS Mainframe Modernization File Transfer.

Ikhtisar

Transfer File terdiri dari yang berikut:

1. Agen yang akan diinstal pada mainframe sumber.
2. Akses ke penemuan, transfer, dan kemampuan konversi kumpulan data langsung dari konsol layanan manajemen Modernisasi AWS Mainframe.

Sebagai pengguna, Anda dapat mentransfer kumpulan data dari mainframe ke bucket Amazon S3 Anda.

Topik

Langkah 1: Transfer paket tar binari agen dari AWS ke partisi logis mainframe

Unduh file tar dari tautan tar M2-Agent.

Langkah 2: Konfigurasikan agen Transfer File pada mainframe sumber

Pada langkah ini, Anda mengonfigurasi dan memulai agen AWS Mainframe Modernization File Transfer pada mainframe sumber. Agen diperlukan untuk memfasilitasi komunikasi antara fitur layanan Transfer File dan mainframe sumber. Setidaknya satu agen diperlukan per mainframe. Lebih dari satu agen dapat dimulai untuk ketersediaan tinggi dan skalabilitas yang ditingkatkan.

Ikuti petunjuk dalam the section called “Instal agen Transfer File” panduan untuk menyelesaikan instalasi agen Transfer File di mainframe.

Langkah 3: Buat titik akhir transfer data

Ikuti langkah-langkah di the section called “Titik akhir transfer data” halaman untuk membuat titik akhir transfer data baru.

Langkah 4: Buat tugas transfer

Ikuti langkah-langkah di the section called “Transfer tugas” halaman untuk membuat dan mengelola tugas transfer Anda.

Langkah 5: Lihat kemajuan tugas transfer

Anda dapat melihat kemajuan tugas transfer Anda di konsol Modernisasi AWS Mainframe. Untuk lebih jelasnya, lihat the section called “Lihat tugas transfer” bagian.

Keamanan dalam Modernisasi AWS Mainframe

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 untuk Modernisasi AWS Mainframe, lihat AWS Services in Scope by Compliance Program AWS Compliance Program.
- 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 Modernisasi AWS Mainframe. Ini menunjukkan kepada Anda cara mengkonfigurasi Modernisasi AWS Mainframe untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga mempelajari cara menggunakan AWS layanan lain yang membantu Anda memantau dan mengamankan sumber daya Modernisasi AWS Mainframe Anda.

AWSModernisasi Mainframe menyediakan sumber daya yang dilindungi IAM sendiri (aplikasi, lingkungan, penyebaran, dll), yang merupakan sumber daya administratif Modernisasi AWS Mainframe, di mana tindakan apa pun harus diizinkan oleh kebijakan IAM.

AWSModernisasi Mainframe untuk replatforming juga diamankan oleh IAM. IAM memberikan atau menolak izin kepada prinsipal untuk tindakan tertentu pada sumber daya yang ditentukan, yang berasal dari lingkungan mainframe asli, melalui kebijakan IAM standar juga. Runtime Modernisasi Modernisasi AWS Mainframe memanggil layanan otorisasi IAM ketika aplikasi mencoba tindakan tersebut pada sumber daya yang dilindungi. IAM akan mengembalikan allow atau deny berdasarkan mekanisme evaluasi kebijakan IAM standar.

Konten

Perlindungan data dalam Modernisasi AWS Mainframe

Model tanggung jawab AWS bersama model berlaku untuk perlindungan data dalam Modernisasi AWS Mainframe. Seperti yang dijelaskan dalam model AWS ini, bertanggung jawab untuk melindungi infrastruktur global yang menjalankan semua AWS Cloud. Anda bertanggung jawab untuk memelihara kendali atas isi yang dihost pada infrastruktur ini. Anda juga bertanggung jawab atas tugas-tugas konfigurasi dan manajemen keamanan untuk Layanan AWS yang Anda gunakan. Lihat informasi yang lebih lengkap tentang privasi data dalam Pertanyaan Umum Privasi Data. Lihat informasi tentang perlindungan data di Eropa di pos blog Model Tanggung Jawab Bersama dan GDPR AWS di Blog Keamanan AWS .

Untuk tujuan perlindungan data, kami menyarankan Anda melindungi Akun AWS kredensyal dan mengatur pengguna individu dengan AWS IAM Identity Center atau AWS Identity and Access Management (IAM). Dengan cara itu, setiap pengguna hanya diberi izin yang diperlukan untuk memenuhi tanggung jawab tugasnya. Kami juga menyarankan supaya Anda mengamankan data dengan cara-cara berikut:

- Gunakan autentikasi multi-faktor (MFA) pada setiap akun.
- Gunakan SSL/TLS untuk berkomunikasi dengan sumber daya. AWS Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Siapkan API dan pencatatan aktivitas pengguna dengan AWS CloudTrail.
- Gunakan solusi AWS enkripsi, bersama dengan semua kontrol keamanan default di dalamnya Layanan AWS.
- Gunakan layanan keamanan terkelola lanjut seperti Amazon Macie, yang membantu menemukan dan mengamankan data sensitif yang disimpan di Amazon S3.
- Jika Anda memerlukan modul kriptografi tervalidasi FIPS 140-2 saat mengakses AWS melalui antarmuka baris perintah atau API, gunakan titik akhir FIPS. Lihat informasi yang lebih lengkap tentang titik akhir FIPS yang tersedia di Standar Pemrosesan Informasi Federal (FIPS) 140-2.

Kami sangat merekomendasikan agar Anda tidak pernah memasukkan informasi identifikasi yang sensitif, seperti nomor rekening pelanggan Anda, ke dalam tanda atau bidang isian bebas seperti bidang Nama. Ini termasuk saat Anda bekerja dengan Modernisasi AWS Mainframe atau lainnya Layanan AWS menggunakan konsol, API AWS CLI, atau SDK. AWS Data apa pun yang Anda masukkan ke dalam tanda atau bidang isian bebas yang digunakan untuk nama dapat digunakan untuk log penagihan atau log diagnostik. Saat Anda memberikan URL ke server eksternal, kami sangat menganjurkan supaya Anda tidak menyertakan informasi kredensial di dalam URL untuk memvalidasi permintaan Anda ke server itu.

Data yang dikumpulkan oleh Modernisasi AWS Mainframe

AWS Modernisasi Mainframe mengumpulkan beberapa jenis data dari Anda:

- `Application configuration`: Ini adalah file JSON yang Anda buat untuk mengkonfigurasi aplikasi Anda. Ini berisi pilihan Anda untuk berbagai opsi yang ditawarkan Modernisasi AWS Mainframe. File ini juga berisi informasi untuk AWS sumber daya dependen seperti jalur Amazon Simple Storage Service tempat artefak aplikasi disimpan atau Amazon Resource Name (ARN) AWS Secrets Manager untuk tempat kredensyal database Anda disimpan.
- `Application executable (binary)`: Ini adalah biner yang Anda kompilasi dan yang ingin Anda terapkan pada Modernisasi AWS Mainframe.
- `Application JCL or scripts`: Kode sumber ini mengelola pekerjaan batch atau pemrosesan lainnya atas nama aplikasi Anda.
- `User application data`: Saat Anda mengimpor kumpulan data, Modernisasi AWS Mainframe menyimpannya dalam database relasional sehingga aplikasi Anda dapat mengaksesnya.
- `Application source code`Melalui Amazon AppStream 2.0, Modernisasi AWS Mainframe menyediakan lingkungan pengembangan bagi Anda untuk menulis dan mengkompilasi kode.

AWS Modernisasi Mainframe menyimpan data ini secara asli di. AWS Data yang kami kumpulkan dari Anda disimpan dalam bucket Amazon S3 yang dikelola Modernisasi AWS Mainframe. Saat Anda menerapkan aplikasi, Modernisasi AWS Mainframe mengunduh data ke instans Amazon Elastic Compute Compute Cloud yang didukung Amazon Elastic Block Store. Saat pembersihan dipicu, data dihapus dari volume Amazon EBS dan dari Amazon S3. Volume Amazon EBS adalah penyewa tunggal, artinya satu instance digunakan untuk satu pelanggan. Contoh tidak pernah dibagikan. Saat Anda menghapus lingkungan runtime, volume Amazon EBS juga akan dihapus. Saat Anda menghapus aplikasi, artefak dan konfigurasi dihapus dari Amazon S3.

Log aplikasi disimpan di Amazon CloudWatch. Pesan log aplikasi pelanggan juga diekspor ke CloudWatch . CloudWatch Log mungkin berisi data sensitif pelanggan, seperti data bisnis atau informasi keamanan dalam pesan debug). Untuk informasi selengkapnya, lihat Memantau Modernisasi AWS Mainframe dengan Amazon CloudWatch.

Selain itu, jika Anda memilih untuk melampirkan satu atau beberapa sistem file Amazon Elastic File System atau Amazon FSx ke lingkungan runtime Anda, data dalam sistem tersebut akan disimpan. AWS Anda perlu membersihkan data itu jika Anda memutuskan untuk berhenti menggunakan sistem file.

Anda dapat menggunakan semua opsi enkripsi Amazon S3 yang tersedia untuk mengamankan data saat menempatkannya di bucket AWS Amazon S3 yang digunakan Modernisasi Mainframe untuk penerapan aplikasi dan impor kumpulan data. Selain itu, Anda dapat menggunakan opsi enkripsi Amazon EFS dan Amazon FSx jika Anda melampirkan satu atau beberapa sistem file ini ke lingkungan runtime Anda.

Enkripsi data saat istirahat untuk layanan Modernisasi AWS Mainframe

AWS Modernisasi Mainframe terintegrasi dengan AWS Key Management Service menyediakan enkripsi sisi server transparan (SSE) pada semua sumber daya dependen yang menyimpan data secara permanen; yaitu Amazon Simple Storage Service, Amazon DynamoDB, dan Amazon Elastic Block Store. AWS Modernisasi Mainframe membuat dan mengelola AWS KMS kunci enkripsi simetris untuk Anda. AWS KMS

Enkripsi data saat istirahat secara default membantu mengurangi overhead operasional dan kompleksitas yang terlibat dalam melindungi data sensitif. Pada saat yang sama, ini memungkinkan Anda untuk memigrasikan aplikasi yang memerlukan kepatuhan enkripsi yang ketat dan persyaratan peraturan.

Anda tidak dapat menonaktifkan lapisan enkripsi ini atau memilih jenis enkripsi alternatif saat Anda membuat lingkungan dan aplikasi runtime.

Anda dapat menggunakan kunci terkelola pelanggan Anda sendiri untuk aplikasi Modernisasi AWS Mainframe dan lingkungan runtime untuk mengenkripsi sumber daya Amazon S3 dan Amazon EBS.

Untuk aplikasi Modernisasi AWS Mainframe Anda, Anda dapat menggunakan kunci ini untuk mengenkripsi definisi aplikasi Anda serta sumber daya aplikasi lainnya, seperti file JCL, yang disimpan di bucket Amazon S3 yang dibuat di akun layanan. Untuk informasi selengkapnya, lihat Membuat aplikasi .

Untuk lingkungan runtime Modernisasi AWS Mainframe Anda, Modernisasi AWS Mainframe menggunakan kunci terkelola pelanggan Anda untuk mengenkripsi volume Amazon EBS yang dibuat dan dilampirkan ke instance AWS Amazon EC2 Modernisasi Mainframe Anda, yang juga ada di akun layanan. Untuk informasi selengkapnya, lihat Buat lingkungan runtime.

> **Note**
>
> Sumber daya DynamoDB selalu dienkripsi menggunakan akun layanan Modernisasi Kunci yang dikelola AWS Mainframe. AWS Anda tidak dapat mengenkripsi sumber daya DynamoDB menggunakan kunci yang dikelola pelanggan.

AWS Modernisasi Mainframe menggunakan kunci terkelola pelanggan Anda untuk tugas-tugas berikut:

- Menerapkan kembali aplikasi.
- Mengganti instans Modernisasi AWS Mainframe Amazon EC2.

AWS Modernisasi Mainframe tidak menggunakan kunci yang dikelola pelanggan untuk mengenkripsi database Amazon Relational Database Service atau Amazon Aurora, antrian Layanan Antrian Sederhana Amazon, dan cache ElastiCache Amazon yang dibuat untuk AWS mendukung aplikasi Modernisasi Mainframe, karena tidak ada satupun yang berisi data pelanggan.

Untuk informasi selengkapnya, lihat Kunci terkelola pelanggan di Panduan AWS Key Management Service Pengembang.

Tabel berikut merangkum bagaimana Modernisasi AWS Mainframe mengenkripsi data sensitif Anda.

Tipe data	Kunci yang dikelola AWS enkripsi	Enkripsi kunci yang dikelola pelanggan
`Definition` Berisi definisi untuk aplikasi tertentu.	Aktif	Aktif
`EnvironmentSummary`	Aktif	Aktif

Tipe data	Kunci yang dikelola AWS enkripsi	Enkripsi kunci yang dikelola pelanggan
Berisi informasi tentang lingkungan runtime.		
`ApplicationSummary` Berisi informasi tentang aplikasi Modernisasi AWS Mainframe.	Aktif	Aktif
`DeploymentSummary` Berisi informasi tentang penyebaran aplikasi Modernisasi AWS Mainframe.	Aktif	Aktif

> **Note**
>
> AWS Modernisasi Mainframe secara otomatis memungkinkan enkripsi saat istirahat digunakan Kunci yang dikelola AWS untuk melindungi data sensitif Anda tanpa biaya. Namun, AWS KMS biaya berlaku untuk menggunakan kunci yang dikelola pelanggan. Untuk informasi selengkapnya tentang harga, lihat AWS Key Management Service Harga.

Untuk informasi lebih lanjut tentang AWS KMS, lihat AWS Key Management Service.

Bagaimana Modernisasi AWS Mainframe menggunakan hibah di AWS KMS

AWS Modernisasi Mainframe membutuhkan hibah untuk menggunakan kunci yang dikelola pelanggan Anda.

Saat Anda membuat aplikasi atau lingkungan runtime, atau menyebarkan aplikasi di Modernisasi AWS Mainframe yang dienkripsi dengan kunci yang dikelola pelanggan, Modernisasi AWS Mainframe membuat hibah atas nama Anda dengan mengirimkan permintaan ke. CreateGrant AWS KMS Hibah AWS KMS digunakan untuk memberikan akses Modernisasi AWS Mainframe ke kunci KMS di akun pelanggan.

AWS Modernisasi Mainframe memerlukan hibah untuk menggunakan kunci yang dikelola pelanggan Anda untuk operasi internal berikut:

- Kirim DescribeKeypermintaan AWS KMS untuk memverifikasi bahwa ID kunci terkelola pelanggan simetris yang dimasukkan saat membuat aplikasi, lingkungan runtime, atau penerapan aplikasi valid.
- Kirim GenerateDataKeypermintaan AWS KMS untuk mengenkripsi volume Amazon EBS yang dilampirkan ke instans Amazon EC2 yang AWS menghosting lingkungan runtime Modernisasi Mainframe.
- Kirim permintaan Dekripsi ke AWS KMS untuk mendekripsi konten terenkripsi di Amazon EBS.

AWS Modernisasi Mainframe menggunakan AWS KMS hibah untuk mendekripsi rahasia Anda yang disimpan di Secrets Manager dan saat membuat lingkungan runtime, membuat atau memindahkan aplikasi, dan membuat penerapan. Hibah yang dibuat oleh Modernisasi AWS Mainframe mendukung operasi berikut:

- Membuat atau memperbarui hibah lingkungan runtime:
  - Dekripsi
  - Enkripsi
  - ReEncryptFrom
  - ReEncryptTo
  - GenerateDataKey
  - DescribeKey
  - CreateGrant
- Membuat atau menerapkan kembali hibah aplikasi:
  - GenerateDataKey
- Buat hibah penerapan:
  - Dekripsi

Anda dapat mencabut akses ke hibah, atau menghapus akses layanan ke kunci yang dikelola pelanggan kapan saja. Jika Anda melakukannya, Modernisasi AWS Mainframe tidak akan dapat mengakses data apa pun yang dienkripsi oleh kunci yang dikelola pelanggan, yang memengaruhi operasi yang bergantung pada data. Misalnya, jika Modernisasi AWS Mainframe mencoba

mengakses definisi aplikasi yang dienkripsi oleh kunci yang dikelola pelanggan tanpa hibah untuk kunci itu, operasi pembuatan aplikasi akan gagal.

AWS Modernisasi Mainframe mengumpulkan konfigurasi aplikasi pengguna (file JSON) dan artefak (binari dan executable). Ini juga menciptakan metadata yang melacak berbagai entitas yang digunakan untuk pengoperasian Modernisasi AWS Mainframe, dan membuat log dan metrik. Log dan metrik yang terlihat pelanggan meliputi:

- CloudWatch log yang mencerminkan aplikasi dan mesin runtime (baik AWS Blu Age atau Micro Focus).
- CloudWatch metrik untuk dasbor operasi.

Selain itu, Modernisasi AWS Mainframe mengumpulkan data penggunaan dan metrik untuk pengukuran, pelaporan aktivitas, dan sebagainya tentang layanan. Data ini tidak terlihat pelanggan.

AWS Modernisasi Mainframe menyimpan data ini di tempat yang berbeda tergantung pada jenis data. Data pelanggan yang Anda unggah disimpan dalam bucket Amazon S3. Data layanan disimpan di Amazon S3 dan DynamoDB. Saat Anda menerapkan aplikasi, data dan data layanan Anda diunduh ke volume Amazon EBS. Jika Anda memilih untuk melampirkan Amazon EFS atau penyimpanan Amazon FSx ke lingkungan runtime Anda, data yang disimpan dalam sistem file tersebut juga diunduh ke volume Amazon EBS.

Enkripsi saat istirahat dikonfigurasi secara default. Anda tidak dapat menonaktifkannya atau mengubahnya. Saat ini, Anda juga tidak dapat mengubah konfigurasinya.

Buat kunci terkelola pelanggan

Anda dapat membuat kunci terkelola pelanggan simetris dengan menggunakan AWS Management Console atau AWS KMS API.

Untuk membuat kunci terkelola pelanggan simetris

Ikuti langkah-langkah untuk Membuat kunci terkelola pelanggan simetris di Panduan AWS Key Management Service Pengembang.

Kebijakan utama

Kebijakan utama mengontrol akses ke kunci yang dikelola pelanggan Anda. Setiap kunci yang dikelola pelanggan harus memiliki persis satu kebijakan utama, yang berisi pernyataan yang

menentukan siapa yang dapat menggunakan kunci dan bagaimana mereka dapat menggunakannya. Saat membuat kunci terkelola pelanggan, Anda dapat menentukan kebijakan kunci. Untuk informasi selengkapnya, lihat Mengelola akses ke kunci terkelola pelanggan di Panduan AWS Key Management Service Pengembang.

Untuk menggunakan kunci terkelola pelanggan Anda dengan sumber daya Modernisasi AWS Mainframe Anda, operasi API berikut harus diizinkan dalam kebijakan kunci:

- `kms:CreateGrant`— Menambahkan hibah ke kunci yang dikelola pelanggan. Memberikan akses kontrol ke kunci KMS tertentu, yang memungkinkan akses ke operasi hibah yang dibutuhkan Modernisasi AWS Mainframe. Untuk informasi selengkapnya tentang Menggunakan Hibah, lihat Panduan AWS Key Management Service Pengembang.

  Hal ini memungkinkan Modernisasi AWS Mainframe untuk melakukan hal berikut:

  - Panggilan `GenerateDataKey` untuk menghasilkan kunci data terenkripsi dan menyimpannya, karena kunci data tidak segera digunakan untuk mengenkripsi.
  - Panggilan `Decrypt` untuk menggunakan kunci data terenkripsi yang disimpan untuk mengakses data terenkripsi.
  - Siapkan kepala sekolah yang pensiun untuk memungkinkan layanan. `RetireGrant`
- `kms:DescribeKey`— Memberikan detail kunci yang dikelola pelanggan untuk memungkinkan Modernisasi AWS Mainframe memvalidasi kunci.

AWS Modernisasi Mainframe memerlukan `kms:CreateGrant` dan `kms:DescribeKey` izin dalam kebijakan utama pelanggan. AWS Modernisasi Mainframe menggunakan kebijakan ini untuk membuat hibah untuk dirinya sendiri.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "Enable IAM User Permissions",
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::AccountId:role/ExampleRole"
        },
        "Action": [
            "kms:CreateGrant",
            "kms:DescribeKey"
        ],
        "Resource": "*"
```

```
    }]
}
```

> **Note**
>
> Peran yang ditampilkan `Principal` dalam contoh sebelumnya adalah yang Anda gunakan untuk operasi Modernisasi AWS Mainframe seperti dan. `CreateApplication` `CreateEnvironment`

Untuk informasi selengkapnya tentang menentukan izin dalam kebijakan, lihat Panduan AWS Key Management Service Pengembang.

Untuk informasi selengkapnya tentang akses kunci pemecahan masalah, lihat Panduan AWS Key Management Service Pengembang.

Menentukan kunci yang dikelola pelanggan untuk Modernisasi AWS Mainframe

Anda dapat menentukan kunci yang dikelola pelanggan untuk sumber daya berikut:

- Aplikasi
- Environment

Saat Anda membuat sumber daya, Anda dapat menentukan kunci dengan memasukkan ID KMS, yang digunakan Modernisasi AWS Mainframe untuk mengenkripsi data sensitif yang disimpan oleh sumber daya.

- ID KMS — Pengidentifikasi kunci untuk kunci yang dikelola pelanggan. Masukkan ID kunci, ARN kunci, nama alias, atau ARN alias.

Anda dapat menentukan kunci yang dikelola pelanggan menggunakan AWS Management Console atau AWS CLI.

Untuk menentukan kunci terkelola pelanggan Anda saat membuat lingkungan runtime di AWS Management Console, lihatBuat lingkungan runtime Modernisasi AWS Mainframe. Untuk menentukan kunci terkelola pelanggan Anda saat membuat aplikasi di AWS Management Console, lihatBuat aplikasi Modernisasi AWS Mainframe.

Untuk menambahkan kunci terkelola pelanggan saat membuat lingkungan runtime dengan AWS CLI, tentukan `kms-key-id` parameternya, sebagai berikut:

```
aws m2 create-environment –engine-type microfocus –instance-type M2.m5.large
--publicly-accessible –engine-version 7.0.3 –name test
--high-availability-config desiredCapacity=2
--kms-key-id myEnvironmentKey
```

Untuk menambahkan kunci terkelola pelanggan Anda saat Anda membuat aplikasi dengan AWS CLI, tentukan `kms-key-id` parameternya, sebagai berikut:

```
aws m2 create-application –name test-application –description my description
--engine-type microfocus
--definition content="$(jq -c . raw-template.json | jq -R)"
--kms-key-id myApplicationKey
```

AWS Konteks enkripsi Modernisasi Mainframe

Konteks enkripsi adalah kumpulan opsional pasangan kunci-nilai yang berisi informasi kontekstual tambahan tentang data.

AWS KMS menggunakan konteks enkripsi sebagai data otentikasi tambahan untuk mendukung enkripsi yang diautentikasi. Bila Anda menyertakan konteks enkripsi dalam permintaan untuk mengenkripsi data, AWS KMS mengikat konteks enkripsi ke data terenkripsi. Untuk mendekripsi data, Anda menyertakan konteks enkripsi yang sama dalam permintaan.

AWS Konteks enkripsi Modernisasi Mainframe

AWS Modernisasi Mainframe menggunakan konteks enkripsi yang sama dalam semua operasi AWS KMS kriptografi yang terkait dengan aplikasi (membuat aplikasi dan membuat penyebaran), di mana kuncinya `aws:m2:app` dan nilainya adalah pengidentifikasi unik aplikasi.

Example

```
"encryptionContextSubset": {
        "aws:m2:app": "a1bc2defabc3defabc4defabcd"
}
```

Menggunakan konteks enkripsi untuk pemantauan

Bila Anda menggunakan kunci terkelola pelanggan simetris untuk mengenkripsi aplikasi atau lingkungan runtime, Anda juga dapat menggunakan konteks enkripsi dalam catatan audit dan log untuk mengidentifikasi bagaimana kunci yang dikelola pelanggan digunakan.

Menggunakan konteks enkripsi untuk mengontrol akses ke kunci terkelola pelanggan Anda

Anda dapat menggunakan konteks enkripsi dalam kebijakan utama dan kebijakan IAM `conditions` untuk mengontrol akses ke kunci terkelola pelanggan simetris Anda. Anda juga dapat menggunakan kendala konteks enkripsi dalam hibah.

AWS Modernisasi Mainframe menggunakan batasan konteks enkripsi dalam hibah untuk mengontrol akses ke kunci yang dikelola pelanggan di akun atau wilayah Anda. Batasan hibah mengharuskan operasi yang diizinkan oleh hibah menggunakan konteks enkripsi yang ditentukan. Contoh berikut adalah hibah yang memanfaatkan Modernisasi AWS Mainframe untuk mengenkripsi artefak aplikasi saat membuat aplikasi.

```
//This grant is retired immediately after create application finish
{
   "grantee-principal": m2.us-west-2.amazonaws.com,
   "retiring-principal": m2.us-west-2.amazonaws.com,
   "operations": [
       "GenerateDataKey"
   ]
   "condition": {
        "encryptionContextSubset": {
            “aws:m2:app”: “a1bc2defabc3defabc4defabcd”
   }
}
```

Memantau kunci enkripsi Anda untuk Modernisasi AWS Mainframe

Saat Anda menggunakan kunci yang dikelola AWS KMS pelanggan dengan sumber daya Modernisasi AWS Mainframe, Anda dapat menggunakan atau AWS CloudTrailAmazon CloudWatch Logs untuk melacak permintaan yang dikirim oleh Modernisasi AWS Mainframe. AWS KMS

Contoh untuk lingkungan runtime

Contoh berikut adalah AWS CloudTrail peristiwa untuk`DescribeKey`,, `CreateGrantGenerateDataKey`, dan `Decrypt` untuk memantau operasi KMS yang dipanggil oleh Modernisasi AWS Mainframe untuk mengakses data yang dienkripsi oleh kunci yang dikelola pelanggan Anda:

DescribeKey

AWS Modernisasi Mainframe menggunakan `DescribeKey` operasi untuk memverifikasi apakah kunci terkelola AWS KMS pelanggan yang terkait dengan lingkungan runtime Anda ada di akun dan wilayah.

Contoh peristiwa berikut mencatat `DescribeKey` operasi:


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
                "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-12-06T19:40:26Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2022-12-06T20:23:43Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DescribeKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "205.251.233.182",
    "userAgent": "ExampleDesktop/1.0 (V1; OS)",
    "requestParameters": {
        "keyId": "00dd0db0-0000-0000-ac00-b0c000SAMPLE"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
```

```
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_256_GCM_SHA384",
        "clientProvidedHostHeader": "kms.us-west-2.amazonaws.com"
    },
    "sessionCredentialFromConsole": "true"
}
```

CreateGrant

Saat Anda menggunakan kunci yang dikelola AWS KMS pelanggan untuk mengenkripsi lingkungan runtime Anda, Modernisasi AWS Mainframe mengirimkan beberapa `CreateGrant` permintaan atas nama Anda untuk melakukan operasi KMS yang diperlukan. Beberapa hibah yang dibuat oleh Modernisasi AWS Mainframe dihentikan segera setelah digunakan. Yang lain pensiun saat Anda menghapus lingkungan runtime.

Contoh peristiwa berikut mencatat `CreateGrant` operasi untuk peran eksekusi Lambda yang terkait dengan alur kerja Create Environment.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
```

```
                "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
                "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-12-06T20:11:45Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "m2.us-west-2.amazonaws.com"
    },
    "eventTime": "2022-12-06T20:23:09Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "m2.us-west-2.amazonaws.com",
    "userAgent": "m2.us-west-2.amazonaws.com",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
        "operations": [
            "Encrypt",
            "Decrypt",
            "ReEncryptFrom",
            "ReEncryptTo",
            "GenerateDataKey",
            "GenerateDataKey",
            "DescribeKey",
            "CreateGrant"
        ],
        "granteePrincipal": "m2.us-west-2.amazonaws.com",
        "retiringPrincipal": "m2.us-west-2.amazonaws.com"
    },
    "responseElements": {
        "grantId":
 "0ab0ac0d0b000f00ea00cc0a0e00fc00bce000c000f0000000c0bc0a0000aaafSAMPLE",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": false,
```

```
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

Contoh peristiwa berikut mencatat `CreateGrant` operasi untuk peran terkait layanan grup Auto Scaling. Peran eksekusi Lambda yang terkait dengan alur kerja Create Environment memanggil operasi ini. `CreateGrant` Ini memberikan izin untuk peran eksekusi untuk membuat subgrant terhadap peran terkait layanan grup Auto Scaling.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROA3YPCLM65MZFUPM4JO:EnvironmentWorkflow-alpha-
CreateEnvironmentLambda7-HfxDj5zz86tr",
        "arn": "arn:aws:sts::111122223333:assumed-role/EnvironmentWorkflow-
alpha-CreateEnvironmentLambdaS-1AU4A8VNQEEKN/EnvironmentWorkflow-alpha-
CreateEnvironmentLambda7-HfxDj5zz86tr",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
                "arn": "arn:aws:iam::111122223333:role/EnvironmentWorkflow-alpha-
CreateEnvironmentLambdaS-1AU4A8VNQEEKN",
                "accountId": "111122223333",
                "userName": "EnvironmentWorkflow-alpha-
CreateEnvironmentLambdaS-1AU4A8VNQEEKN"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-12-06T20:22:28Z",
```

```
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2022-12-06T20:23:09Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "54.148.236.160",
    "userAgent": "aws-sdk-java/2.18.21 Linux/4.14.255-276-224.499.amzn2.x86_64
 OpenJDK_64-Bit_Server_VM/11.0.14.1+10-LTS Java/11.0.14.1 vendor/Amazon.com_Inc. md/
internal exec-env/AWS_Lambda_java11 io/sync http/Apache cfg/retry-mode/legacy",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
        "operations": [
            "Encrypt",
            "Decrypt",
            "ReEncryptFrom",
            "ReEncryptTo",
            "GenerateDataKey",
            "GenerateDataKey",
            "DescribeKey",
            "CreateGrant"
        ],
        "granteePrincipal": "m2.us-west-2.amazonaws.com",
        "retiringPrincipal": "m2.us-west-2.amazonaws.com"
    },
    "responseElements": {
        "grantId":
 "0ab0ac0d0b000f00ea00cc0a0e00fc00bce000c000f0000000c0bc0a0000aaafSAMPLE",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
```

```
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_256_GCM_SHA384",
        "clientProvidedHostHeader": "kms.us-west-2.amazonaws.com"
    }
}
}
```

GenerateDataKey

Saat Anda mengaktifkan kunci terkelola AWS KMS pelanggan untuk sumber daya lingkungan runtime Anda, Auto Scaling akan membuat kunci unik untuk mengenkripsi volume Amazon EBS yang terkait dengan lingkungan runtime. Ini mengirimkan `GenerateDataKey` permintaan ke AWS KMS yang menentukan kunci yang dikelola AWS KMS pelanggan untuk sumber daya.

Contoh peristiwa berikut mencatat `GenerateDataKey` operasi:

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROA3YPCLM65EEXVIEH7D:AutoScaling",
        "arn": "arn:aws:sts::111122223333:assumed-role/AWSServiceRoleForAutoScaling/
AutoScaling",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
                "arn": "arn:aws:iam::111122223333:role/aws-service-role/
autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling",
                "accountId": "111122223333",
                "userName": "AWSServiceRoleForAutoScaling"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-12-06T20:23:16Z",
```

```
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "autoscaling.amazonaws.com"
    },
    "eventTime": "2022-12-06T20:23:18Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "autoscaling.amazonaws.com",
    "userAgent": "autoscaling.amazonaws.com",
    "requestParameters": {
        "encryptionContext": {
            "aws:ebs:id": "vol-080f7a32d290807f3"
        },
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
        "numberOfBytes": 64
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```


Decrypt

Saat Anda mengakses lingkungan runtime terenkripsi, Amazon EBS memanggil `Decrypt` operasi untuk menggunakan kunci data terenkripsi yang disimpan untuk mengakses data terenkripsi.

Contoh peristiwa berikut mencatat `Decrypt` operasi:

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "ebs.amazonaws.com"
    },
    "eventTime": "2022-12-06T20:23:22Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "ebs.amazonaws.com",
    "userAgent": "ebs.amazonaws.com",
    "requestParameters": {
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "encryptionContext": {
            "aws:ebs:id": "vol-080f7a32d290807f3"
        }
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "sharedEventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventCategory": "Management"
}
```

Contoh untuk aplikasi

Contoh berikut adalah AWS CloudTrail peristiwa untuk `CreateGrant` dan `GenerateDataKey` untuk memantau operasi KMS yang dipanggil oleh Modernisasi AWS Mainframe untuk mengakses data yang dienkripsi oleh kunci yang dikelola pelanggan Anda:

CreateGrant

Saat Anda menggunakan kunci yang dikelola AWS KMS pelanggan untuk mengenkripsi sumber daya aplikasi Anda, peran eksekusi Lambda mengirimkan `CreateGrant` permintaan atas nama Anda untuk mengakses kunci KMS di akun Anda. AWS Hibah ini memungkinkan peran eksekusi Lambda untuk mengunggah sumber daya aplikasi pelanggan ke Amazon S3 menggunakan kunci terkelola pelanggan Anda. Hibah ini dihentikan segera setelah aplikasi dibuat.

Contoh peristiwa berikut mencatat `CreateGrant` operasi:


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
                "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-12-06T21:51:45Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "m2.us-west-2.amazonaws.com"
    },
    "eventTime": "2022-12-06T22:47:04Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "m2.us-west-2.amazonaws.com",
    "userAgent": "m2.us-west-2.amazonaws.com",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
```

```
        "constraints": {
            "encryptionContextSubset": {
                "aws:m2:app": "a1bc2defabc3defabc4defabcd"
            }
        },
        "retiringPrincipal": "m2.us-west-2.amazonaws.com",
        "operations": [
            "GenerateDataKey"
        ],
        "granteePrincipal": "m2.us-west-2.amazonaws.com"
    },
    "responseElements": {
        "grantId":
 "0ab0ac0d0b000f00ea00cc0a0e00fc00bce000c000f0000000c0bc0a0000aaafSAMPLE",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

GenerateDataKey

Saat Anda mengaktifkan kunci terkelola AWS KMS pelanggan untuk sumber daya aplikasi Anda, peran eksekusi Lambda akan membuat kunci yang digunakan untuk mengenkripsi dan mengunggah data pelanggan ke Amazon Simple Storage Service. Peran eksekusi Lambda mengirimkan `GenerateDataKey` permintaan yang menentukan kunci AWS KMS yang dikelola AWS KMS pelanggan untuk sumber daya.

Contoh peristiwa berikut mencatat `GenerateDataKey` operasi:

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROA3YPCLM65CLCEKKC7Z:ApplicationWorkflow-alpha-
CreateApplicationVersion-CstWZUn5R4u6",
        "arn": "arn:aws:sts::111122223333:assumed-role/ApplicationWorkflow-
alpha-CreateApplicationVersion-1IZRBZYDG20B/ApplicationWorkflow-alpha-
CreateApplicationVersion-CstWZUn5R4u6",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
                "arn": "arn:aws:iam::111122223333:role/ApplicationWorkflow-alpha-
CreateApplicationVersion-1IZRBZYDG20B",
                "accountId": "111122223333",
                "userName": "ApplicationWorkflow-alpha-
CreateApplicationVersion-1IZRBZYDG20B"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-12-06T23:28:32Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "m2.us-west-2.amazonaws.com"
    },
    "eventTime": "2022-12-06T23:29:08Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "m2.us-west-2.amazonaws.com",
    "userAgent": "m2.us-west-2.amazonaws.com",
    "requestParameters": {
        "encryptionContext": {
            "aws:m2:app": "a1bc2defabc3defabc4defabcd",
            "aws:s3:arn": "arn:aws:s3:::supernova-processedtemplate-111122223333-us-
west-2/111122223333/a1bc2defabc3defabc4defabcd/1/cics-transaction/ZBNKE35.so"
        },
        "keySpec": "AES_256",
```

```
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

Contoh untuk penerapan

Contoh berikut adalah AWS CloudTrail peristiwa untuk `CreateGrant` dan `Decrypt` untuk memantau operasi KMS yang dipanggil oleh Modernisasi AWS Mainframe untuk mengakses data yang dienkripsi oleh kunci yang dikelola pelanggan Anda:

CreateGrant

Saat Anda menggunakan kunci yang dikelola AWS KMS pelanggan untuk mengenkripsi sumber daya penerapan Anda, Modernisasi AWS Mainframe mengirimkan dua `CreateGrant` permintaan atas nama Anda. Hibah pertama bertentangan dengan peran eksekusi Lambda saat ini untuk dipanggil ListBatchJobScriptFiles, dan dihentikan segera setelah penerapan selesai. Hibah kedua bertentangan dengan peran instans bawah cakupan Amazon EC2 sehingga Amazon EC2 dapat mengunduh sumber daya aplikasi pelanggan dari Amazon S3. Hibah ini dihentikan saat aplikasi dihapus dari lingkungan runtime.

Contoh peristiwa berikut mencatat `CreateGrant` operasi:

```
{
    "eventVersion": "1.08",
```

```
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
                "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-12-06T21:51:45Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "m2.us-west-2.amazonaws.com"
    },
    "eventTime": "2022-12-06T23:40:07Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "m2.us-west-2.amazonaws.com",
    "userAgent": "m2.us-west-2.amazonaws.com",
    "requestParameters": {
        "operations": [
            "Decrypt"
        ],
        "constraints": {
            "encryptionContextSubset": {
                "aws:m2:app": "a1bc2defabc3defabc4defabcd"
            }
        },
        "granteePrincipal": "m2.us-west-2.amazonaws.com",
        "retiringPrincipal": "m2.us-west-2.amazonaws.com",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
    "responseElements": {
```

```
        "grantId":
 "0ab0ac0d0b000f00ea00cc0a0e00fc00bce000c000f0000000c0bc0a0000aaafSAMPLE",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

Decrypt

Saat Anda mengakses penerapan, Amazon EC2 memanggil operasi untuk menggunakan kunci data terenkripsi `Decrypt` yang disimpan untuk mendekripsi dan mengunduh data pelanggan terenkripsi dari Amazon S3.

Contoh peristiwa berikut mencatat `Decrypt` operasi:

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROA3YPCLM65BSPZ37E6G:m2-hm-bqe367dxtfcpdbzmnhfzranisu",
        "arn": "arn:aws:sts::111122223333:assumed-role/
SupernovaEnvironmentInstanceScopeDownRole/m2-hm-bqe367dxtfcpdbzmnhfzranisu",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
```

```
                "arn": "arn:aws:iam::111122223333:role/
SupernovaEnvironmentInstanceScopeDownRole",
                "accountId": "111122223333",
                "userName": "SupernovaEnvironmentInstanceScopeDownRole"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-12-06T23:19:29Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "m2.us-west-2.amazonaws.com"
    },
    "eventTime": "2022-12-06T23:40:15Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "m2.us-west-2.amazonaws.com",
    "userAgent": "m2.us-west-2.amazonaws.com",
    "requestParameters": {
        "encryptionContext": {
            "aws:m2:app": "a1bc2defabc3defabc4defabcdm",
            "aws:s3:arn": "arn:aws:s3:::supernova-processedtemplate-111122223333-us-
west-2/111122223333/a1bc2defabc3defabc4defabcdm/1/cics-transaction/BBANK40P.so"
        },
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
```

```
}
```

Pelajari selengkapnya

Sumber daya berikut memberikan informasi lebih lanjut tentang enkripsi data saat istirahat.

- Untuk informasi selengkapnya tentang konsep AWS Key Management Service dasar, lihat Panduan AWS Key Management Service Pengembang.
- Untuk informasi selengkapnya tentang praktik terbaik Keamanan AWS Key Management Service, lihat Panduan AWS Key Management Service Pengembang.

Enkripsi dalam bergerak

Untuk aplikasi interaktif yang merupakan bagian dari beban kerja transaksional, pertukaran data antara emulator terminal dan titik akhir layanan Modernisasi AWS Mainframe untuk protokol TN3270 tidak dienkripsi dalam perjalanan. Jika aplikasi memerlukan enkripsi dalam perjalanan, Anda mungkin ingin menerapkan beberapa mekanisme tunneling tambahan.

AWS Modernisasi Mainframe menggunakan HTTPS untuk mengenkripsi API layanan. Semua komunikasi lain dalam Modernisasi AWS Mainframe dilindungi oleh VPC layanan atau grup keamanan, serta HTTPS. AWS Modernisasi Mainframe mentransfer artefak aplikasi, konfigurasi, dan data aplikasi. Artefak aplikasi disalin dari bucket Amazon S3 yang Anda miliki, seperti data aplikasi. Anda dapat memberikan konfigurasi aplikasi menggunakan tautan ke Amazon S3 atau dengan mengunggah file secara lokal.

Enkripsi dasar dalam perjalanan dikonfigurasi secara default, tetapi tidak berlaku untuk protokol TN3270. AWS Modernisasi Mainframe menggunakan HTTPS untuk titik akhir API, yang juga dikonfigurasi secara default.

Identity and Access Management untuk AWS Modernisasi Mainframe

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 AWS sumber daya Modernisasi Mainframe. IAM adalah Layanan AWS yang dapat Anda gunakan tanpa biaya tambahan.

Topik

Audiens

Cara Anda menggunakan AWS Identity and Access Management (IAM) berbeda, tergantung pada pekerjaan yang Anda lakukan di Modernisasi AWS Mainframe.

Pengguna layanan - Jika Anda menggunakan layanan Modernisasi AWS Mainframe untuk melakukan pekerjaan Anda, maka administrator Anda memberi Anda kredensi dan izin yang Anda butuhkan. Saat Anda menggunakan lebih banyak fitur Modernisasi AWS Mainframe 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 Modernisasi AWS Mainframe, lihat. Pemecahan Masalah Identitas dan akses Modernisasi AWS Mainframe

Administrator layanan - Jika Anda bertanggung jawab atas sumber daya Modernisasi AWS Mainframe di perusahaan Anda, Anda mungkin memiliki akses penuh ke AWS Modernisasi Mainframe. Tugas Anda adalah menentukan fitur dan sumber daya Modernisasi AWS Mainframe 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 Modernisasi AWS Mainframe, lihat. Bagaimana Modernisasi AWS Mainframe bekerja dengan IAM

Administrator IAM - Jika Anda seorang administrator IAM, Anda mungkin ingin mempelajari detail tentang cara menulis kebijakan untuk mengelola akses ke Modernisasi AWS Mainframe. Untuk melihat contoh kebijakan berbasis identitas Modernisasi AWS Mainframe yang dapat Anda gunakan di IAM, lihat. Contoh kebijakan berbasis identitas untuk Modernisasi Mainframe AWS

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 kredensi 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 lebih lanjut, 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 tugas lengkap

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 kredensi sementara.

Untuk pengelolaan akses terpusat, sebaiknya 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 yang dimaksud 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 temporer, dan bukan 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, sebaiknya rotasikan kunci akses. Untuk informasi selengkapnya, lihat Merotasi kunci akses secara teratur untuk kasus penggunaan yang memerlukan kredensial jangka panjang dalam Panduan Pengguna IAM.

Grup IAM adalah identitas yang menentukan 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 tersebut dimaksudkan untuk dapat diambil 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 metode untuk 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 diautentikasi, identitas tersebut dikaitkan dengan peran dan diberikan izin yang ditentukan oleh peran. Untuk informasi tentang peran untuk federasi, lihat Membuat peran untuk Penyedia Identitas pihak ketiga dalam Panduan Pengguna IAM. Jika Anda menggunakan Pusat Identitas IAM, Anda mengonfigurasi sekumpulan izin. Untuk mengontrol apa yang dapat diakses identitas Anda setelah identitas tersebut diautentikasi, Pusat Identitas IAM mengaitkan izin yang ditetapkan 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 secara 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. Contoh, ketika Anda melakukan panggilan dalam layanan, umumnya layanan tersebut menjalankan aplikasi di Amazon EC2 atau menyimpan objek di Amazon S3. Suatu layanan mungkin melakukan hal tersebut menggunakan izin pengguna utama panggilan, 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. Ketika 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 ketika mengajukan permintaan FAS, lihat Meneruskan sesi akses.
  - Peran IAM – Peran layanan adalah peran IAM yang diambil 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 tersebut dapat mengambil peran untuk melakukan sebuah 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 menentukan apakah permintaan diizinkan atau ditolak. Sebagian besar kebijakan disimpan AWS sebagai dokumen JSON. Untuk

informasi selengkapnya tentang struktur dan isi dokumen kebijakan JSON, lihat Ikhtisar 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 Anda lampirkan 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 dalam organisasi, Anda dapat menerapkan kebijakan kontrol layanan (SCP) ke sebagian 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 di salah satu kebijakan ini akan membatalkan izin tersebut. Untuk informasi selengkapnya, lihat Kebijakan sesi dalam Panduan Pengguna IAM.

Berbagai jenis kebijakan

Jika beberapa jenis kebijakan diberlakukan untuk satu 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 Modernisasi AWS Mainframe bekerja dengan IAM

Sebelum Anda menggunakan IAM untuk mengelola akses ke Modernisasi AWS Mainframe, pelajari fitur IAM apa yang tersedia untuk digunakan dengan Modernisasi Mainframe. AWS

Fitur IAM yang dapat Anda gunakan dengan Modernisasi AWS Mainframe

Fitur IAM	AWS Dukungan Modernisasi Mainframe
Kebijakan berbasis identitas	Ya
Kebijakan berbasis sumber daya	Tidak
Tindakan kebijakan	Ya
Sumber daya kebijakan	Ya
Kunci persyaratan kebijakan	Ya
ACL	Tidak
ABAC (tanda dalam kebijakan)	Ya
Kredensial sementara	Ya
Sesi akses teruskan (FAS)	Ya
Peran layanan	Ya

Fitur IAM	AWS Dukungan Modernisasi Mainframe
Peran terkait layanan	Ya

Untuk mendapatkan tampilan tingkat tinggi tentang bagaimana Modernisasi AWS Mainframe dan AWS layanan lainnya bekerja dengan sebagian besar fitur IAM, lihat AWS layanan yang bekerja dengan IAM di Panduan Pengguna IAM.

Kebijakan berbasis identitas untuk Modernisasi Mainframe AWS

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 Modernisasi Mainframe AWS

Untuk melihat contoh kebijakan berbasis identitas Modernisasi AWS Mainframe, lihat. Contoh kebijakan berbasis identitas untuk Modernisasi Mainframe AWS

Kebijakan berbasis sumber daya dalam Modernisasi Mainframe AWS

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 Modernisasi AWS Mainframe

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 Modernisasi AWS Mainframe, lihat Tindakan yang Ditentukan oleh Modernisasi AWS Mainframe di Referensi Otorisasi Layanan.

Tindakan kebijakan dalam Modernisasi AWS Mainframe menggunakan awalan berikut sebelum tindakan:

```
m2
```

Untuk menetapkan secara spesifik beberapa tindakan dalam satu pernyataan, pisahkan tindakan-tindakan tersebut dengan koma.

```
"Action": [
      "m2:StartApplication",
      "m2:StopApplication"
         ]
```

Anda juga dapat menentukan beberapa tindakan menggunakan wildcard (*). Sebagai contoh, untuk menentukan semua tindakan yang dimulai dengan kata `List`, sertakan tindakan berikut:

```
"Action": "m2:List*"
```

Untuk melihat contoh kebijakan berbasis identitas Modernisasi AWS Mainframe, lihat. Contoh kebijakan berbasis identitas untuk Modernisasi Mainframe AWS

Sumber daya kebijakan untuk Modernisasi AWS Mainframe

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": "*"
```

Anda dapat membatasi akses ke sumber daya Modernisasi AWS Mainframe tertentu dengan menggunakan ARN mereka untuk mengidentifikasi sumber daya yang diterapkan kebijakan IAM. Untuk informasi selengkapnya tentang format ARN, lihat Amazon Resource Name (ARN) di Referensi Umum AWS.

Misalnya, lingkungan Modernisasi AWS Mainframe memiliki ARN berikut.

```
"Resource": "arn:aws:m2:regionId:accountId:env/service-generated-unique-identifier"
```

Aplikasi Modernisasi AWS Mainframe memiliki ARN berikut.

```
"Resource": "arn:aws:m2:regionId:accountId:app/service-generated-unique-identifier"
```

Tidak semua tindakan Modernisasi AWS Mainframe mendukung izin tingkat sumber daya. Untuk tindakan yang tidak mendukung izin tingkat sumber daya, Anda harus menggunakan wildcard (*).

Tindakan Modernisasi AWS Mainframe berikut tidak mendukung izin tingkat sumber daya.

```
ListApplications
            ListApplicationVersions
            ListBatchJobDefinitions
            ListBatchJobExecutions
            ListDataSetImportHistory
            ListDataSets
            ListDeployments
            ListEngineVersions
            ListEnvironments
            ListTagsForResource
```

Untuk melihat daftar jenis sumber daya Modernisasi AWS Mainframe dan ARNnya, lihat Sumber Daya yang Ditentukan oleh Modernisasi AWS Mainframe di Referensi Otorisasi Layanan. Untuk mempelajari tindakan mana yang dapat Anda tentukan ARN dari setiap sumber daya, lihat Tindakan yang Ditentukan oleh Modernisasi AWS Mainframe.

Untuk melihat contoh kebijakan berbasis identitas Modernisasi AWS Mainframe, lihat. Contoh kebijakan berbasis identitas untuk Modernisasi Mainframe AWS

AWS Izin API Modernisasi Mainframe: Referensi tindakan, sumber daya, dan kondisi

Saat menulis kebijakan izin yang dapat dilampirkan ke identitas IAM (kebijakan berbasis identitas), Anda dapat menggunakan tabel berikut sebagai referensi. Tabel tersebut meliputi yang berikut:

- Setiap operasi AWS API Modernisasi Mainframe
- Tindakan terkait yang dapat Anda berikan izin untuk melakukan tindakan tersebut
- AWS Sumber daya yang dapat Anda berikan izin

Anda menentukan tindakan dalam bidang `Action` kebijakan, dan nilai sumber daya di dalam bidang `Resource` kebijakan.

Anda dapat menggunakan kunci kondisi AWS global dalam kebijakan Modernisasi AWS Mainframe Anda untuk menyatakan kondisi. Untuk daftar lengkap AWS kunci, lihat Kunci Kondisi Global yang Tersedia di Panduan Pengguna IAM.

> **Note**
>
> Untuk menentukan tindakan, gunakan awalan `m2:` diikuti dengan nama operasi API (misalnya, `m2:CreateApplication`).

AWS API Modernisasi Mainframe dan izin yang diperlukan untuk tindakan

AWS Operasi API Modernisasi Mainframe	Izin yang Diperlukan (Tindakan API)	Sumber daya
CancelBatchJobExecution		Aplikasi

CreateApplication	`s3:GetObject`
	
	`s3:ListBucket`
		Aplikasi

AWS Operasi API Modernisasi Mainframe	Izin yang Diperlukan (Tindakan API)	Sumber daya

CreateDataSetImportTask	`m2:CreateDataSetImportTask`
	
	`s3:GetObject`
		Aplikasi

CreateDeployment	`elasticloadbalancing:AddTags`
	
	`elasticloadbalancing:CreateListener`
	
	`elasticloadbalancing:CreateTargetGroup`
	
	`elasticloadbalancing:RegisterTargets`
		Aplikasi

AWS Operasi API Modernisasi Mainframe	Izin yang Diperlukan (Tindakan API)	Sumber daya
CreateEnvironment	ec2:CreateNetworkInterface ec2:CreateNetworkInterfacePermission ec2:DescribeNetworkInterfaces ec2:DescribeSecurityGroups ec2:DescribeSubnets ec2:DescribeVpcAttribute ec2:DescribeVpcs ec2:ModifyNetworkInterfaceAttribute elasticfilesystem:DescribeMountTargets elasticloadbalancing:AddTags elasticloadbalancing:CreateLoadBalancer fsx:DescribeFileSystems iam:CreateServiceLinkedRole	Environment

AWS Operasi API Modernisasi Mainframe	Izin yang Diperlukan (Tindakan API)	Sumber daya

DeleteApplication	`elasticloadbalancing:DeleteListener`
	`elasticloadbalancing:DeleteTargetGroup`
	`logs:DeleteLogDelivery`
		Aplikasi

DeleteApplicationFromEnvironment	`elasticloadbalancing:DeleteListener` `elasticloadbalancing:DeleteTargetGroup`	Aplikasi Environment
DeleteEnvironment	`elasticloadbalancing:DeleteLoadBalancer`	Environment
GetApplication		Aplikasi
GetApplicationVersion		Aplikasi
GetBatchJobExecution		Aplikasi
GetDataSetDetails		Aplikasi
GetDataSetImportTask		Aplikasi
GetDeployment		Aplikasi
GetEnvironment		Environment
ListApplications		*
ListApplicationVersions		*

AWS Operasi API Modernisasi Mainframe	Izin yang Diperlukan (Tindakan API)	Sumber daya
ListBatchJobDefinitions		*
ListBatchJobExecutions		*
ListDataSetImportHistory		*
ListDataSets		*
ListDeployments		*
ListEngineVersions		*
ListEnvironments		*
ListTagsForResource		*
StartApplication		Aplikasi
StartBatchJob		Aplikasi
StopApplication		Aplikasi
TagResource		*
UntagResource		*
UpdateApplication	`s3:GetObject` `s3:ListBucket`	Aplikasi
UpdateEnvironment		Environment

Kunci kondisi kebijakan untuk Modernisasi AWS Mainframe

Mendukung kunci kondisi kebijakan spesifik layanan	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 `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. Sebagai contoh, Anda dapat memberikan izin kepada pengguna IAM untuk mengakses sumber daya hanya jika izin tersebut mempunyai tanda yang sesuai dengan nama pengguna IAM mereka. Untuk informasi selengkapnya, silakan lihat Elemen kebijakan IAM: variabel dan tanda 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.

Kunci kondisi berikut khusus untuk Modernisasi AWS Mainframe

```
m2:EngineType
            m2:InstanceType
```

Untuk melihat daftar kunci kondisi Modernisasi AWS Mainframe, lihat Kunci Kondisi untuk Modernisasi AWS Mainframe di Referensi Otorisasi Layanan. Untuk mempelajari tindakan dan sumber daya yang dapat Anda gunakan kunci kondisi, lihat Tindakan yang Ditentukan oleh Modernisasi AWS Mainframe.

Untuk melihat contoh kebijakan berbasis identitas Modernisasi AWS Mainframe, lihat. Contoh kebijakan berbasis identitas untuk Modernisasi Mainframe AWS

Daftar kontrol akses (ACL) di Modernisasi AWS Mainframe

Mendukung ACL	Tidak

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 Modernisasi Mainframe AWS

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-operasi ketika tanda milik pengguna utama cocok dengan tanda yang ada di sumber daya yang ingin diakses.

ABAC sangat berguna di lingkungan yang berkembang dengan cepat dan berguna dalam situasi di mana pengelolaan kebijakan menjadi rumit.

Untuk mengendalikan akses berdasarkan tag, berikan informasi tentang tanda di elemen syarat dari sebuh kebijakan dengan menggunakan kunci-kunci persyaratan `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 AWS Modernisasi Mainframe

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 kredensi 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 kredensi sementara alih-alih menggunakan kunci akses jangka panjang. Untuk informasi selengkapnya, lihat Kredensial keamanan sementara di IAM.

Teruskan sesi akses untuk Modernisasi AWS Mainframe

Mendukung sesi akses maju (FAS)	Ya

Saat Anda menggunakan pengguna atau peran IAM untuk melakukan tindakan AWS, Anda dianggap sebagai prinsipal. Ketika 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 ketika mengajukan permintaan FAS, lihat Meneruskan sesi akses.

> **Important**
>
> Token ini memberikan akses Modernisasi AWS Mainframe ke data pelanggan tanpa persetujuan eksplisit Anda; misalnya, Modernisasi AWS Mainframe menyebarkan artefak aplikasi dengan data bisnis terkait dari bucket Amazon S3 tanpa mendapatkan izin eksplisit dari pelanggan. Anda mungkin perlu memperbarui dokumentasi kepatuhan apa pun yang sesuai.

Peran layanan untuk Modernisasi AWS Mainframe

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.

AWS Modernisasi Mainframe mendukung peran layanan untuk kait aktivitas (transaksi/pekerjaan abends atau penyelesaian, dll).

> **Warning**
>
> Mengubah izin untuk peran layanan dapat merusak fungsionalitas Modernisasi AWS Mainframe. Edit peran layanan hanya ketika Modernisasi AWS Mainframe memberikan panduan untuk melakukannya.

Memilih peran IAM dalam Modernisasi AWS Mainframe

Jika sebelumnya Anda telah membuat peran IAM yang dapat diasumsikan oleh aplikasi yang berjalan di Amazon EC2, Anda dapat memilih peran ini saat membuat templat peluncuran atau konfigurasi peluncuran. AWS Modernisasi Mainframe memberi Anda daftar peran untuk dipilih. Saat membuat peran ini, penting untuk mengaitkan kebijakan hak akses IAM yang membatasi akses ke panggilan API khusus yang diperlukan aplikasi. Untuk informasi selengkapnya, lihat IAM role untuk aplikasi yang berjalan di Instans Amazon EC2 di Panduan Pengguna Amazon EC2 Auto Scaling.

Peran terkait layanan untuk AWS Modernisasi Mainframe

Mendukung peran yang terkait layanan	Ya

Peran terkait layanan adalah jenis peran layanan yang ditautkan ke. Layanan AWS Layanan tersebut dapat mengambil peran untuk melakukan sebuah 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 membuat atau mengelola peran terkait layanan Modernisasi AWS Mainframe, lihat. Menggunakan peran terkait layanan untuk Modernisasi Mainframe

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 Modernisasi Mainframe AWS

Secara default, pengguna dan peran tidak memiliki izin untuk membuat atau memodifikasi AWS sumber daya Modernisasi Mainframe. 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 Modernisasi AWS Mainframe, termasuk format ARN untuk setiap jenis sumber daya, lihat Tindakan, Sumber Daya, dan Kunci Kondisi untuk Modernisasi AWS Mainframe dalam Referensi Otorisasi Layanan.

Topik

- Praktik terbaik kebijakan
- Menggunakan konsol Modernisasi AWS Mainframe
- Izinkan pengguna melihat izin mereka sendiri

Praktik terbaik kebijakan

Kebijakan berbasis identitas menentukan apakah seseorang dapat membuat, mengakses, atau menghapus AWS sumber daya Modernisasi Mainframe 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. Misalnya, Anda dapat menulis syarat kebijakan untuk menentukan bahwa semua pengajuan 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 Modernisasi AWS Mainframe

Untuk mengakses konsol Modernisasi AWS Mainframe, Anda harus memiliki set izin minimum. Izin ini harus memungkinkan Anda untuk membuat daftar dan melihat detail tentang sumber daya Modernisasi AWS Mainframe 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 Modernisasi AWS Mainframe, lampirkan juga Modernisasi AWS Mainframe `ConsoleAccess` atau `ReadOnly` AWS kebijakan 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 Identitas dan akses Modernisasi AWS Mainframe

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah umum yang mungkin Anda temui saat bekerja dengan Modernisasi AWS Mainframe dan IAM.

Topik

- Saya tidak berwenang untuk melakukan iam: PassRole
- Saya ingin mengizinkan orang di luar saya Akun AWS untuk mengakses sumber daya Modernisasi AWS Mainframe saya

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 Modernisasi AWS Mainframe.

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 dalam Modernisasi AWS Mainframe. 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 saya Akun AWS untuk mengakses sumber daya Modernisasi AWS Mainframe 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 Modernisasi AWS Mainframe mendukung fitur-fitur ini, lihat. Bagaimana Modernisasi AWS Mainframe 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.

Menggunakan peran terkait layanan untuk Modernisasi Mainframe

AWS Mainframe Modernization menggunakan AWS Identity and Access Management (IAM) peran tertaut layanan. Peran terkait layanan adalah jenis unik peran IAM yang terkait langsung dengan Modernisasi Mainframe. Peran terkait layanan telah ditentukan sebelumnya oleh Modernisasi Mainframe dan mencakup semua izin yang diperlukan layanan untuk memanggil layanan lain atas nama Anda. AWS

Peran terkait layanan membuat pengaturan Modernisasi Mainframe lebih mudah karena Anda tidak perlu menambahkan izin yang diperlukan secara manual. Modernisasi Mainframe mendefinisikan izin dari peran terkait layanannya, dan kecuali ditentukan lain, hanya Modernisasi Mainframe yang dapat mengambil perannya. Izin yang ditentukan mencakup kebijakan kepercayaan dan kebijakan izin, serta bahwa kebijakan izin tidak dapat dilampirkan ke entitas IAM lainnya.

Anda dapat menghapus peran tertaut layanan hanya setelah menghapus sumber daya terkait terlebih dahulu. Ini melindungi sumber daya Modernisasi Mainframe Anda karena Anda tidak dapat secara tidak sengaja menghapus izin untuk mengakses sumber daya.

Untuk informasi tentang layanan lain yang mendukung peran tertaut layanan, silakan lihat Layanan AWS yang Bisa Digunakan dengan IAM dan carilah layanan yang memiliki opsi Ya di kolom Peran tertaut layanan. Pilih Ya dengan sebuah tautan untuk melihat dokumentasi peran tertaut layanan untuk layanan tersebut.

Izin peran terkait layanan untuk Modernisasi Mainframe

Modernisasi Mainframe menggunakan peran terkait layanan bernama AWSServiceRoleForAWSM2 — konfigurasikan jaringan untuk terhubung ke VPC Anda dan mengakses sumber daya seperti sistem file.

AWSServiceRoleForAWSM2 peran terkait layanan mempercayai layanan berikut untuk mengambil peran:

- `m2.amazonaws.com`

Kebijakan izin peran bernama AWSM2ServicePolicy memungkinkan Modernisasi Mainframe untuk menyelesaikan tindakan berikut pada sumber daya yang ditentukan:

- Buat, hapus, jelaskan, dan lampirkan izin ke antarmuka jaringan Amazon EC2 untuk lingkungan Modernisasi Mainframe guna membangun konektivitas ke VPC pelanggan.

- Daftarkan atau hapus pendaftaran entri dari Elastic Load Balancing, yang merupakan cara pelanggan terhubung ke lingkungan Modernisasi Mainframe.
- Jelaskan sistem file Amazon EFS atau Amazon FSx, jika digunakan.
- Memancarkan metrik ke pelanggan CloudWatch dari lingkungan runtime.

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "ec2:DescribeSubnets",
    "ec2:CreateNetworkInterface",
    "ec2:DeleteNetworkInterface",
    "ec2:DescribeNetworkInterfaces",
    "ec2:CreateNetworkInterfacePermission",
    "ec2:ModifyNetworkInterfaceAttribute"
   ],
   "Resource": "*"
  },
  {
   "Effect": "Allow",
   "Action": [
    "elasticfilesystem:DescribeMountTargets"
   ],
   "Resource": "*"
  },
  {
   "Effect": "Allow",
   "Action": [
    "elasticloadbalancing:RegisterTargets",
    "elasticloadbalancing:DeregisterTargets"
   ],
   "Resource": "*"
  },
  {
   "Effect": "Allow",
   "Action": [
    "fsx:DescribeFileSystems"
   ],
   "Resource": "*"
```

```
  },
  {
   "Effect": "Allow",
   "Action": [
    "cloudwatch:PutMetricData"
   ],
   "Resource": "*",
   "Condition": {
    "StringEquals": {
     "cloudwatch:namespace": [
      "AWS/M2"
     ]
    }
   }
  }
 ]
}
```

Anda harus mengonfigurasi izin agar entitas IAM (seperti pengguna, grup, atau peran) dapat membuat, mengedit, atau menghapus peran tertaut layanan. Untuk informasi selengkapnya, silakan lihat Izin peran tertaut layanan di Panduan Pengguna IAM.

Membuat peran terkait layanan untuk Modernisasi Mainframe

Anda tidak perlu membuat peran tertaut layanan secara manual. Saat Anda membuat lingkungan runtime di, APIAWS Management Console, atau AWS APIAWS CLI, Modernisasi Mainframe membuat peran terkait layanan untuk Anda.

Jika Anda menghapus peran tertaut layanan ini, dan ingin membuatnya lagi, Anda dapat mengulangi proses yang sama untuk membuat kembali peran tersebut di akun Anda. Saat Anda membuat lingkungan runtime, Modernisasi Mainframe menciptakan peran terkait layanan untuk Anda lagi.

Mengedit peran terkait layanan untuk Modernisasi Mainframe

Modernisasi Mainframe tidak memungkinkan Anda untuk mengedit AWSServiceRoleForAWSM 2 peran terkait layanan. Setelah membuat peran tertaut layanan, Anda tidak dapat mengubah nama peran karena berbagai entitas mungkin merujuk peran tersebut. Namun demikian, Anda dapat menyunting penjelasan peran menggunakan IAM. Untuk informasi selengkapnya, silakan lihat Menyunting peran tertaut layanan dalam Panduan Pengguna IAM.

Menghapus peran terkait layanan untuk Modernisasi Mainframe

Jika Anda tidak perlu lagi menggunakan fitur atau layanan yang memerlukan peran terkait layanan, kami merekomendasikan Anda menghapus peran tersebut. Dengan begitu, Anda tidak memiliki entitas yang tidak digunakan yang tidak dipantau atau dipelihara secara aktif. Tetapi, Anda harus membersihkan sumber daya peran terkait layanan sebelum menghapusnya secara manual.

> **Note**
>
> Jika layanan Modernisasi Mainframe menggunakan peran saat Anda mencoba menghapus sumber daya, maka penghapusan mungkin gagal. Jika hal itu terjadi, tunggu beberapa menit dan coba mengoperasikannya lagi.

Untuk menghapus sumber daya Modernisasi Mainframe yang digunakan oleh 2 AWSServiceRoleForAWSM

- Hapus lingkungan runtime di Modernisasi Mainframe. Pastikan untuk menghapus aplikasi dari lingkungan sebelum menghapus lingkungan itu sendiri.

Untuk menghapus peran tertaut layanan secara manual menggunakan IAM

Gunakan konsol IAM, theAWS CLI, atau AWS API untuk menghapus AWSServiceRoleForAWSM 2 peran terkait layanan. Untuk informasi selengkapnya, lihat Menghapus peran tertaut layanan di Panduan Pengguna IAM.

Wilayah yang didukung untuk peran terkait layanan Modernisasi Mainframe

Modernisasi Mainframe mendukung penggunaan peran terkait layanan di semua wilayah tempat layanan tersedia. Untuk informasi selengkapnya, silakan lihat Wilayah AWS dan titik akhir.

Validasi kepatuhan untukAWS

Auditor pihak ketiga menilai keamanan dan kepatuhan ModernisasiAWS Mainframe sebagai bagian dari beberapa programAWS kepatuhan. Program ini mencakup SOC, PCI, FedRAMP, HIPAA, dan lainnya.

Untuk daftar layanan AWS dalam cakupan program kepatuhan tertentu, lihat Layanan AWS dalam Cakupan Program Kepatuhan. Untuk informasi umum, lihat Program Kepatuhan AWS.

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 ModernisasiAWS Mainframe ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, serta undang-undang 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.
- Merancang Laporan Resmi Keamanan dan Kepatuhan HIPAA – Laporan resmi ini menjelaskan cara perusahaan dapat menggunakan AWS untuk membuat aplikasi yang patuh-HIPAA.
- 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, pedoman industri, dan peraturan.
- AWS Security Hub – Layanan AWS ini akan menyediakan tampilan komprehensif status keamanan dalam AWS yang akan membantu Anda dalam memeriksa kepatuhan terhadap standar industri keamanan dan praktik terbaik.

Ketahanan diAWSModernisasi Mainframe

Infrastruktur global AWS dibangun di sekitar Wilayah AWS dan Zona Ketersediaan. Wilayah memberikan beberapa Zona Ketersediaan yang terpisah dan terisolasi secara fisik, yang terkoneksi melalui jaringan latensi rendah, throughput tinggi, dan sangat redundan. Dengan Zona Ketersediaan, Anda dapat merancang serta mengoperasikan aplikasi dan basis data yang secara otomatis melakukan fail over di antara zona tanpa gangguan. Availability Zone lebih tersedia, memiliki toleransi kesalahan, dan dapat diskalakan dibandingkan dengan satu atau beberapa infrastruktur pusat data tradisional.

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

Keamanan infrastruktur dalam AWS Mainframe Modernization

Sebagai layanan terkelola, AWS Mainframe Modernization dilindungi oleh keamanan jaringan AWS global. Untuk informasi tentang layanan AWS keamanan dan cara AWS melindungi infrastruktur, lihat Keamanan AWS Cloud. Untuk mendesain AWS lingkungan Anda menggunakan praktik terbaik untuk keamanan infrastruktur, lihat Perlindungan Infrastruktur dalam Kerangka Kerja yang AWSDiarsiteksikan dengan Baik Pilar Keamanan.

Anda menggunakan panggilan API yang AWS dipublikasikan untuk mengakses Modernisasi Mainframe melalui jaringan. Klien harus mendukung hal-hal berikut:

- Keamanan Lapisan Transportasi (TLS). Kami membutuhkan TLS 1.2 dan merekomendasikan TLS 1.3.
- Cipher suite dengan perfect forward secrecy (PFS) seperti DHE (Ephemeral Diffie-Hellman) atau ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Sebagian besar sistem modern seperti Java 7 dan sistem yang lebih baru mendukung mode ini.

Selain itu, permintaan harus ditandatangani menggunakan access key ID dan secret access key yang terkait dengan principal IAM. Atau Anda bisa menggunakan AWS Security Token Service (AWS STS) untuk membuat kredensial keamanan sementara guna menandatangani permintaan.

Akses AWS Mainframe Modernization menggunakan endpoint antarmuka () AWS PrivateLink

Anda dapat menggunakan AWS PrivateLink untuk membuat koneksi pribadi antara VPC Anda dan. AWS Mainframe Modernization Anda dapat mengakses Modernisasi Mainframe seolah-olah berada di VPC Anda, tanpa menggunakan gateway internet, perangkat NAT, koneksi VPN, atau koneksi. AWS Direct Connect Instans di VPC Anda tidak memerlukan alamat IP publik untuk mengakses Modernisasi Mainframe.

Anda membuat koneksi pribadi ini dengan membuat titik akhir antarmuka, yang didukung olehAWS PrivateLink. Kami membuat antarmuka jaringan endpoint di setiap subnet yang Anda aktifkan untuk titik akhir antarmuka. Ini adalah antarmuka jaringan yang dikelola pemohon yang berfungsi sebagai titik masuk untuk lalu lintas yang ditujukan untuk Modernisasi Mainframe.

Untuk informasi selengkapnya, lihat Mengakses Layanan AWS melalui AWS PrivateLink di Panduan AWS PrivateLink.

Pertimbangan untuk Modernisasi Mainframe

Sebelum Anda menyiapkan titik akhir antarmuka untuk Modernisasi Mainframe, tinjau Pertimbangan dalam Panduan. AWS PrivateLink

Modernisasi Mainframe mendukung panggilan ke semua tindakan API-nya melalui titik akhir antarmuka.

Buat titik akhir antarmuka untuk Modernisasi Mainframe

Anda dapat membuat titik akhir antarmuka untuk Modernisasi Mainframe menggunakan konsol VPC Amazon atau (). AWS Command Line Interface AWS CLI Untuk informasi selengkapnya, lihat Membuat titik akhir antarmuka di AWS PrivateLinkPanduan.

Buat endpoint antarmuka untuk Modernisasi Mainframe menggunakan nama layanan berikut:

```
com.amazonaws.region.m2
```

Jika Anda mengaktifkan DNS pribadi untuk titik akhir antarmuka, Anda dapat membuat permintaan API ke Modernisasi Mainframe menggunakan nama DNS Regional default. Sebagai contoh, `m2.us-east-1.amazonaws.com`.

Buat kebijakan titik akhir untuk titik akhir antarmuka Anda

Kebijakan endpoint adalah sumber daya IAM yang dapat Anda lampirkan ke titik akhir antarmuka. Kebijakan endpoint default memungkinkan akses penuh ke Modernisasi Mainframe melalui titik akhir antarmuka. Untuk mengontrol akses yang diizinkan ke Modernisasi Mainframe dari VPC Anda, lampirkan kebijakan endpoint khusus ke titik akhir antarmuka.

kebijakan titik akhir mencantumkan informasi berikut:

- Prinsipal yang dapat melakukan tindakan (Akun AWS, pengguna, dan peran IAM).
- Tindakan-tindakan yang dapat dilakukan.
- Sumber daya untuk melakukan tindakan.

Untuk informasi selengkapnya, lihat Mengontrol akses ke layanan menggunakan kebijakan titik akhir di Panduan AWS PrivateLink.

Contoh: Kebijakan titik akhir VPC untuk tindakan Modernisasi Mainframe

Berikut ini adalah contoh kebijakan endpoint kustom. Saat Anda melampirkan kebijakan ini ke titik akhir antarmuka Anda, kebijakan ini akan memberikan akses ke tindakan Modernisasi Mainframe yang terdaftar untuk semua prinsipal di semua sumber daya.

```
//Example of an endpoint policy where access is granted to the
//listed AWS Mainframe Modernization actions for all principals on all resources
{"Statement": [
      {"Principal": "*",
         "Effect": "Allow",
         "Action": [
            "m2:ListApplications",
            "m2:ListEnvironments",
            "m2:ListDeployments"
         ],
         "Resource":"*"
      }
   ]
}

//Example of an endpoint policy where access is denied to all the
//AWS Mainframe Modernization CREATE actions for all principals on all resources
{"Statement": [
      {"Principal": "*",
         "Effect": "Deny",
         "Action": [
            "m2:Create*"
         ],
         "Resource":"*"
      }
   ]
}
```

Pemantauan AWS Modernisasi Mainframe

Pemantauan adalah bagian penting dalam menjaga keandalan, ketersediaan, dan kinerja Modernisasi AWS Mainframe dan solusi AWS Anda yang lain. AWS menyediakan alat pemantauan berikut untuk menonton Modernisasi AWS Mainframe, 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 tahan lama. 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.

Memantau Modernisasi AWS Mainframe dengan Amazon CloudWatch

Anda dapat memantau Modernisasi AWS Mainframe menggunakan CloudWatch, yang mengumpulkan data mentah dan memprosesnya menjadi metrik yang dapat dibaca, mendekati waktu nyata. 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.

Tabel berikut mencantumkan metrik dan dimensi untuk Modernisasi AWS Mainframe. Namespace untuk metrik ini adalah. `AWS/M2`

Metrik Lingkungan Runtime

Metrik	Deskripsi
CPUUtilization	Pemanfaatan CPU instance di lingkungan. Dimensi: EnvironmentID Unit: Persen Statistik yang valid: Rata-rata, Minimum, Maksimum
InboundNetworkThroughput	Throughput jaringan inbound dari instance di lingkungan. Dimensi: EnvironmentID Unit: Byte per detik Statistik yang valid: Rata-rata, Minimum, Maksimum
MemoryUtilization	Pemanfaatan memori instance di lingkungan. Dimensi: EnvironmentID Unit: Persen Statistik yang valid: Rata-rata, Minimum, Maksimum
OutboundNetworkThroughput	Throughput jaringan keluar dari instance di lingkungan. Dimensi: EnvironmentID Unit: Byte per detik

Metrik	Deskripsi
	Statistik yang valid: Rata-rata, Minimum, Maksimum

Metrik Aplikasi

Metrik	Deskripsi
BatchJobCompletedCount	Jumlah pekerjaan yang diselesaikan selama interval waktu. Metrik ini tersedia untuk Micro Focus dan untuk AWS Blu Age 3.7.0 dan rilis yang lebih baru. Dimensi: ApplicationId Unit: Hitungan Statistik yang valid: Jumlah
BatchJobFailedCount	Jumlah pekerjaan yang gagal selama interval waktu. Metrik ini tersedia untuk Micro Focus dan untuk AWS Blu Age 3.7.0 dan rilis yang lebih baru. Dimensi: ApplicationId Unit: Hitungan Statistik yang valid: Jumlah
JvmMemoryFree	Jumlah memori yang tersedia yang saat ini tidak digunakan oleh Java Virtual Machine. Metrik ini hanya tersedia untuk mesin runtime AWS Blu Age. Ini tersedia untuk AWS Blu Age 3.7.0 dan rilis yang lebih baru.

Metrik	Deskripsi
	Dimensi: ApplicationId Satuan: Byte Statistik yang valid: Rata-rata, Minimum, Maksimum
JvmMemoryMax	Jumlah maksimum memori yang diizinkan untuk Java Virtual Machine. Metrik ini hanya tersedia untuk mesin runtime AWS Blu Age. Ini tersedia untuk AWS Blu Age 3.7.0 dan rilis yang lebih baru. Dimensi: ApplicationId Satuan: Byte Statistik yang valid: Rata-rata, Minimum, Maksimum
JvmMemoryUsed	Jumlah memori yang aktif digunakan oleh Java Virtual Machine. Metrik ini hanya tersedia untuk mesin runtime AWS Blu Age. Ini tersedia untuk AWS Blu Age 3.7.0 dan rilis yang lebih baru. Dimensi: ApplicationId Satuan: Byte Statistik yang valid: Rata-rata, Minimum, Maksimum

Metrik	Deskripsi
ProcessesActiveCount	Jumlah aktif proses eksekusi layanan bersamaan yang memproses permintaan. Metrik ini hanya tersedia untuk mesin runtime Micro Focus. Dimensi: ApplicationId Unit: Hitungan Statistik yang valid: Jumlah
SessionCount	Jumlah sesi HTTP untuk aplikasi. Metrik ini hanya tersedia untuk mesin runtime AWS Blu Age. Ini tersedia untuk AWS Blu Age 3.7.0 dan rilis yang lebih baru. Dimensi: ApplicationId Unit: Hitungan Statistik yang valid: Rata-rata, Minimum, Maksimum
SharedMemoryFree	Memori yang tersedia untuk server perusahaan untuk menyimpan semua informasi yang dibutuhkan untuk menjalankan transaksi dan pekerjaan. Metrik ini hanya tersedia untuk mesin runtime Micro Focus. Dimensi: ApplicationId Unit: Hitungan Statistik yang valid: Rata-rata, Minimum, Maksimum

Metrik	Deskripsi
ThreadActiveCount	Jumlah utas mesin yang memproses permintaan. Metrik ini hanya tersedia untuk mesin runtime AWS Blu Age. Ini tersedia untuk AWS Blu Age 3.7.0 dan rilis yang lebih baru. Dimensi: ApplicationId Unit: Hitungan Statistik yang valid: Rata-rata, Minimum, Maksimum
TransactionCompletedCount	Jumlah transaksi yang dilakukan selama interval waktu. Metrik ini tersedia untuk Micro Focus dan untuk AWS Blu Age 3.7.0 dan rilis yang lebih baru. Dimensi: ApplicationId Unit: Hitungan Statistik yang valid: Jumlah
TransactionFailedCount	Jumlah transaksi yang gagal selama interval waktu. Metrik ini tersedia untuk Micro Focus dan untuk AWS Blu Age 3.7.0 dan rilis yang lebih baru. Dimensi: ApplicationId Unit: Hitungan Statistik yang valid: Jumlah

Metrik	Deskripsi
TransactionResponseTime	Jumlah waktu dari saat pengguna mengirim permintaan hingga waktu aplikasi menunjukkan bahwa permintaan telah selesai. Metrik ini tersedia untuk Micro Focus dan untuk AWS Blu Age 3.7.0 dan rilis yang lebih baru. Dimensi: ApplicationId Unit: Milidetik Statistik yang valid: Rata-rata, Minimum, Maksimum

Dimensi

Dimensi	Deskripsi
applicationId	Dimensi ini menyaring metrik ke aplikasi yang diidentifikasi oleh ID.
lingkunganTid	Dimensi ini menyaring metrik ke lingkungan yang diidentifikasi oleh ID.

Logging AWS panggilan API Modernisasi Mainframe menggunakan AWS CloudTrail

AWSModernisasi Mainframe terintegrasi denganAWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan dalam AWS Modernisasi Mainframe. CloudTrail menangkap semua panggilan API untuk Modernisasi AWS Mainframe sebagai peristiwa. Panggilan yang diambil termasuk panggilan dari konsol Modernisasi AWS Mainframe dan panggilan kode ke operasi API Modernisasi AWS Mainframe. Jika Anda membuat jejak, Anda dapat mengaktifkan pengiriman CloudTrail acara secara berkelanjutan ke bucket Amazon S3, termasuk acara untuk Modernisasi AWS Mainframe. Jika Anda tidak mengonfigurasi jejak, Anda masih

dapat melihat peristiwa terbaru di CloudTrail konsol dalam Riwayat acara. Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat untuk Modernisasi AWS Mainframe, alamat IP dari mana permintaan dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan.

Untuk mempelajari selengkapnya CloudTrail, lihat Panduan AWS CloudTrail Pengguna.

AWSInformasi Modernisasi Mainframe di CloudTrail

CloudTrail diaktifkan di AWS akun Anda saat Anda membuat akun. Ketika aktivitas terjadi dalam Modernisasi AWS Mainframe, aktivitas tersebut dicatat dalam suatu CloudTrail peristiwa bersama dengan peristiwa AWS layanan lainnya dalam riwayat Acara. Anda dapat melihat, mencari, dan mengunduh peristiwa terbaru di akun AWS. Untuk informasi selengkapnya, lihat Melihat Acara dengan Riwayat CloudTrail Acara.

Untuk catatan peristiwa yang sedang berlangsung di AWS akun Anda, termasuk acara untuk Modernisasi AWS Mainframe, buat jejak. Jejak memungkinkan CloudTrail untuk mengirimkan file log ke bucket Amazon S3. Secara default, ketika Anda membuat jejak di konsol tersebut, jejak diterapkan ke semua Wilayah AWS. Jejak mencatat kejadian dari semua Wilayah di partisi AWS dan mengirimkan berkas log ke bucket Amazon S3 yang Anda tentukan. Selain itu, Anda dapat mengonfigurasi AWS layanan lain untuk menganalisis lebih lanjut dan menindaklanjuti data peristiwa yang dikumpulkan dalam CloudTrail log. Untuk informasi selengkapnya, lihat berikut:

- Gambaran umum untuk membuat jejak
- CloudTrail layanan dan integrasi yang didukung
- Mengonfigurasi notifikasi Amazon SNS untuk CloudTrail
- Menerima file CloudTrail log dari beberapa Wilayah
- Menerima file CloudTrail log dari beberapa akun

Semua tindakan Modernisasi AWS Mainframe dicatat oleh CloudTrail dan didokumentasikan dalam Referensi API Modernisasi AWSMainframe. Misalnya, panggilan ke`CreateApplication`, `CreateEnvironment` dan `CreateDeployment` 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 berikut hal ini:

- Baik permintaan tersebut dibuat dengan kredensial pengguna root atau pengguna.

- Baik permintaan tersebut dibuat dengan kredensial keamanan sementara untuk peran atau pengguna gabungan.
- Bahwa permintaan tersebut dibuat oleh layanan AWS lainnya.

Untuk informasi selengkapnya, lihat Elemen userIdentity CloudTrail .

Memahami AWS entri file log Modernisasi Mainframe

Trail adalah konfigurasi yang memungkinkan pengiriman peristiwa sebagai file log ke bucket Amazon S3 yang Anda tentukan. CloudTrail file log berisi satu atau lebih entri log. Peristiwa mewakili permintaan tunggal dari sumber manapun dan mencakup informasi tentang tindakan yang diminta, tanggal dan waktu tindakan, parameter permintaan, dan sebagainya. CloudTrail file log bukanlah jejak tumpukan yang diurutkan dari panggilan API publik, jadi file tersebut tidak muncul dalam urutan tertentu.

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan `CreateApplication` tindakan.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAII6WZTHGYAEXAMPLE",
        "arn": "arn:aws:sts::444455556666:assumed-role/Admin/Mary_Major",
        "accountId": "444455556666",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAII6WZTHGYAEXAMPLE",
                "arn": "arn:aws:iam::444455556666:role/Admin",
                "accountId": "444455556666",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-06-01T20:38:22Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2022-06-01T20:40:39Z",
```

```
    "eventSource": "m2.amazonaws.com",
    "eventName": "CreateApplication",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "72.21.196.65",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:91.0) Gecko/20100101
 Firefox/91.0",
    "requestParameters": {
        "clientToken": "1abc23de-f45g-6789-h01i-jkl2m3456789",
        "name": "MyApp",
        "description": "",
        "engineType": "microfocus",
        "definition": {
            "content": "{}"
        },
        "tags": {}
    },
    "responseElements": {
        "applicationVersion": 1,
        "Access-Control-Expose-Headers": "x-amzn-RequestId,x-amzn-ErrorType,x-amzn-
ErrorMessage,Date",
        "applicationArn": "arn:aws:m2:us-east-1:444455556666:app/
lsfhmwhw7fffrosff2lncwqcua",
        "applicationId": "lsfhmwhw7fffrosff2lncwqcua"
    },
    "requestID": "36982d38-fcde-4bfe-a89a-7bd78d43c926",
    "eventID": "d7f0fc36-46ae-4157-9a79-c79f385fda98",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "444455556666",
    "eventCategory": "Management"
}
```

Memecahkan masalah

Gunakan informasi di bagian ini untuk membantu Anda memecahkan masalah kesalahan umum dalam aplikasi Modernisasi AWS Mainframe dan lingkungan runtime menggunakan mesin AWS Blu Age dan Micro Focus.

Topik

Kesalahan: Waktu habis sambil menunggu nama dataset dibuka kuncinya

- Mesin: AWS Blu Age
- Komponen: Blusam

Jika Anda melihat kesalahan ini di CloudWatch log Amazon untuk aplikasi Modernisasi AWS Mainframe menggunakan mesin AWS Blu Age dan berjalan di lingkungan dengan pola Ketersediaan Tinggi, ini menunjukkan bahwa aplikasi lain memegang kunci pada kumpulan data bersama. Biasanya, situasi ini terjadi jika aplikasi lain macet atau gagal dan tidak melepaskan kunci.

Bagaimana kesalahan ini terjadi

Aplikasi `example-app-1` mencoba mengunci catatan `example-record-1` untuk operasi tulis. Operasi ini membuat kunci pada dataset`example-dataset-1`, yang memiliki`example-record-1`, dan kunci pada `example-record-1` dirinya sendiri. Sekarang aplikasi lain,`example-app-2`, mencoba mengunci catatan yang sama`example-record-1`. Dataset dan catatan sudah terkunci, jadi `example-app-2` tunggu sampai kunci dilepaskan. Jika `example-app-1` mogok, kunci yang ditahan pada kumpulan data `example-dataset-1` masih ada, yang menyebabkan membatalkan upaya penulisannya dan `example-app-2` memunculkan pengecualian batas waktu. Situasi kebuntuan ini mencegah semua aplikasi mencapai`example-dataset-1`.

Bagaimana Anda tahu jika ini adalah situasi Anda?

Cari aplikasi yang gagal dan periksa apakah ia menggunakan kumpulan data yang sama yang disebutkan dalam pesan kesalahan. Periksa apakah aplikasi berjalan di lingkungan runtime dengan pola Ketersediaan Tinggi. Aplikasi yang mengangkat pengecualian batas waktu tidak dapat melanjutkan dan akan menampilkan `Failed` status.

Apa yang bisa kau lakukan?

Untuk menyelesaikan situasi segera, Anda dapat memaksa kunci untuk melepaskan. Untuk mencegah situasi serupa terjadi di masa depan, Anda dapat mengonfigurasi dua parameter yang mengontrol mekanisme perbaikan otomatis Blusam.

Paksa kunci untuk melepaskan

Manajer kunci Blusam menggunakan Amazon ElastiCache untuk Redis untuk menyediakan kunci bersama antar aplikasi. Untuk melepaskan kunci ElastiCache, gunakan utilitas Redis CLI. Anda tidak dapat menghapus kunci catatan individual. Anda harus menghapus semua kunci dari dataset yang dimiliki. Selesaikan langkah-langkah berikut:

1. Connect ke Anda ElastiCache menggunakan perintah berikut:

   ```
   redis-cli -h hostname -p port
   ```

   Anda dapat menemukan detail Anda ElastiCache di ElastiCache konsol di https://console.aws.amazon.com/elasticache/.
2. Masukkan kata sandi Anda.
3. Masukkan perintah yang ingin Anda jalankan, sebagai berikut:

Perintah	Tujuan
KEYS *	Dapatkan semua kunci yang ada.
KUNCI* *YOUR_DATASET_NAME*	Dapatkan kunci kunci dataset.
BAGIAN *THE_RETURNED_KEY*	Hapus kunci dataset.
FLUSHDB	Bersihkan seluruh Redis.

Perintah	Tujuan
	Warning Semua data dalam cache Redis akan hilang. Jika Redis digunakan untuk tujuan lain, seperti menangani sesi http, Anda mungkin tidak ingin menggunakannya`FLUSHDB`.

Konfigurasikan mekanisme perbaikan otomatis Blusam

Manajer kunci Blusam menyertakan mekanisme perbaikan otomatis untuk mencegah kebuntuan pada kumpulan data atau catatan. Anda dapat menyesuaikan parameter berikut dalam definisi aplikasi (`application-main.yml`) untuk mengonfigurasi mekanisme perbaikan otomatis:

- `locksDeadTime`: mengacu pada waktu maksimum aplikasi dapat menahan kunci. Ketika waktu ini berlalu, kunci dinyatakan kedaluwarsa dan segera dirilis. `locksDeadTime`Nilainya dalam milidetik, dan nilai defaultnya adalah 1000.
- `locksCheck`: mendefinisikan strategi manajer kunci Blusam untuk memeriksa kunci. Semua kunci Blusam diberi stempel waktu dan ElastiCache memiliki waktu kedaluwarsa. Nilai `locksCheck` parameter menentukan apakah kunci kedaluwarsa dihapus.
  - `off`: tidak ada pemeriksaan yang dijalankan kapan saja. Kebuntuan mungkin terjadi. (Tidak direkomendasikan)
  - `reboot`: pemeriksaan dijalankan ketika instance aplikasi Modernisasi AWS Mainframe yang berjalan di lingkungan runtime Modernisasi AWS Mainframe dimulai atau di-boot ulang. Semua kunci kedaluwarsa dilepaskan segera. (Default)
  - `timeout`: pemeriksaan dijalankan ketika instance aplikasi Modernisasi AWS Mainframe yang berjalan di lingkungan runtime Modernisasi AWS Mainframe dimulai atau di-boot ulang, atau ketika batas waktu berakhir selama upaya untuk mengunci kumpulan data. Kunci kedaluwarsa segera dilepaskan.

Untuk informasi lebih lanjut tentang definisi aplikasi untuk aplikasi AWS Blu Age, lihatAWS Sampel definisi aplikasi Blu Age.

Manajer kunci Blusam

Dalam konteks lingkungan runtime Modernisasi AWS Mainframe menggunakan pola Ketersediaan Tinggi, aplikasi AWS Blu Age mungkin digunakan beberapa kali. Untuk aplikasi yang menangani kumpulan data Blusam, masalah akses bersamaan mungkin terjadi. Manajer kunci Blusam memastikan integritas data dan mengelola akses baca dan tulis ke catatan dan kumpulan data dengan menyediakan kunci bersama antar aplikasi yang digunakan. ElastiCache Mekanisme ini memungkinkan lebih dari satu aplikasi untuk membaca catatan secara bersamaan, dan memastikan bahwa hanya satu aplikasi pada satu waktu yang menulis catatan.

Tulis kunci

Untuk memperbarui atau menghapus catatan tertentu, aplikasi harus terlebih dahulu mengunci kumpulan data yang memiliki catatan, lalu mengunci catatan itu sendiri. Saat catatan dikunci, kunci kumpulan data dilepaskan, dan catatan lain dari kumpulan data yang sama tersedia untuk digunakan. Ketika operasi pembaruan atau penghapusan selesai, kunci rekor yang ditahan dilepaskan. Hanya satu aplikasi pada satu waktu yang dapat memperbarui catatan, yang memblokir aplikasi lain dari membaca atau menulis hingga kunci dilepaskan, jika kebijakan aplikasi yang ditentukan memungkinkan menunggu rilis.

Baca kunci

Selama tidak ada kunci tulis yang disimpan pada catatan atau kumpulan data, beberapa aplikasi dapat membaca catatan yang sama pada saat yang bersamaan. Untuk mengunci catatan untuk operasi tulis, semua kunci baca harus dilepaskan.

> **Note**
>
> Manajer kunci Blusam menangani akses dari beberapa utas dalam aplikasi tertentu menggunakan mekanisme penguncian yang sama.

Tidak dapat mengakses URL aplikasi

- Mesin: Usia AWS Blu dan Fokus Mikro
- Komponen: aplikasi

Jika Anda tidak dapat mengakses URL untuk aplikasi Modernisasi AWS Mainframe yang sedang berjalan yang Anda buat dan gunakan ke lingkungan runtime Modernisasi AWS Mainframe, Anda mungkin perlu mengonfigurasi aturan masuk pada grup keamanan yang Anda kaitkan dengan lingkungan runtime.

Bagaimana kesalahan ini terjadi

Saat Anda membuat lingkungan runtime, grup keamanan yang Anda berikan, termasuk grup keamanan default, harus memiliki aturan masuk yang dikonfigurasi untuk mengizinkan lalu lintas ke aplikasi yang diterapkan dari luar VPC, jika Anda ingin mengizinkan jenis akses ini.

Bagaimana Anda tahu jika ini adalah situasi Anda?

Aplikasi dimulai dengan sukses dan berjalan normal, tetapi Anda tidak dapat terhubung menggunakan URL-nya.

Apa yang bisa kau lakukan?

Periksa apakah grup keamanan Amazon VPC yang terkait dengan lingkungan runtime memungkinkan lalu lintas ke lingkungan pada port aplikasi yang sesuai. Untuk memeriksa aturan grup keamanan, selesaikan langkah-langkah berikut:

1. Buka konsol Modernisasi AWS Mainframe di https://console.aws.amazon.com/m2/.
2. Di navigasi kiri, pilih Lingkungan.
3. Pilih lingkungan runtime yang menghosting aplikasi yang ingin Anda sambungkan.
4. Pilih Konfigurasi.
5. Di Keamanan & Jaringan, pilih grup keamanan. Tautan membuka detail grup keamanan di konsol VPC Amazon.
6. Jika perlu, pilih Edit aturan masuk dan tambahkan aturan berikut jika belum ada:

   Tipe

   TCP Kustom

   Port

   8196 atau port yang cocok dengan properti listener yang ditentukan dalam definisi aplikasi. Untuk informasi selengkapnya, lihat Langkah 2: Buat definisi aplikasi.

Sumber

Alamat IP dari tempat Anda memanggil aplikasi. Anda dapat memilih MyIP dari dropdown. Jika Anda masih memiliki masalah batas waktu, coba pilih Anywhere IPV4 atau Anywhere IPV6. Pastikan untuk menghentikan aplikasi dan memulainya lagi setelah Anda menambahkan aturan masuk pada grup keamanan.

Untuk informasi selengkapnya, lihat Bekerja dengan aturan grup keamanan di Panduan Pengguna Amazon VPC.

AWSBlu Insights tidak terbuka dari konsol

- Mesin: AWS Blu Age
- Komponen: Blu Insights

Saat Anda mencoba mengakses Blu Insights dari konsol Modernisasi AWS Mainframe, itu tidak terbuka dan tab baru segera ditutup.

Bagaimana kesalahan ini terjadi

Peran yang Anda gunakan untuk mengakses Blu Insights tidak memiliki izin yang memadai.

Apa yang bisa kau lakukan?

Lampirkan kebijakan IAM ke peran untuk memungkinkannya mengakses Blu Insights. Pastikan kebijakan tersebut mencakup setidaknya izin berikut.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "m2:GetSignedBluinsightsUrl"
            ],
            "Resource": "arn:aws:m2:region:account:*"
            }
    ]
```

```
}
```

Pastikan untuk mengganti `region` dan `account` dengan yang benar Wilayah AWS danAkun AWS.

Riwayat dokumen untuk Panduan Pengguna Modernisasi AWS Mainframe

Tabel berikut menjelaskan rilis dokumentasi untuk Modernisasi AWS Mainframe.

Perubahan	Deskripsi	Tanggal
Update Managed Runtime untuk tutorial Micro Focus	Tutorial ini menunjukkan cara menerapkan dan menjalankan aplikasi CardDemo sampel dalam lingkungan runtime terkelola Modernisasi AWS Mainframe dengan mesin runtime Micro Focus.	Februari 5, 2024
Catatan rilis untuk AWS Blu Age Runtime dan Modernization Tools versi 3.9.0.	Rilis AWS Blu Age Runtime dan Modernization Tools ini difokuskan pada beberapa peningkatan transversal di seluruh produk yang berusaha meningkatkan kinerja dalam arsitektur ketersediaan tinggi, bersama dengan kemampuan baru untuk meningkatkan eksekusi pekerjaan ke tingkat berikutnya.	18 Desember 2023
Transfer file antara mainframe dan AWS	Fitur baru dirilis untuk mentransfer file dari mainframe sumber keAWS.	27 November 2023
Mengelola transaksi untuk aplikasi	Fitur baru dirilis untuk menampilkan dan mengedit transaksi untuk aplikasi untuk Modernisasi AWS Mainframe.	16 Oktober 2023

Catatan rilis untuk AWS Blu Age Runtime dan Modernization Tools versi 3.6.0.	Rilis AWS Blu Age Runtime dan Modernization Tools ini menyediakan fitur baru untuk migrasi lama ZoS dan AS400, terutama berorientasi pada perluasan mekanisme dukungan CICS, melengkapi kemampuan JCL, mengoptimalkan kinerja dalam fitur bersamaan dan volume tinggi, dan menambahkan kemampuan. multi-data-source	4 Agustus 2023
Anda sekarang dapat menerapkan versi baru aplikasi saat aplikasi dihentikan.	Sebelumnya, untuk menyebarkan versi baru aplikasi, Anda harus menghapus versi yang digunakan. Sekarang Anda bisa menghentikan versi yang diterapkan dan menerapkan versi baru.	26 Juli 2023
AWSBlu Age runtime dikemas untuk penyebaran Amazon EC2 yang lebih mudah	AWSModernisasi Mainframe dengan runtime AWS Blu Age kini tersedia dengan lebih banyak fleksibilitas untuk mengonfigurasi tumpukan dan penerapan lengkap pada instans Amazon EC2 di instans Amazon EC2. Akun AWS	6 Juli 2023
Masuk tunggal ke Blu Age AWSAWS Blu Insights.	AWSAWSBlu Age Blu Insights tersedia dari AWS Management Console melalui single sign-on.	31 Maret 2023

Rilis GA	Rilis GA dari Panduan Pengguna Modernisasi AWS Mainframe.	Juni 8, 2022
Rilis awal	Rilis awal (pratinjau publik) dari Panduan Pengguna Modernisasi AWS Mainframe.	30 November 2021

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.