Manual do usuário

# AWS Modernização do mainframe

# AWS Modernização do mainframe: Manual do usuário

# Table of Contents

# O que é modernização AWS do mainframe?

AWS A modernização do mainframe ajuda você a modernizar seus aplicativos de mainframe para AWS ambientes de tempo de execução gerenciados. Ele fornece ferramentas e recursos para ajudar você a planejar e implementar a migração e a modernização. Você pode analisar seus aplicativos de mainframe existentes, desenvolvê-los ou atualizá-los usando COBOL ou PL/I e implementar um pipeline automatizado para integração contínua e entrega contínua (CI/CD) dos aplicativos. Você pode escolher entre padrões automatizados de refatoração e replataforma, dependendo das necessidades de seus clientes. Se você é um consultor ajudando um cliente a migrar suas cargas de trabalho de mainframe, você pode usar as ferramentas de modernização de AWS mainframe em todas as fases da jornada de migração e modernização, desde o planejamento inicial até as operações de nuvem pós-migração.

Você pode usar a modernização do AWS mainframe para ajudá-lo a criar e gerenciar com eficiência o ambiente de execução de seus aplicativos de mainframe, bem como para gerenciar e monitorar seus aplicativos modernizados. AWS

Tópicos



> **Note**
>
> Você se envolveu com parceiros de competência em migração de AWS mainframe ou serviços AWS profissionais para seu projeto de modernização de mainframe? Caso contrário, é altamente recomendável que você contrate especialistas para o seu projeto.
>
> - AWS Parceiros de competência em modernização de mainframe
> - AWS Professional Services

Os recursos e os casos de uso da modernização do AWS mainframe oferecem suporte a uma abordagem de modernização evolutiva, que oferece ganhos de curto prazo ao melhorar a agilidade e muitas oportunidades para otimizar e inovar posteriormente. Para ter mais informações, consulte Abordagem de modernização.

# Características da modernização do AWS mainframe

AWS Os recursos de modernização do mainframe oferecem suporte aos seguintes casos de uso:

- Avalie: o recurso de avaliação da modernização do AWS mainframe pode ajudá-lo a avaliar, definir o escopo e planejar um projeto de migração e modernização.
- Refatoração: com tecnologia AWS Blu Age, você pode usar a refatoração para converter linguagens de programação de aplicativos legadas, criar macrosserviços ou microsserviços e modernizar interfaces de usuário (UIs) e pilhas de software de aplicativos.

  AWS O Blu Insights agora está disponível a AWS Management Console partir do login único. Você não precisa mais gerenciar credenciais separadas do AWS Blu Insights. Você pode acessar os recursos do AWS AWS Blu Age Codebase e do Transformation Center diretamente do. AWS Management Console
- Replataforma: com a solução Micro Focus Enterprise, você pode portar o aplicativo para o qual grande parte do código-fonte do aplicativo é recompilado sem alterações.
- IDE do desenvolvedor: a modernização do AWS mainframe oferece um ambiente de desenvolvimento integrado (IDE) sob demanda para que os desenvolvedores possam escrever código mais rapidamente com edição e depuração inteligentes, compilação instantânea de código e testes unitários.
- Tempo de execução gerenciado: o ambiente de execução gerenciada da modernização do AWS mainframe monitora continuamente seus clusters para manter as cargas de trabalho corporativas funcionando com computação autorrecuperável e escalabilidade automatizada.
- Integração e entrega contínuas (CI/CD): o recurso CI/CD da AWS Mainframe Modernization ajuda as equipes de desenvolvimento de aplicativos a realizar alterações de código com mais frequência e confiabilidade, o que acelera a velocidade de migração, aumenta a qualidade e ajuda a reduzir o lançamento de novas funções de negócios. time-to-market
- Integrações com outros AWS serviços: a modernização do AWS mainframe oferece suporte AWS CloudFormation AWS PrivateLink, e AWS Key Management Service para implantação repetível e maior segurança e conformidade.

- Disponibilidade ampliada: a modernização do AWS mainframe agora está disponível no Leste dos EUA (Ohio), Oeste dos EUA (Norte da Califórnia), Ásia-Pacífico (Mumbai), Ásia-Pacífico (Seul), Ásia-Pacífico (Cingapura), Ásia-Pacífico (Tóquio), Europa (Londres) e Europa (Paris).

Para obter mais informações sobre os recursos de modernização do AWS mainframe, consulte. https://aws.amazon.com/mainframe-modernization/features/

# Padrões

O padrão Automated Refactoring, desenvolvido pela AWS Blu Age, está focado em acelerar a modernização, convertendo a pilha completa de aplicativos legados e sua camada de dados em um aplicativo moderno baseado em Java, preservando a equivalência funcional. Durante essa transformação automatizada, ele cria um aplicativo de várias camadas com um front-end baseado em Angular, um back-end Java habilitado para API e uma camada de dados que acessa armazenamentos de dados modernos. O processo de refatoração fornece funcionalidade equivalente à pilha antiga para aumentar a automação do projeto, resultando em velocidade, qualidade e menor custo para obter benefícios comerciais mais rapidamente. Para obter mais informações, consulte AWS Mainframe Modernization Automated Refactor.

O padrão de replataforma, desenvolvido pela suíte Micro Focus Enterprise, tem como foco preservar a linguagem, o código e os artefatos do aplicativo a fim de minimizar o impacto nos ativos e nas equipes do aplicativo. Ele ajuda os clientes a manter o conhecimento e as habilidades do aplicativo. Embora as mudanças nos aplicativos sejam limitadas, esse padrão também facilita a modernização da infraestrutura e dos processos. A infraestrutura é alterada para um serviço gerenciado moderno baseado em nuvem, enquanto os processos são alterados para seguir as melhores práticas de desenvolvimento de aplicativos e operações de TI. Para obter mais informações, consulte AWS Mainframe Modernization Replatform.

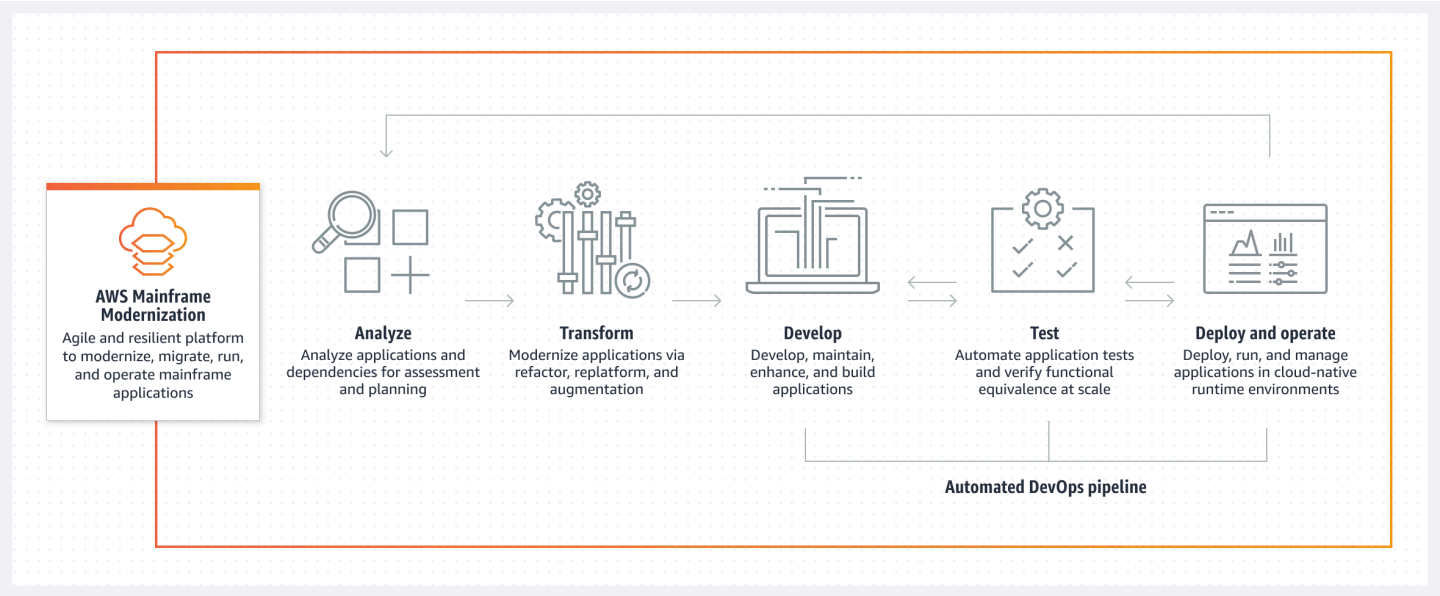
# Como começar a modernizar o AWS mainframe

Experimente! Oferecemos tutoriais e exemplos de aplicativos para ajudar você a ter uma ideia do que a modernização do AWS mainframe oferece. Escolha o Tutorial: Tempo de execução gerenciado para AWS Blu Age ou o Tutorial: Tempo de execução gerenciado para Micro Focus para obter um step-by-step tutorial completo.

Se você estiver interessado em refatoração automatizada, confira as ferramentas do AWS Blu Age em. BluInsights Você também pode configurar o AppStream 2.0 para acessar o AWS Blu

Age Developer IDE ou as ferramentas Micro Focus Enterprise Analyzer e Micro Focus Enterprise Developer.

Os tutoriais e os aplicativos de amostra dão apenas uma ideia do que a modernização do AWS mainframe oferece. Quando você estiver pronto para iniciar um projeto de modernização, consulte Abordagem de modernização para obter detalhes sobre os estágios e tarefas de um projeto de modernização.




# Serviços relacionados

Além do Blu Insights para refatoração automatizada, você pode usar os seguintes AWS serviços com a modernização do mainframe. AWS

- Amazon RDS para hospedar seus bancos de dados migrados.
- Amazon S3 para armazenar binários de aplicativos e arquivos de definição.
- Amazon FSx ou Amazon EFS para armazenar dados de aplicativos.
- Amazon AppStream para acesso às ferramentas Micro Focus Enterprise Analyzer e Micro Focus Enterprise Developer.
- AWS CloudFormation para o DevOps pipeline automatizado que você pode usar para configurar o CI/CD para seus aplicativos migrados.
- AWS Migration Hub
- AWS DMS para migrar seus bancos de dados.

# Acessando a AWS modernização do mainframe

Atualmente, você pode acessar a Modernização do AWS Mainframe por meio do console em https://console.aws.amazon.com/m2/. Para obter uma lista das regiões em que a modernização de AWS mainframe está disponível, consulte endpoints e cotas de modernização de AWS mainframe no. Referência geral da Amazon Web Services

## Você é um usuário iniciante de modernização de AWS mainframe?

Se você é um usuário iniciante da modernização de AWS mainframe, recomendamos que comece lendo as seguintes seções:

- Conceitos Básicos
- Configuração

## Definição de preço

AWS A modernização do mainframe cobra pelo uso de instâncias que dão suporte aos ambientes de tempo de execução gerenciados. Além disso, a modernização do AWS mainframe oferece algumas ferramentas sem custos adicionais. Você é responsável pelas taxas incorridas por outros AWS serviços que você usa em conexão com a modernização do AWS mainframe. AWS fornecerá um aviso prévio de 30 dias antes que qualquer alteração de preço entre em vigor para o uso da modernização do AWS mainframe. Para obter mais informações, consulte Modernização de mainframe com. AWS

Com o AWS Blu Insights, você paga pelo uso do Transformation Center. Para obter mais informações, consulte Preço do AWS Mainframe Modernization.

# Configurando a modernização AWS do mainframe

Antes de começar a usar a Modernização do AWS Mainframe, você ou seu administrador precisam concluir algumas etapas.

Tópicos



## Inscreva-se para um Conta da AWS

Se você não tiver um Conta da AWS, conclua as etapas a seguir para criar um.

Para se inscrever em um Conta da AWS

1. Acesse https://portal.aws.amazon.com/billing/signup.
2. Siga as instruções on-line.

   Durante a criação da conta, você vai receber uma chamada telefônica e inserir um código de verificação no teclado do telefone.

   Quando você se inscreve em um Conta da AWS, um Usuário raiz da conta da AWSé criado. O usuário raiz tem acesso a todos os Serviços da AWS e atributos na conta. Como prática recomendada de segurança, atribua acesso administrativo a um usuário administrativo e utilize somente o usuário raiz para executar as tarefas que exigem acesso do usuário raiz.

AWS envia um e-mail de confirmação após a conclusão do processo de inscrição. A qualquer momento, é possível visualizar as atividades da conta atual e gerenciar sua conta acessando https://aws.amazon.com/ e selecionando Minha conta.

## Criar um usuário administrativo

Depois de se inscrever em um Conta da AWS, proteja seu Usuário raiz da conta da AWS AWS IAM Identity Center, habilite e crie um usuário administrativo para que você não use o usuário root nas tarefas diárias.

### Proteja seu Usuário raiz da conta da AWS

1. Faça login AWS Management Consolecomo proprietário da conta escolhendo Usuário raiz e inserindo seu endereço de Conta da AWS e-mail. Na próxima página, insira a senha.

   Para obter ajuda ao fazer login usando o usuário raiz, consulte Fazer login como usuário raiz no Guia do usuário do Início de Sessão da AWS .
2. Ative a autenticação multifator (MFA) para o usuário raiz.c

   Para obter instruções, consulte Habilitar um dispositivo de MFA virtual para seu usuário Conta da AWS raiz (console) no Guia do usuário do IAM.

### Criar um usuário administrativo

1. Habilitar o IAM Identity Center.

   Para obter instruções, consulte Habilitar AWS IAM Identity Center no Guia do usuário do AWS IAM Identity Center .
2. No Centro de Identidade do IAM, conceda acesso administrativo a um usuário administrativo.

   Para ver um tutorial sobre como usar o Diretório do Centro de Identidade do IAM como fonte de identidade, consulte Configurar o acesso do usuário com o padrão Diretório do Centro de Identidade do IAM no Guia AWS IAM Identity Center do usuário.

### Login como usuário administrativo

- Para fazer login com o usuário do Centro de Identidade do IAM, utilize o URL de login enviado ao seu endereço de e-mail quando você criou o usuário do Centro do Usuário do IAM.

  Para obter ajuda para fazer login usando um usuário do IAM Identity Center, consulte Como fazer login no portal de AWS acesso no Guia Início de Sessão da AWS do usuário.

# Introdução à modernização AWS do mainframe

Para começar a usar a modernização do AWS mainframe, você pode seguir os tutoriais que apresentam o serviço e cada mecanismo de tempo de execução.

Tópicos



Para continuar aprendendo, consulte os tutoriais a seguir.



## Tutorial: Tempo de execução gerenciado para AWS Blu Age

Este tutorial mostra como implantar um aplicativo modernizado do AWS Blu Age em um ambiente de tempo de execução do AWS Mainframe Modernization.

Tópicos

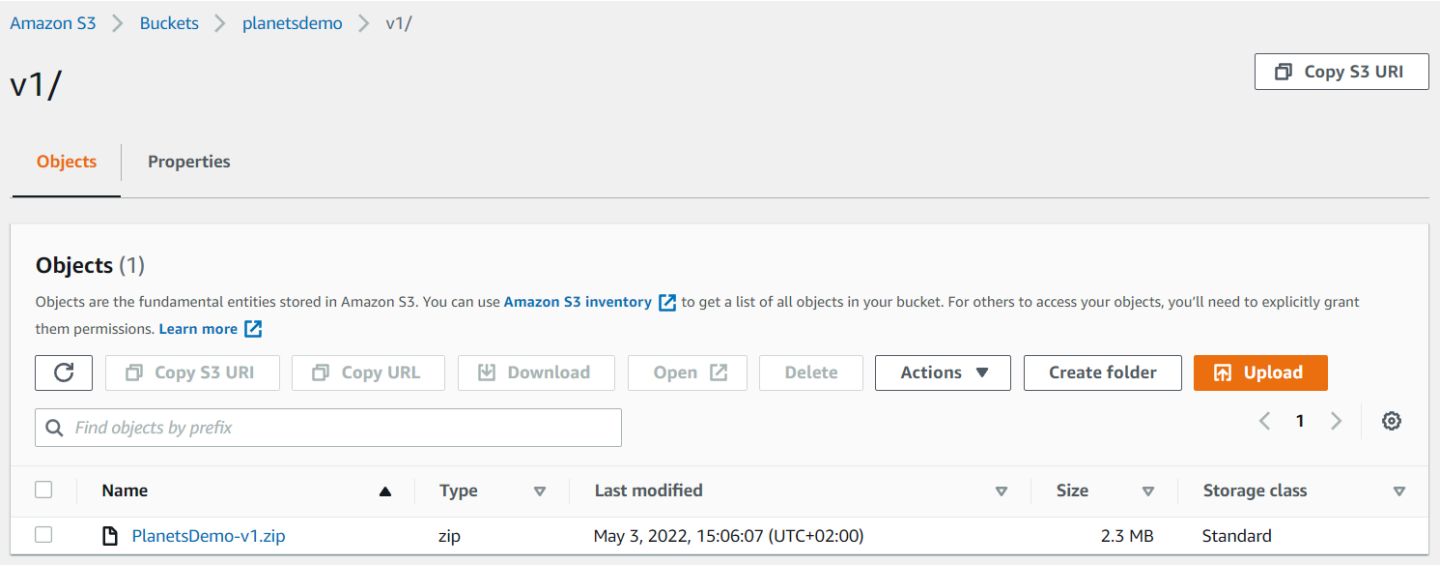
## Pré-requisitos

Para concluir este tutorial, baixe o arquivo de aplicativos de demonstração PlanetsDemo-v1.zip.

O aplicativo de demonstração em execução requer um navegador moderno para acesso. Se você executa esse navegador a partir do seu desktop ou de uma instância do Amazon Elastic Compute Cloud, por exemplo, dentro da VPC, determina suas configurações de segurança.

## Etapa 1: Faça o upload do aplicativo de demonstração

Faça upload do aplicativo de demonstração em um bucket do Amazon S3. Certifique-se de que esse bucket esteja no mesmo Região da AWS local em que você implantará o aplicativo. O exemplo a seguir mostra um bucket chamado planetsdemo, com um prefixo de chave, ou pasta, chamado v1 e um arquivo chamado `planetsdemo-v1.zip`.




> **Note**
>
> A pasta no bucket é obrigatória.

## Etapa 2: Criar o aplicativo

Para implantar um aplicativo no tempo de execução gerenciado, você precisa de uma definição de aplicativo do AWS Mainframe Modernization Essa definição é um arquivo JSON que descreve a localização e as configurações do aplicativo. O exemplo a seguir é uma definição de aplicativo desse tipo para o aplicativo de demonstração:

```
{
    "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"
        }
    }
}
```

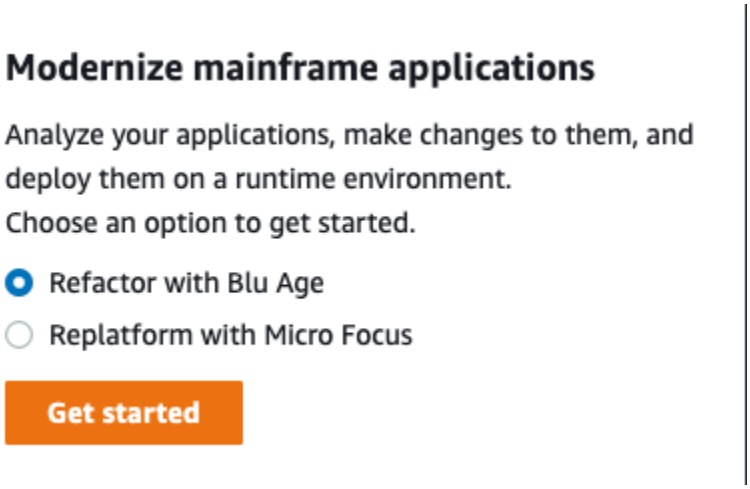
Altere a `s3-bucket` entrada para o nome do bucket em que você armazenou o arquivo zip do aplicativo de amostra.

Para obter mais informações sobre a definição do aplicativo, consulte AWS Exemplo de definição do aplicativo Blu Age.

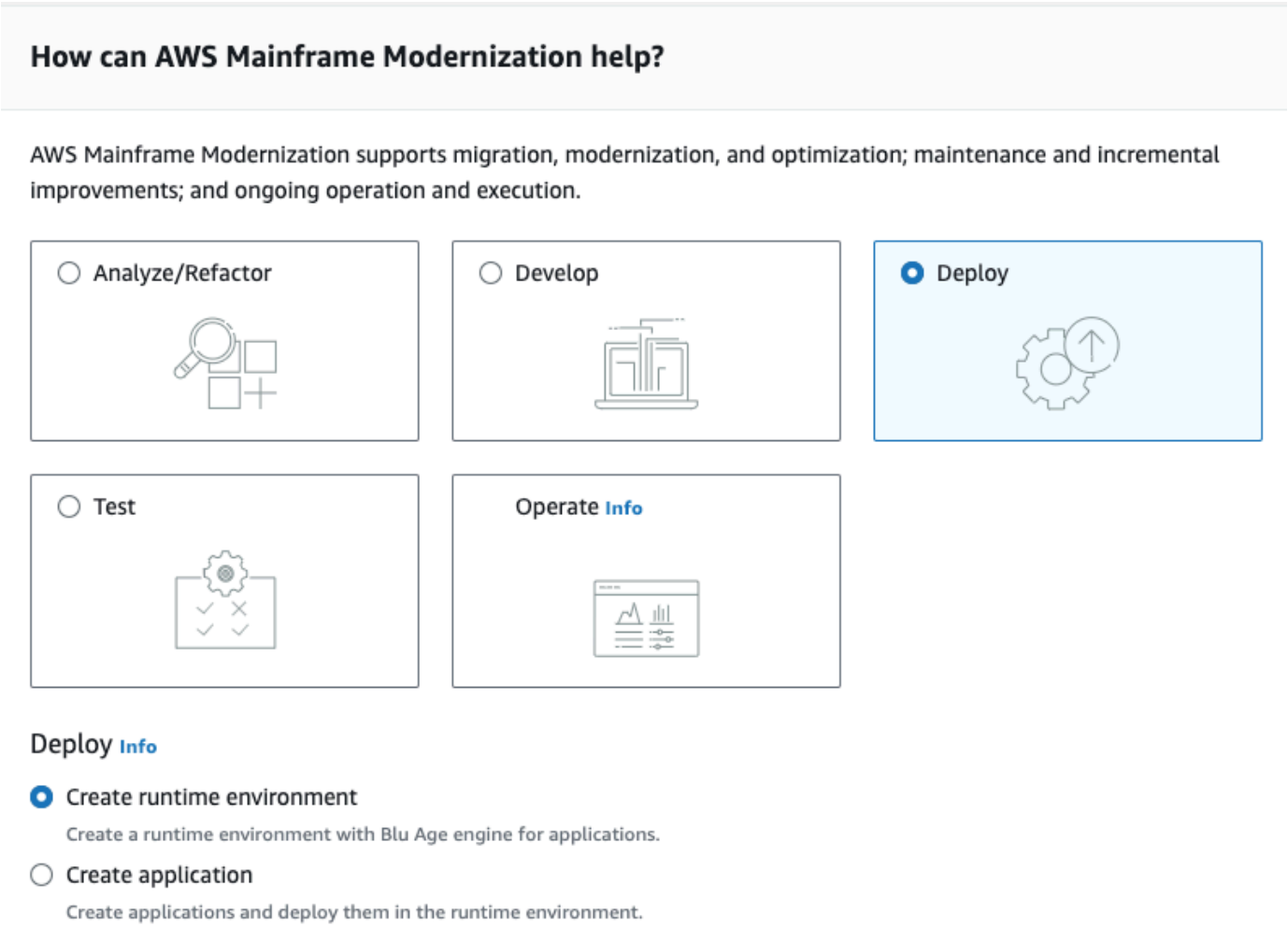
## Etapa 3: Criar um ambiente de tempo de execução

Para criar o ambiente de tempo de execução do AWS Mainframe Modernization, execute as seguintes etapas:

1. Abra o console do AWS Mainframe Modernization.
2. No seletor Região da AWS, escolha a região onde você deseja criar o ambiente. Isso Região da AWS deve corresponder à região em que você criou o bucket do S3. Etapa 1: Faça o upload do aplicativo de demonstração
3. Em Modernizar aplicativos de mainframe, escolha Refatorar com Blu Age e, em seguida, escolha Começar.

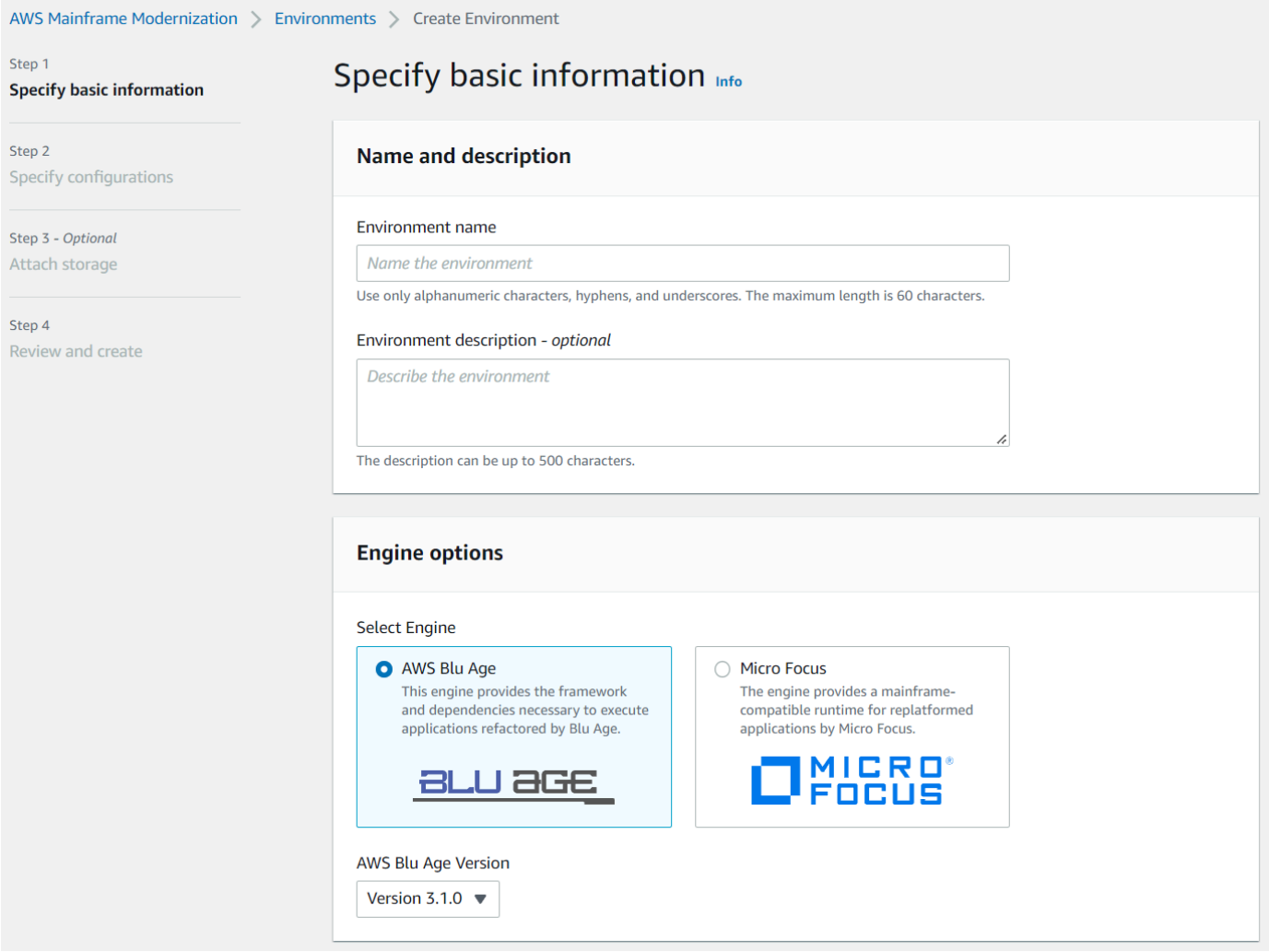



4. Em Como o AWS Mainframe Modernization pode ajudar, escolha Implantar e Criar ambiente de runtime.

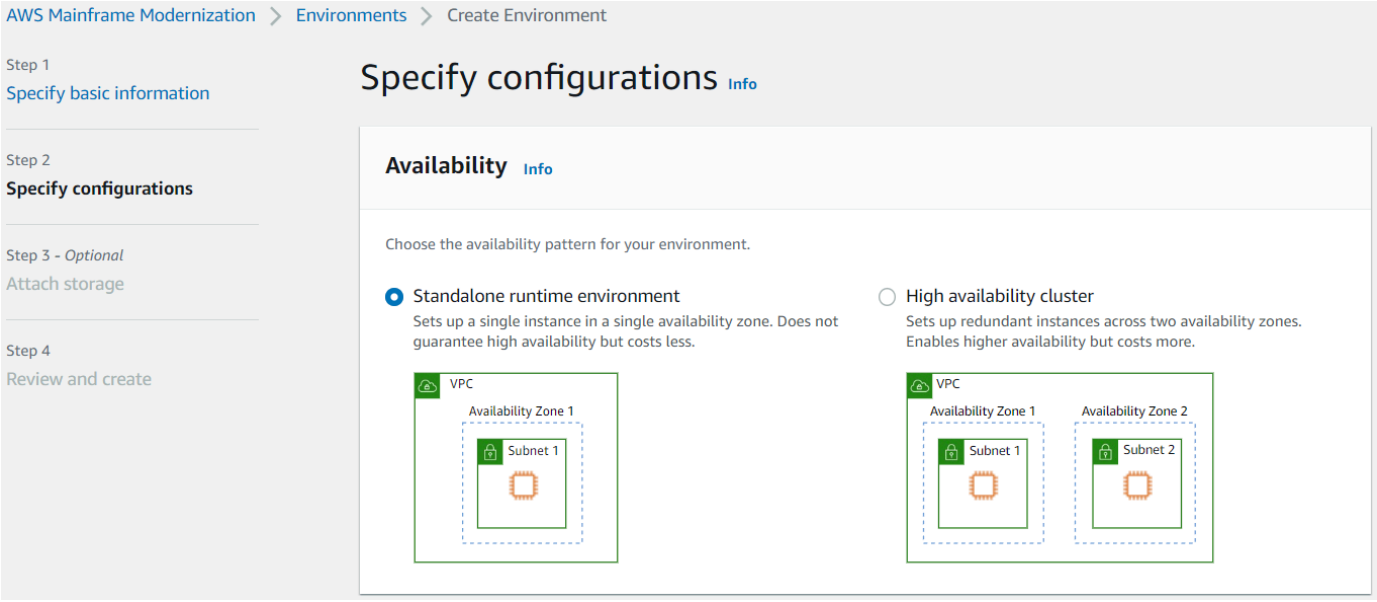



5. No painel de navegação, escolha Ambientes de computação, Criar ambiente. Na página Especificar informações básicas, insira um nome e uma descrição para seu ambiente e, em

seguida, certifique-se de que o mecanismo AWS Blu Age esteja selecionado. Opcionalmente, você pode adicionar etiquetas ao recurso criado. Em seguida, escolha Próximo.

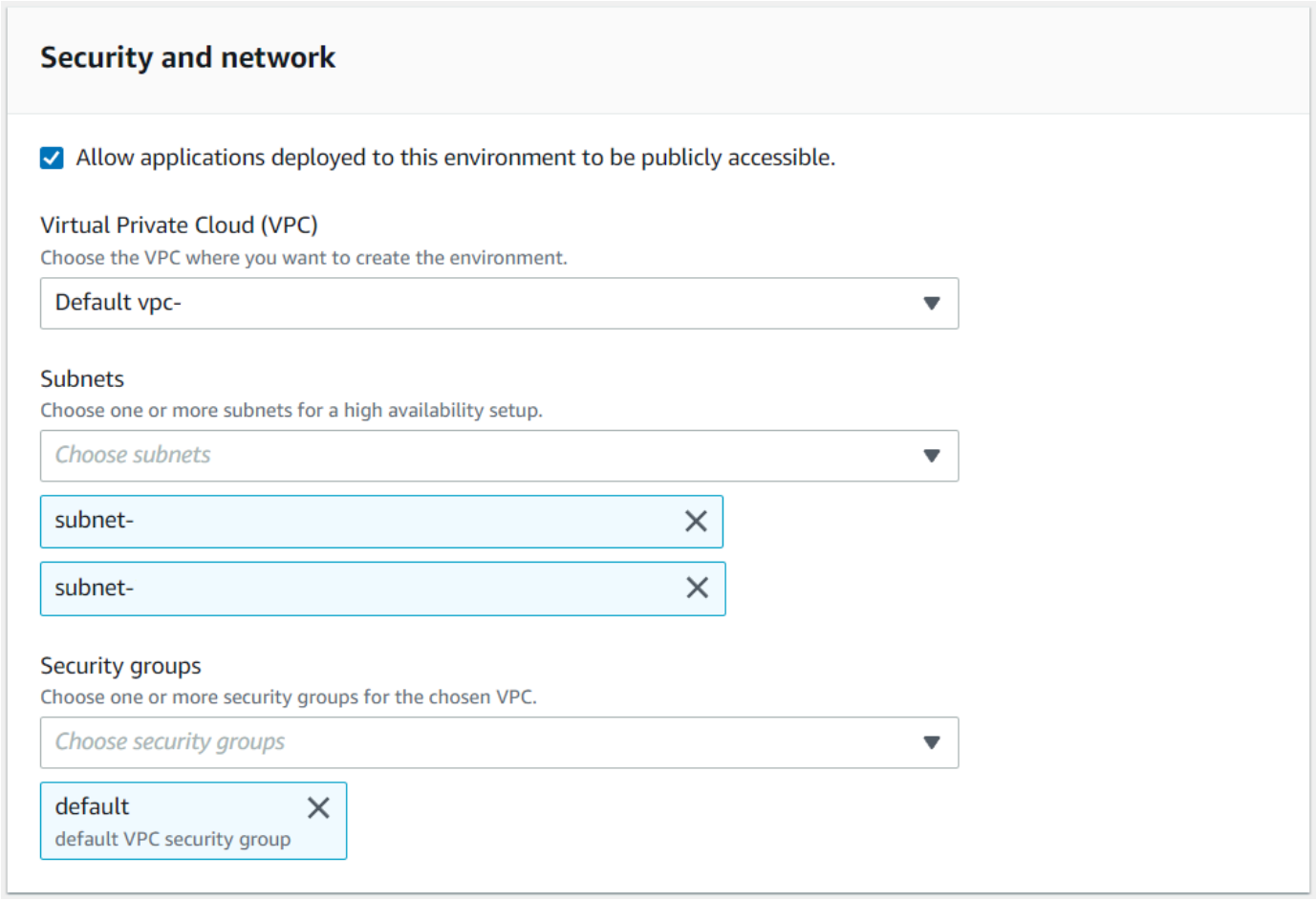



6. Na página Especificar configurações, escolha Ambiente de tempo de execução autônomo.



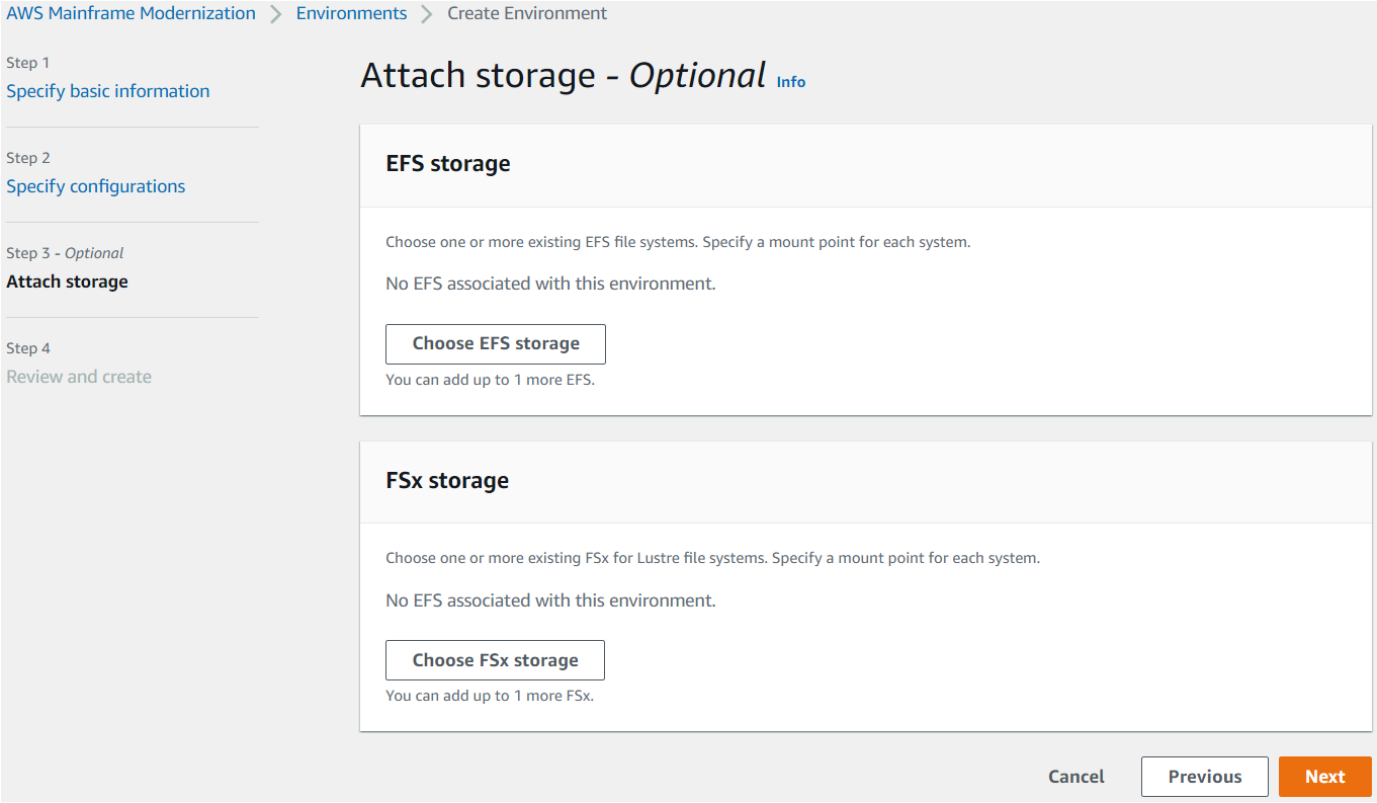

7. Em Segurança e rede, faça as seguintes alterações:

   - Escolha Permitir que os aplicativos implantados nesse ambiente sejam acessíveis ao público. Essa opção atribui um endereço IP público ao aplicativo para que você possa acessá-lo do seu desktop.
   - Escolha uma VPC. Você pode usar o Padrão.
   - Escolha duas sub-redes. Certifique-se de que as sub-redes permitam a atribuição de endereços IP públicos.
   - Escolher grupo de segurança Você pode usar o Padrão. Certifique-se de que o grupo de segurança escolhido permita o acesso do endereço IP do navegador à porta especificada na `listener` propriedade da definição do aplicativo. Para ter mais informações, consulte Etapa 2: Criar o aplicativo .




Se você quiser acessar o aplicativo de fora da VPC escolhida, certifique-se de que as regras de entrada dessa VPC estejam configuradas corretamente. Para ter mais informações, consulte Não é possível acessar o URL de um aplicativo.

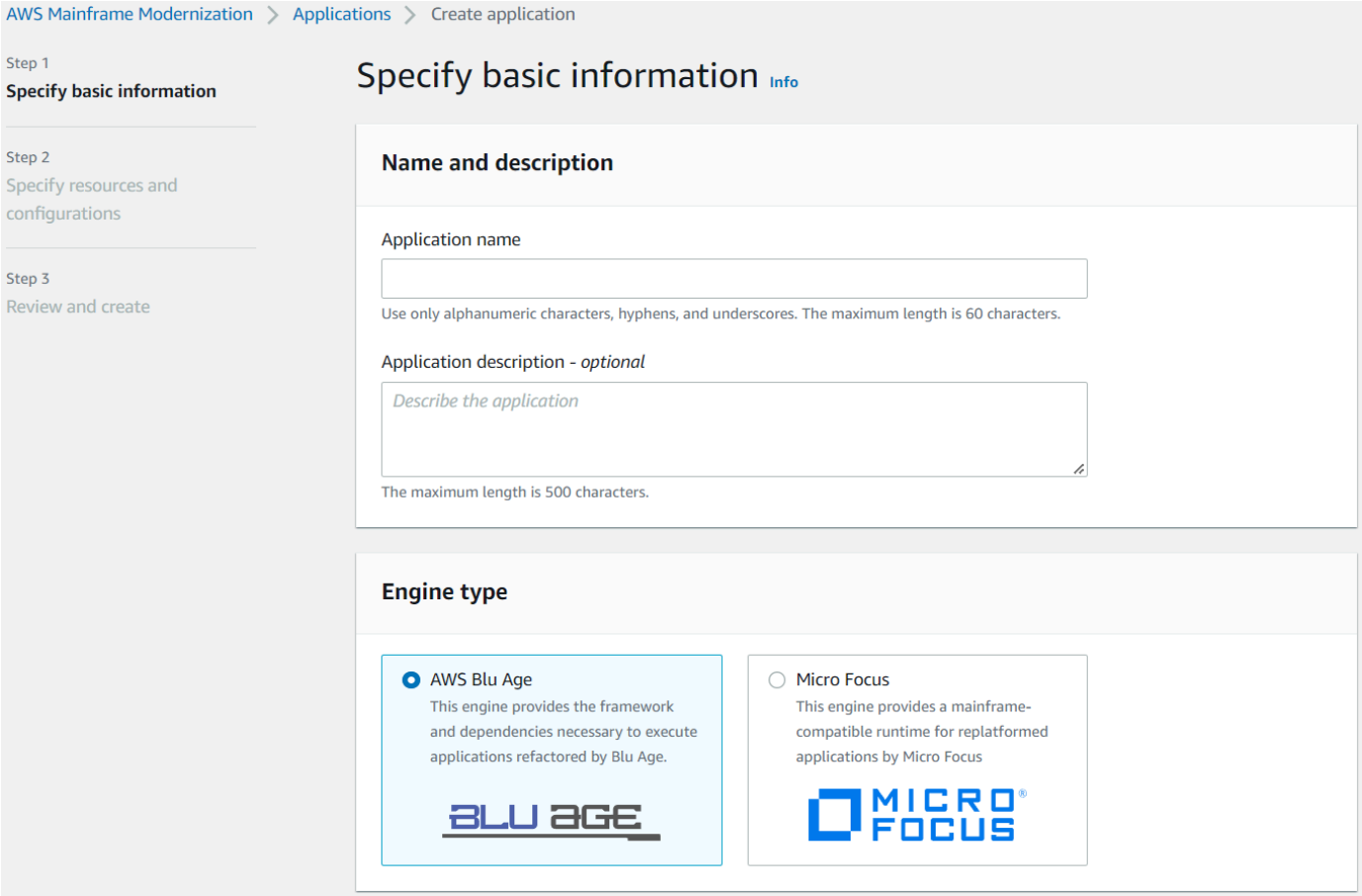
8. Escolha Próximo.
9. Em Anexar armazenamento - Opcional, deixe as seleções padrão e escolha Próximo.



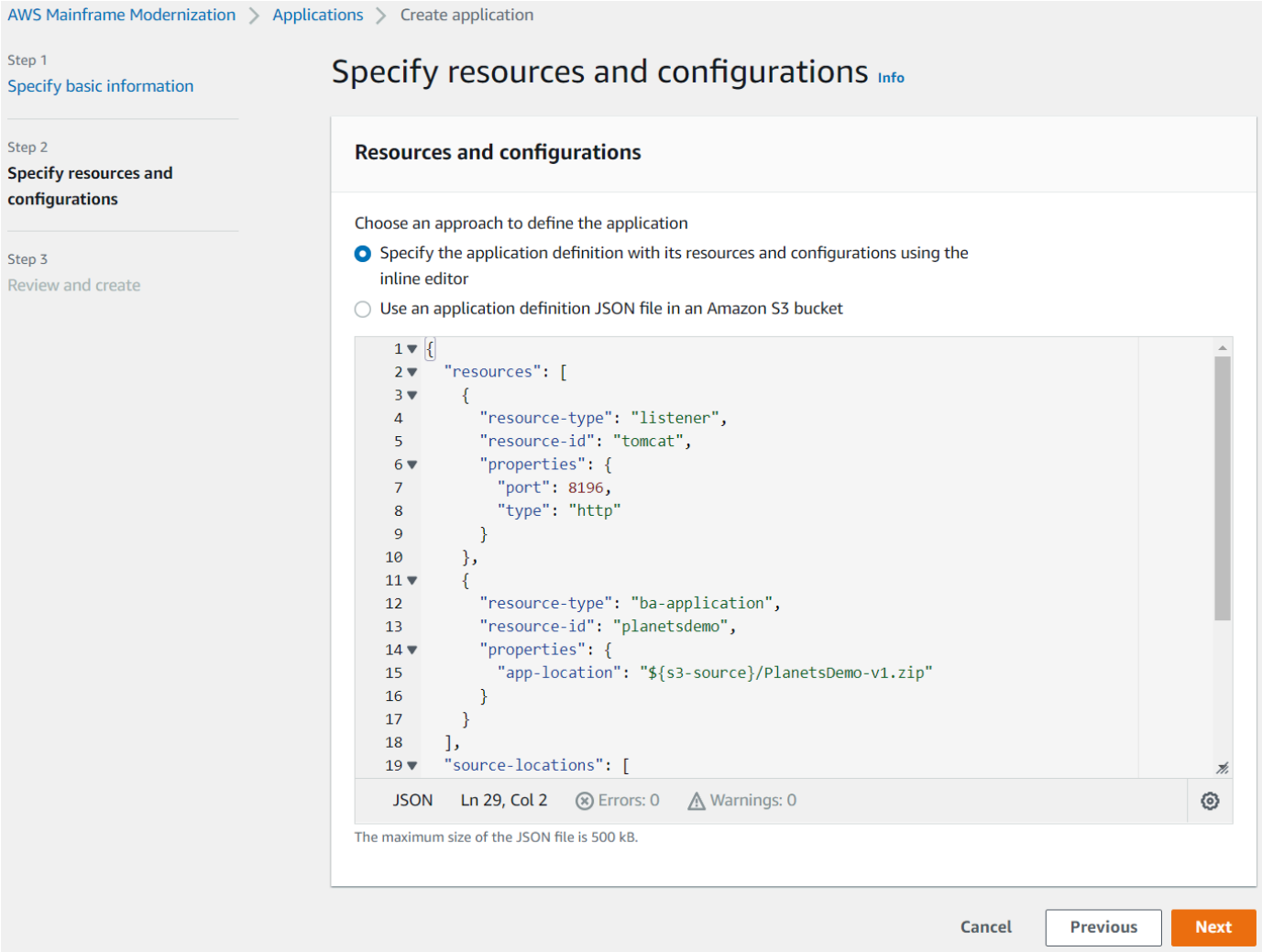

10. Em Agendar manutenção, escolha Sem preferência e, em seguida, escolha Próximo.
11. Em Revisar e criar, revise as informações e escolha Criar ambiente.

## Etapa 4: Criar um aplicativo

1. Navegue até a AWS Mainframe Modernization no AWS Management Console.
2. No painel de navegação, escolha Aplicativos e Criar uma nova aplicação. Na página Especificar informações básicas, insira um nome e uma descrição para o aplicativo e certifique-se de que o mecanismo AWS Blu Age esteja selecionado. Em seguida, escolha Próximo.




3. Na página Especificar recursos e configurações, copie e cole o JSON de definição da aplicação atualizado que você criou no the section called “Etapa 2: Criar o aplicativo ”.

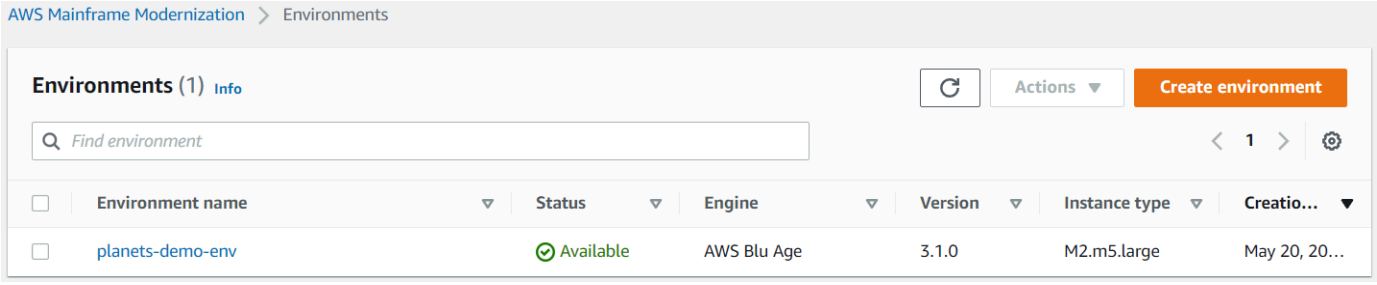



4. Na página Revisar e criar, revise suas escolhas e, em seguida, selecione Criar aplicação.

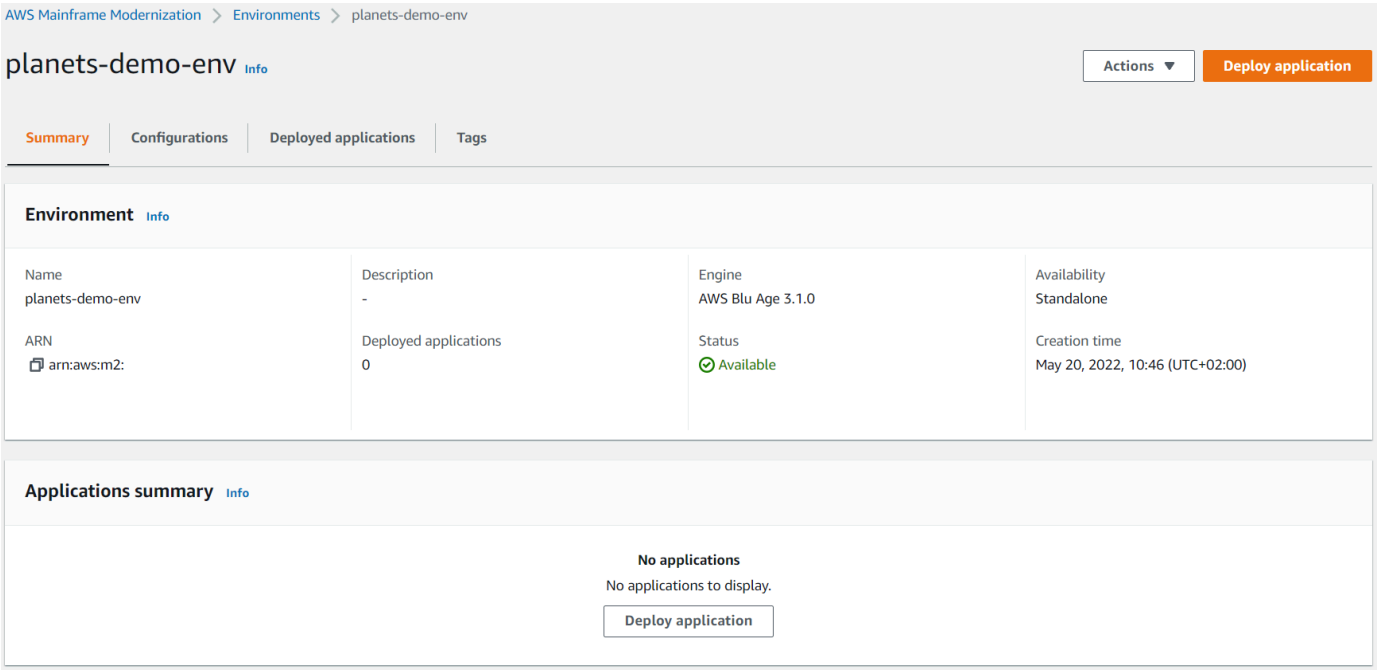
## Etapa 5: Implantar um aplicativo

Depois de criar com êxito o ambiente de tempo de execução e o aplicativo do AWS Mainframe Modernization, e ambos estiverem no estado Disponível, você poderá implantar o aplicativo no ambiente de tempo de execução. Para fazer isso, conclua as seguintes etapas:

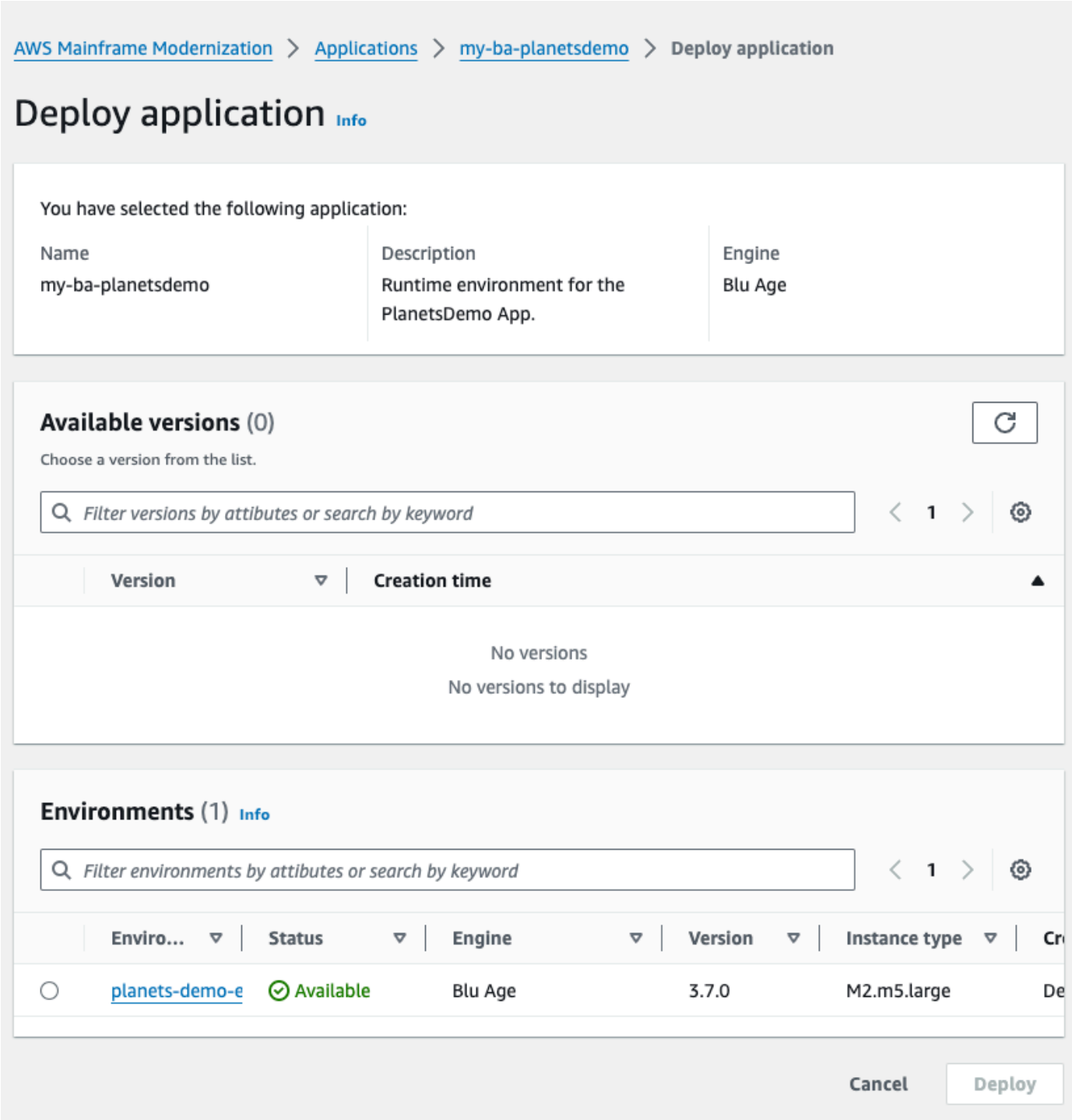
1. Navegue até o AWS Mainframe Modernization no console de gerenciamento AWS. No painel de navegação, escolha Ambientes. A página da lista de ambientes é exibida.




2. Escolha o ambiente de execução criado anteriormente. A página do ambiente é exibida.
3. Escolha Implantar aplicativo.



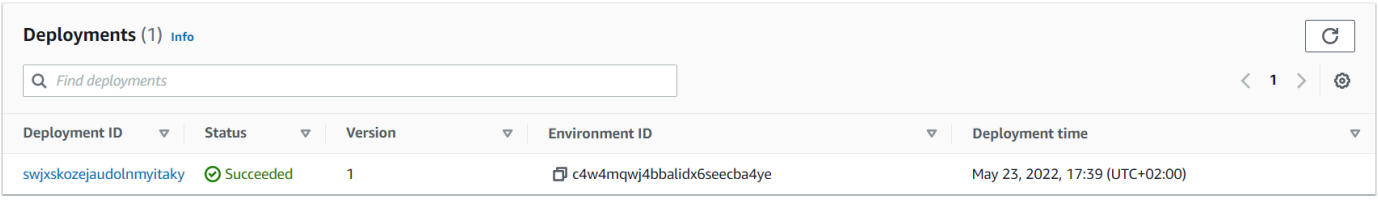

4. Escolha a aplicação criada anteriormente e, em seguida, escolha a versão na qual você deseja implantar a aplicação. Selecione Deploy (Implantar).




5. Espere até que o aplicativo conclua sua implantação. Você verá um banner com a mensagem O aplicativo foi implantado com sucesso.

## Etapa 9: Inicie e teste o aplicativo

1. Navegue até a AWS Mainframe Modernization no .
2. Vá até a página do aplicativo e escolha Implantar. O status da aplicação deve ser Sucedido.



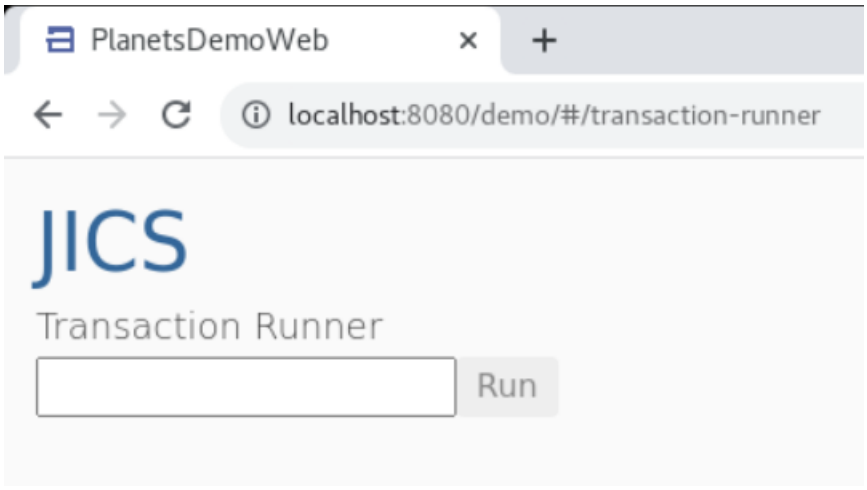

3. Escolha Ações e Excluir aplicativo.

## Etapa 7: acessar o aplicativo

1. Espere até que o aplicativo esteja no estado Executando. Você verá um banner com a mensagem O aplicativo foi iniciado com sucesso.
2. Copie o nome de host DNS do aplicativo. Você pode encontrar esse nome de host na seção Informações do aplicativo.
3. Em um navegador, navegue até `http://{hostname}:{portname}/PlanetsDemo-web-1.0.0/`, onde:

   - `hostname` é o nome do host DNS copiado anteriormente.
   - `portname` é a porta Tomcat definida na definição do aplicativo em Etapa 2: Criar o aplicativo que você criou.

   A tela JICS é exibida.



Se você não conseguir acessar o aplicativo, consulte Não é possível acessar o URL de um aplicativo.

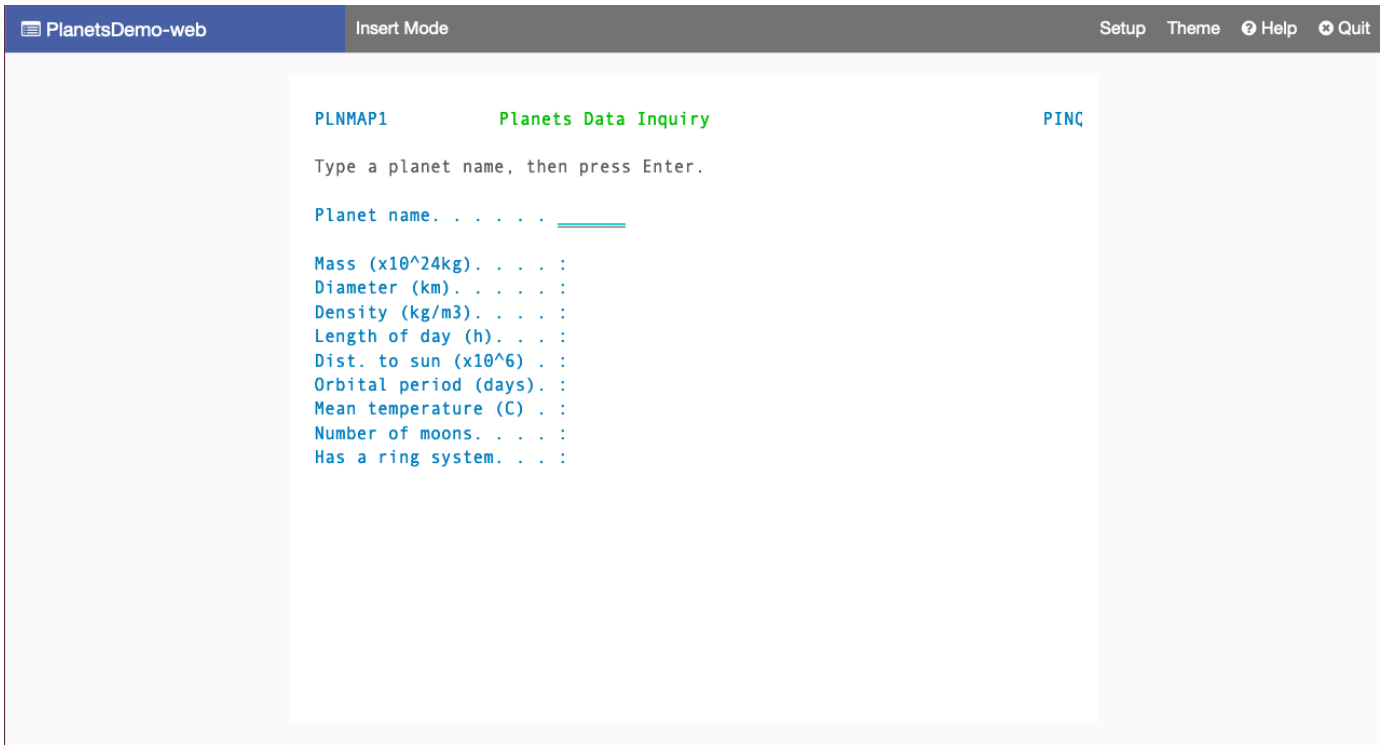
> **Note**
>
> Se a aplicação não estiver acessível e a regra de entrada no grupo de segurança tiver “Meu IP” selecionado na porta 8196, especifique a regra para permitir o tráfego de LB i/p na porta 8196.

## Etapa 8: Testar o aplicativo

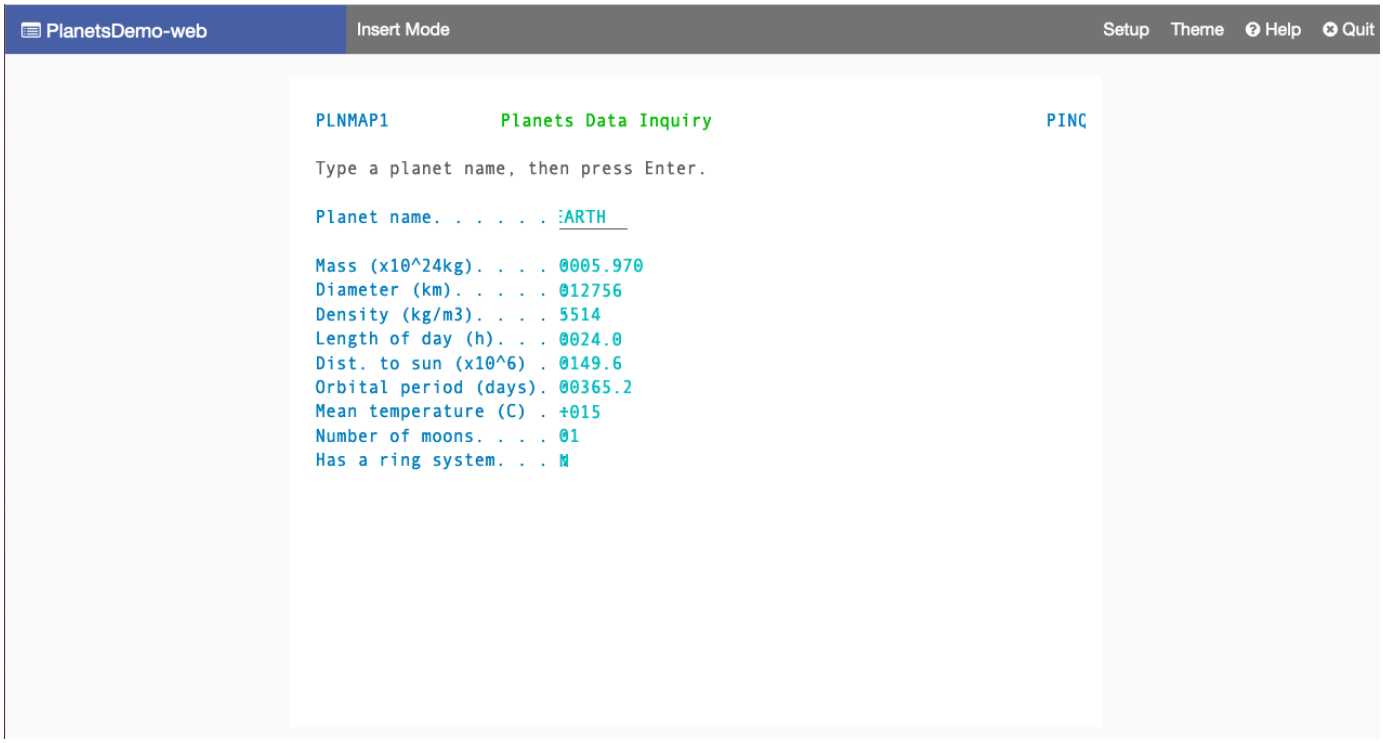
Nesta etapa, você executa uma transação no aplicativo migrado.

1. Na tela JICS, insira `PINQ` o campo de entrada e escolha Executar (ou pressione Enter) para iniciar a transação do aplicativo.

   A tela do aplicativo de demonstração deve aparecer.




2. Digite o nome do planeta no campo correspondente e pressione Enter.




Você deve ver detalhes sobre o planeta.

## Limpeza de recursos

Se os recursos criados para este tutorial não forem mais necessários, exclua-os para evitar cobranças adicionais. Para fazer isso, conclua as etapas a seguir:

- Se o aplicativo AWS Mainframe Modernization ainda estiver em execução, interrompa-o.
- Exclua o aplicativo do . Para ter mais informações, consulte Excluir um aplicativo do AWS Mainframe Modernization.
- Exclua o ambiente de execução. Para ter mais informações, consulte Excluir um ambiente de execução do AWS Mainframe Modernization.

# Tutorial: Tempo de execução gerenciado para Micro Focus

Este tutorial mostra como implantar e executar o aplicativo de CardDemo amostra em um ambiente de tempo de execução gerenciado de modernização de AWS mainframe com o mecanismo de tempo de execução da Micro Focus. O aplicativo CardDemo de amostra é um aplicativo simplificado

de cartão de crédito desenvolvido para testar e mostrar a tecnologia de uma parceria para casos AWS de uso de modernização de mainframe.

No tutorial, você cria recursos em outros Serviços da AWS Isso inclui o Amazon Simple Storage Service, o Amazon Relational Database ServiceAWS Key Management Service, AWS Secrets Manager e.

Tópicos



## Pré-requisitos

- Verifique se você tem acesso a um emulador 3270 para usar a conexão CICS. Os emuladores 3270 gratuitos e de teste estão disponíveis em sites de terceiros. Como alternativa, você pode iniciar uma instância Micro Focus de Modernização de AWS Mainframe AppStream 2.0 e usar o emulador Rumba 3270 (não disponível gratuitamente).

  Para obter informações sobre AppStream 2.0, consultethe section called “Tutorial: Configurar o AppStream 2.0 para Enterprise Analyzer e Enterprise Developer”.

> **Note**
>
> Ao criar a pilha, escolha a opção Enterprise Developer (ED) e não Enterprise Analyzer (EA).

- Baixe o aplicativo CardDemo de amostra e descompacte o arquivo baixado em qualquer diretório local. Esse diretório conterá um subdiretório intituladoCardDemo.
- Identifique uma VPC em sua conta na qual você possa definir os recursos criados neste tutorial. A VPC precisará de sub-redes em pelo menos duas zonas de disponibilidade. Para obter mais informações sobre a Amazon VPC, consulte Como a Amazon VPC funciona.

## Etapa 1: criar e carregar um bucket do Amazon S3

Nesta etapa, você cria um bucket do Amazon S3 e carrega CardDemo arquivos para esse bucket. Posteriormente neste tutorial, você usará esses arquivos para implantar e executar o aplicativo de CardDemo amostra em um ambiente Micro Focus Managed Runtime de modernização de AWS mainframe.

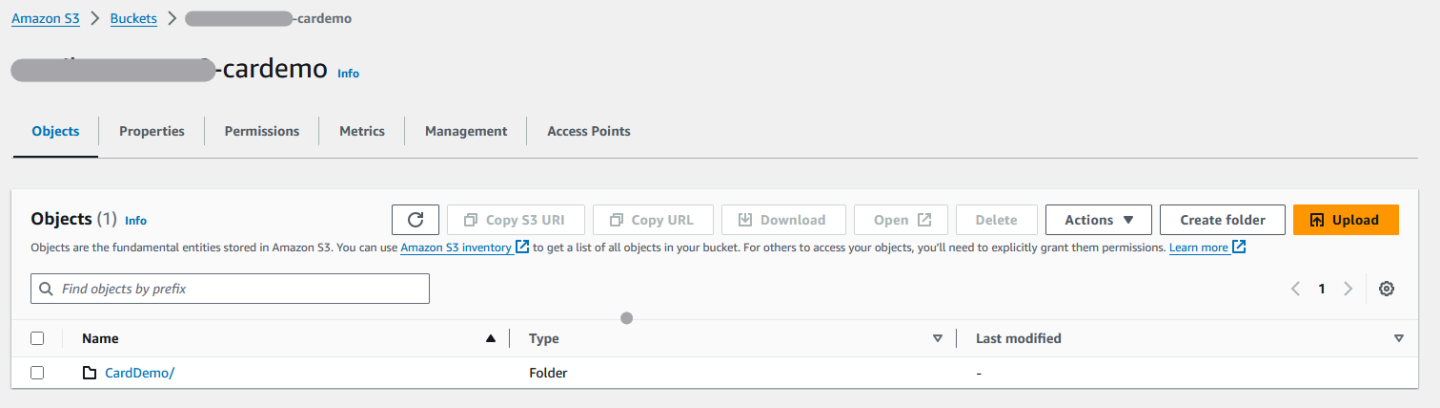
> **Note**
>
> Você não precisa criar um novo bucket do S3, mas o bucket escolhido deve estar na mesma região dos outros recursos usados neste tutorial.

Como criar um bucket do Amazon S3

1. Abra o console do Amazon S3 e escolha Create bucket.
2. Em Configuração geral, escolha a região da AWS em que você deseja criar o Micro Focus Managed Runtime de modernização de AWS mainframe.
3. Insira um nome de bucket, por exemplo,`yourname-aws-region-carddemo`. Mantenha as configurações padrão e escolha Criar bucket. Como alternativa, você também pode copiar as configurações de um bucket Amazon S3 existente e, em seguida, escolher Create bucket.
4. Escolha o bucket que você acabou de criar e, em seguida, escolha Upload.
5. Na seção Carregar, escolha Adicionar pasta e, em seguida, navegue até o `CardDemo` diretório a partir do seu computador local.

6. Escolha Upload para iniciar o processo de upload. Os tempos de upload variam de acordo com a velocidade da sua conexão.
7. Quando o upload for concluído, confirme se todos os arquivos foram carregados com sucesso e escolha Fechar.

Seu bucket do Amazon S3 agora contém a `CardDemo` pasta.




Para obter informações sobre buckets do S3, consulte Criação, configuração e trabalho com buckets do Amazon S3.

## Etapa 2: criar e configurar um banco de dados

Nesta etapa, você cria um banco de dados PostgreSQL no Amazon Relational Database Service (Amazon RDS). Para o tutorial, esse banco de dados contém os conjuntos de dados que o aplicativo de CardDemo amostra usa para tarefas do cliente relacionadas a transações com cartão de crédito.

Para criar um banco de dados no Amazon RDS

1. Abra o console do Amazon RDS.
2. Escolha a região da AWS na qual você deseja criar a instância do banco de dados.
3. Escolha Databases (Bancos de dados) no painel de navegação.
4. Escolha Criar banco de dados e, em seguida, escolha Criação padrão.
5. Em Tipo de mecanismo, escolha PostgreSQL.
6. Escolha uma versão do Engine de 15 ou superior.

> **Note**
>
> Salve a versão do mecanismo porque você precisará dela posteriormente neste tutorial.

7. Na seção Modelos, escolha Nível gratuito.
8. Altere o identificador da instância de banco de dados para algo significativo, por exemplo,`MicroFocus-Tutorial`.
9. Evite gerenciar credenciais mestras em. AWS Secrets Manager Em vez disso, insira uma senha mestra e confirme-a.

> **Note**
>
> Salve o nome de usuário e a senha que você usa para o banco de dados. Você os armazenará com segurança nas próximas etapas deste tutorial.

10. Em Conectividade, escolha a VPC em que você deseja criar o ambiente de tempo de execução gerenciado da modernização do AWS mainframe.
11. Selecione Criar banco de dados.

Para criar um grupo de parâmetros personalizado no Amazon RDS

1. No painel de navegação do console Amazon RDS, escolha Parameter groups e, em seguida, escolha Create parameter group.
2. Na janela Criar grupo de parâmetros, em Família de grupos de parâmetros, selecione a opção Postgres que corresponda à versão do seu banco de dados.

> **Note**
>
> Algumas versões do Postgres exigem um Tipo. Selecione DB Parameter Group, se necessário. Insira um nome de grupo e uma descrição para o grupo de parâmetros.

3. Selecione Create.

Para configurar o grupo de parâmetros personalizado

1. Escolha o grupo de parâmetros recém-criado.

2. Escolha Ações e, em seguida, escolha Editar.
3. Filtre `max_prepared_transactions` e altere o valor do parâmetro para 100.
4. Escolha Salvar alterações.

Para associar o grupo de parâmetros personalizados ao banco de dados

1. No painel de navegação do console Amazon RDS, escolha Bancos de dados e, em seguida, escolha a instância do banco de dados que você deseja modificar.
2. Selecione Modify. A página Modify DB instance (Modificar instância de banco de dados) será exibida.

> **Note**
>
> A opção Modificar não está disponível até que o banco de dados termine de criar e fazer backup, o que pode levar alguns minutos.

3. Na página Modificar instância de banco de dados, navegue até Configuração adicional e altere o grupo de parâmetros do banco de dados para seu grupo de parâmetros. Se seu grupo de parâmetros não estiver disponível na lista, verifique se ele foi criado com a versão correta do banco de dados.
4. Escolha Continuar e verifique o resumo das modificações.
5. Escolha Aplicar imediatamente para aplicar as alterações instantaneamente.
6. Selecione Modificar instância de banco de dados para salvar as alterações.

Para obter mais informações sobre grupos de parâmetros, consulte Trabalho com grupos de parâmetros.

> **Note**
>
> Você também pode usar um banco de dados Amazon Aurora PostgreSQL com a modernização do AWS mainframe, mas não há opção de nível gratuito. Para obter mais informações, consulte Trabalhando com o Amazon Aurora PostgreSQL.

# Etapa 3: criar e configurar um AWS KMS key

Para armazenar credenciais com segurança para a instância do Amazon RDS, primeiro crie um. AWS KMS key

Para criar uma AWS KMS key

1. Abra o console do Key Management Service.
2. Escolha Create Key (Criar chave).
3. Deixe os padrões de Symmetric para o tipo de chave e Criptografar e descriptografar para o uso da chave.
4. Escolha Próximo.
5. Dê à chave um alias, como por exemplo, `MicroFocus-Tutorial-RDS-Key` e uma descrição opcional.
6. Escolha Próximo.
7. Atribua um administrador de chaves marcando a caixa ao lado do seu usuário ou função.
8. Escolha Avançar e, em seguida, escolha Avançar novamente.
9. Na tela de revisão, edite a política de chaves e insira o seguinte:

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

   Essa política concede permissões de decodificação da Modernização do AWS Mainframe usando essa política de chaves específica.
10. Escolha Concluir para criar a chave.

Para obter mais informações, consulte Criação de chaves no Guia do AWS Key Management Service desenvolvedor.

# Etapa 4: criar e configurar um segredo de AWS Secrets Manager banco de dados

Agora, armazene as credenciais do banco de dados com segurança usando e. AWS Secrets Manager AWS KMS key

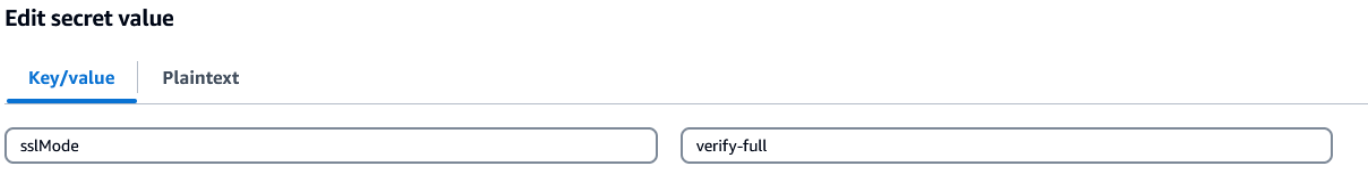
Para criar e configurar um segredo de AWS Secrets Manager banco de dados

1. Abra o console do Secrets Manager.
2. No painel de navegação, escolha Secrets (Segredos).
3. Em Segredos, escolha Armazenar um novo segredo.
4. Defina o tipo de segredo como Credenciais para o banco de dados Amazon RDS.
5. Insira as credenciais que você especificou ao criar o banco de dados.
6. Em Chave de criptografia, selecione a chave que você criou na etapa 3.
7. Na seção Banco de dados, selecione o banco de dados que você criou para este tutorial e escolha Avançar.
8. Em Nome secreto, insira um nome como `MicroFocus-Tutorial-RDS-Secret` e uma descrição opcional.
9. Na seção Permissões de recursos, escolha Editar permissões e substitua o conteúdo pela seguinte política:

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

10. Escolha Salvar.
11. Escolha Avançar para as telas subsequentes e, em seguida, escolha Armazenar. Atualize a lista de segredos para ver o novo segredo.

12. Escolha o segredo recém-criado e anote-o `Secret ARN` porque você precisará dele posteriormente no tutorial.
13. Na guia Visão geral do segredo, escolha Recuperar valor secreto.
14. Escolha Editar e, em seguida, escolha Adicionar linha.
15. Adicione uma chave para `sslMode` com um valor de`verify-full`:

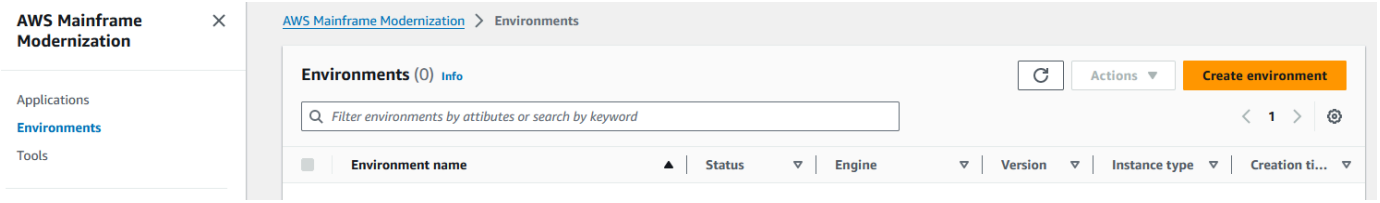



16. Escolha Salvar.

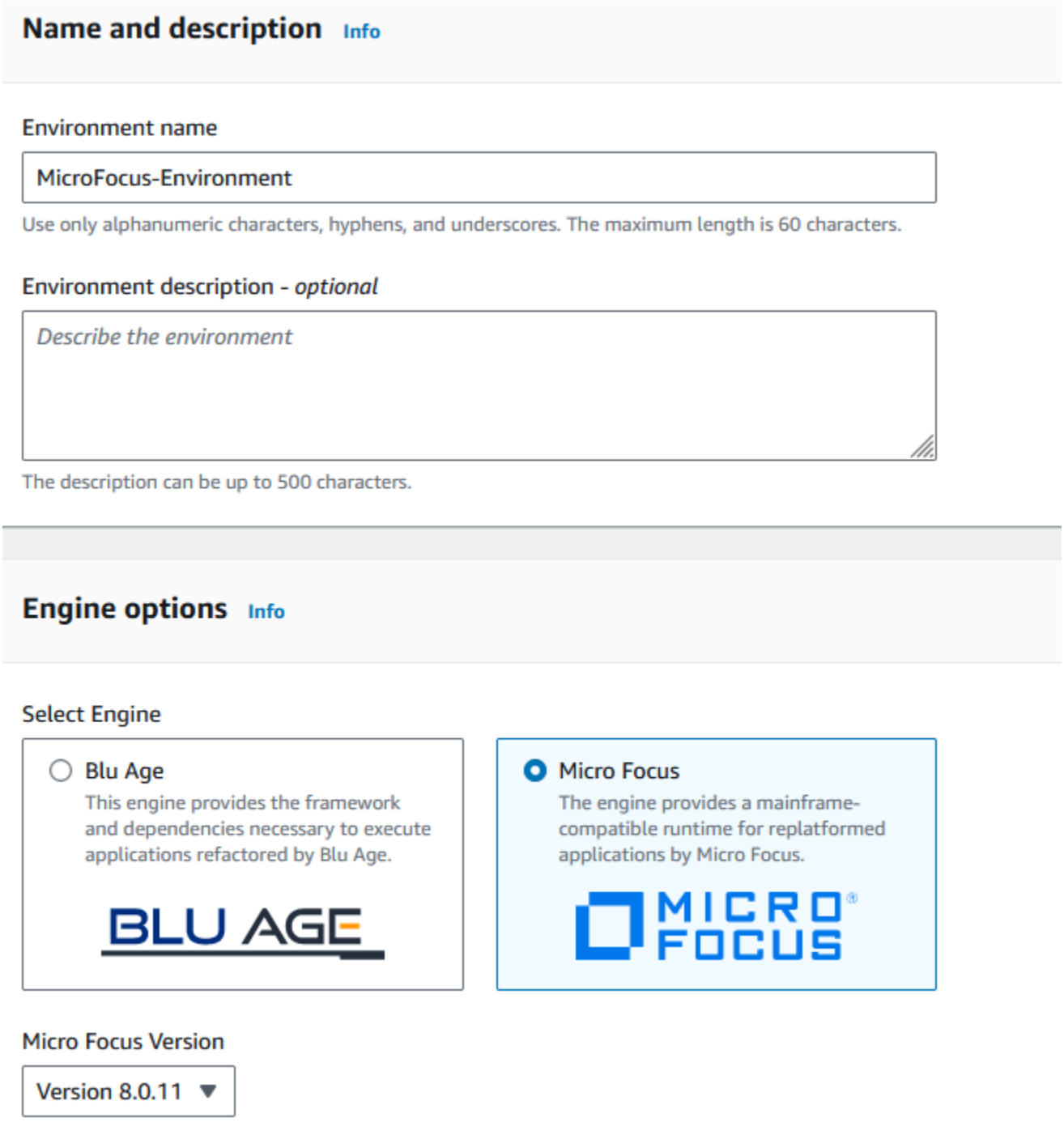
## Etapa 5: criar um ambiente de tempo de execução

Para criar um ambiente de tempo de execução

1. Abra o console do AWS Mainframe Modernization.
2. No painel de navegação, escolha Ambientes. Em seguida, escolha Criar ambiente.




3. Em Especificar informações básicas,
    a. Insira `MicroFocus-Environment` o nome do ambiente.
    b. Nas opções do motor, verifique se a opção Micro Focus está selecionada.
    c. Escolha a versão mais recente do Micro Focus.
    d. Escolha Próximo.




4. Configure o ambiente

   a. Em Disponibilidade, escolha Cluster de alta disponibilidade.

   b. Em Recursos, escolha um `M2.c5.large` ou `M2.m5.large` para o tipo de instância e o número de instâncias que você deseja. Especifique até duas instâncias.

c. Em Segurança e rede, escolha Permitir que aplicativos implantados nesse ambiente sejam acessíveis publicamente e escolha pelo menos duas sub-redes públicas.

d. Escolha Próximo.

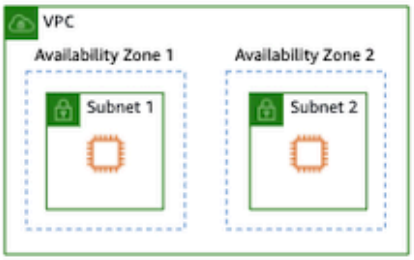
# Specify configurations Info

## 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.

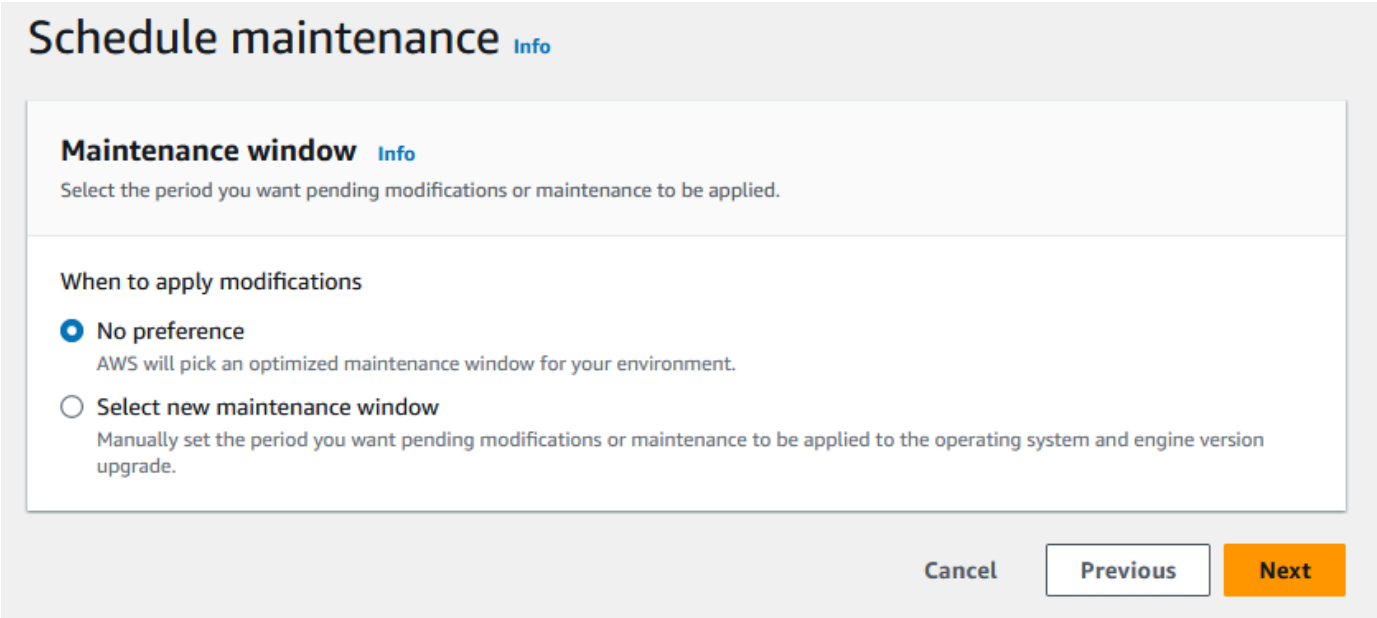
Choose subnets

subnet-56f1a | us-west-2a

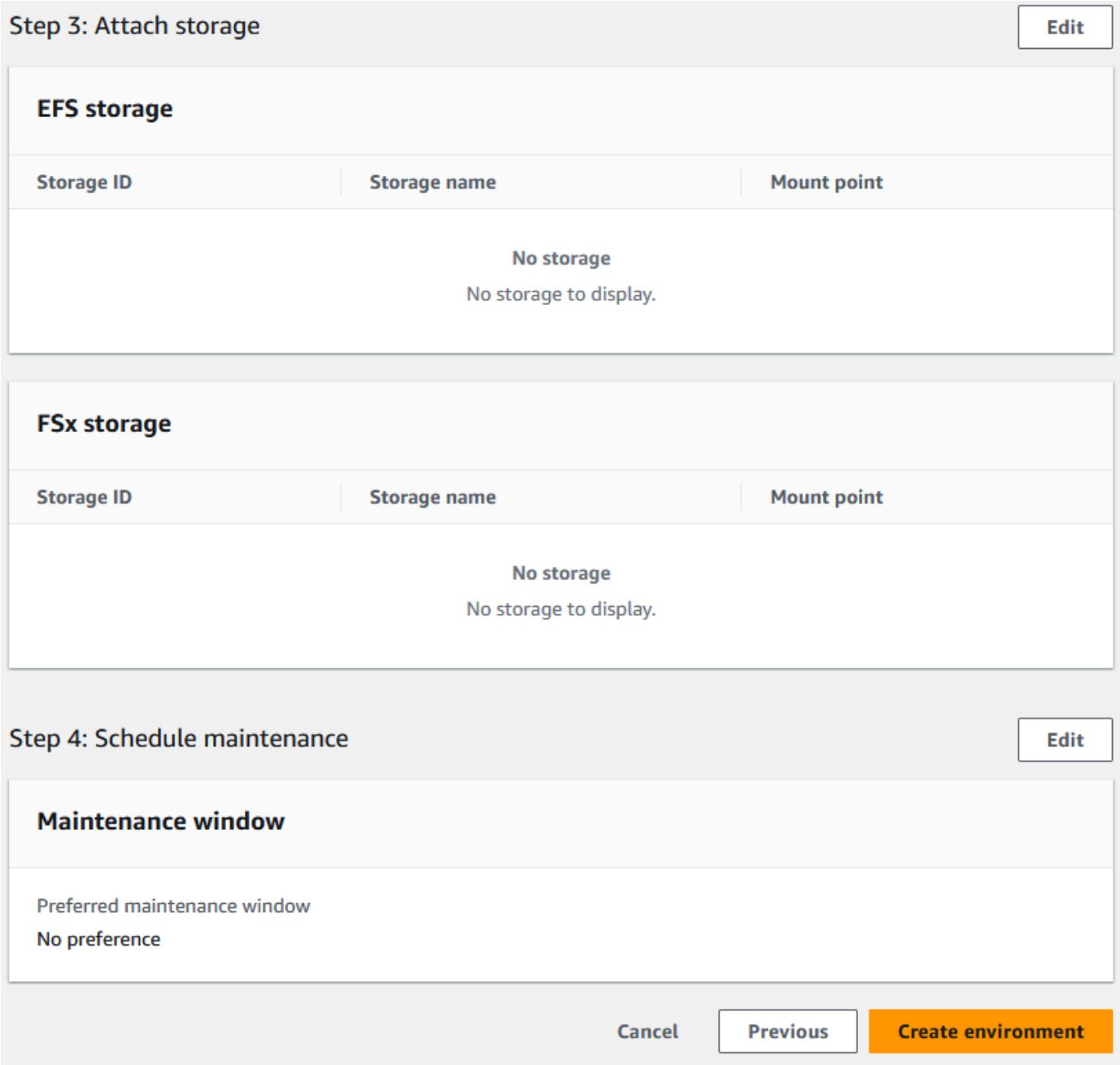
subnet-6685 | us-west-2b

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

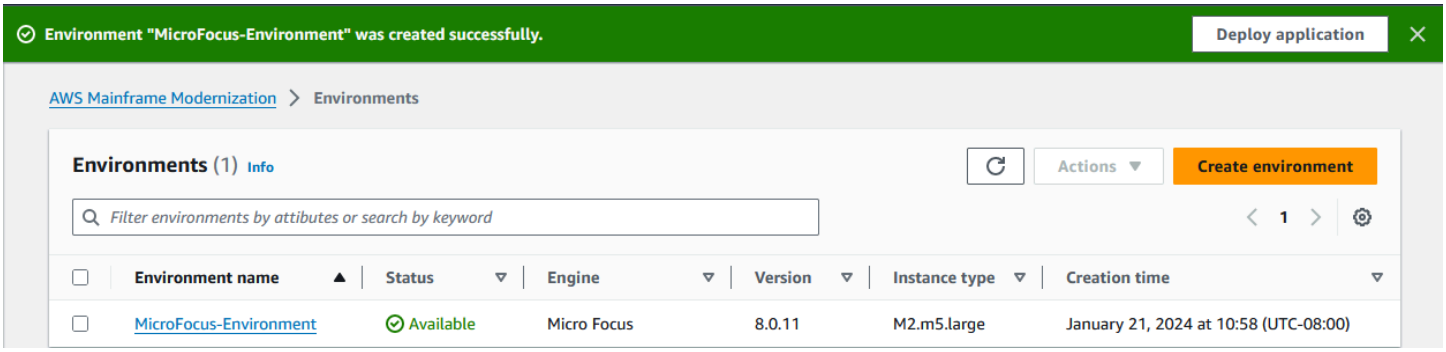
5. Na página Anexar armazenamento, escolha Próximo.
6. Na página Programar manutenção, escolha Sem preferência e, em seguida, escolha Avançar.




7. Na página Revisar e criar, revise todas as configurações que você forneceu para o ambiente de tempo de execução e escolha Criar ambiente.



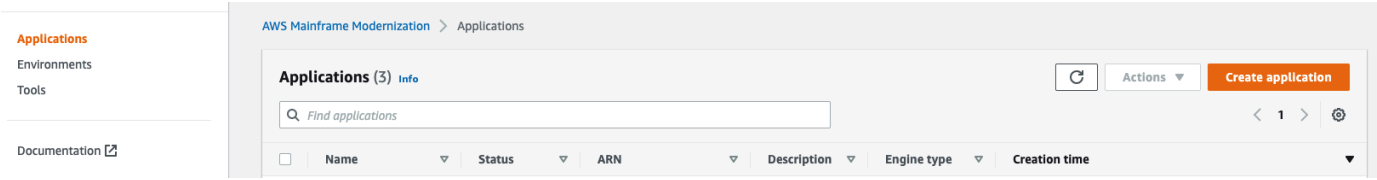

Quando você cria seu ambiente, aparece um banner que diz `Environment` *`name`* `was created successfully`, e o campo Status muda para Disponível. O processo de criação do ambiente leva alguns minutos, mas você pode continuar com as próximas etapas enquanto ele é executado.



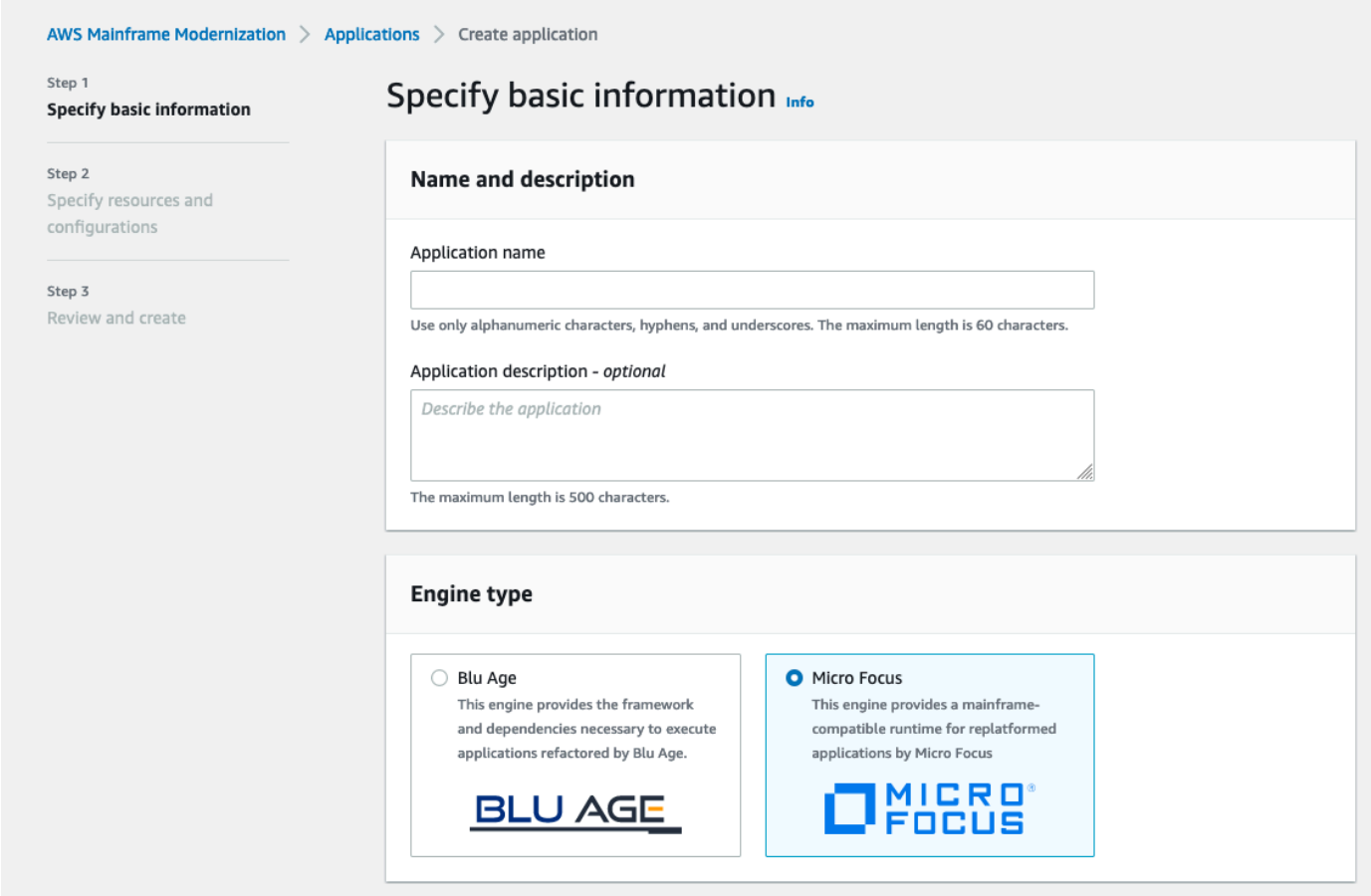

# Etapa 6: criar um aplicativo

Para criar um aplicativo.

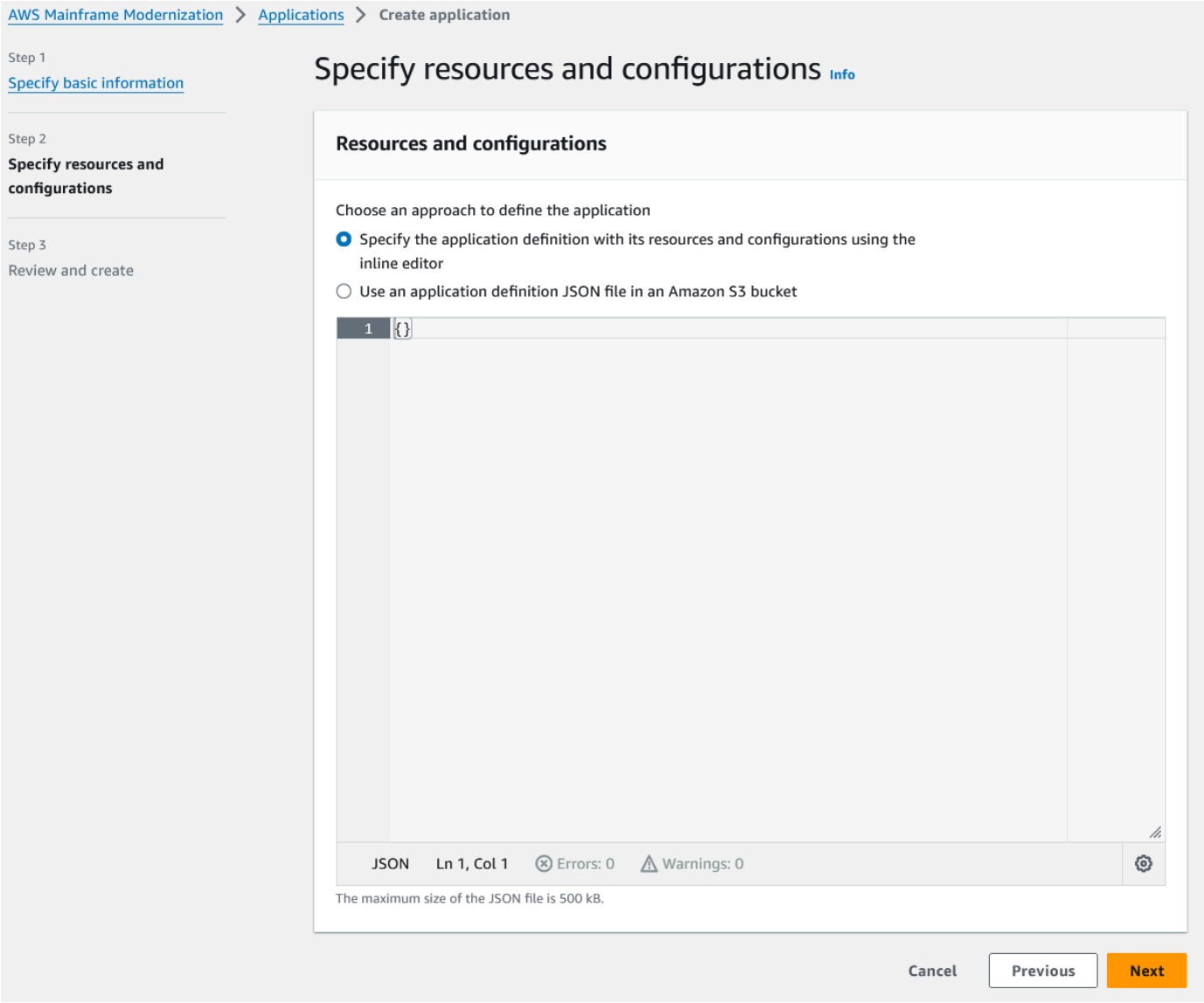
1. No painel de navegação, escolha Aplicativos. Escolha Criar aplicativo.




2. Na página Criar aplicativo, em Especificar informações básicas, insira `MicroFocus-CardDemo` o nome do aplicativo e, em Tipo de mecanismo, verifique se a opção Micro Focus está selecionada. Em seguida, escolha Próximo.




3. Em Especificar recursos e configurações, escolha a opção para especificar a definição do aplicativo com seus recursos e configurações usando o editor embutido.




Insira a seguinte definição de aplicativo no 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"
      }
    ]
  }
}
```

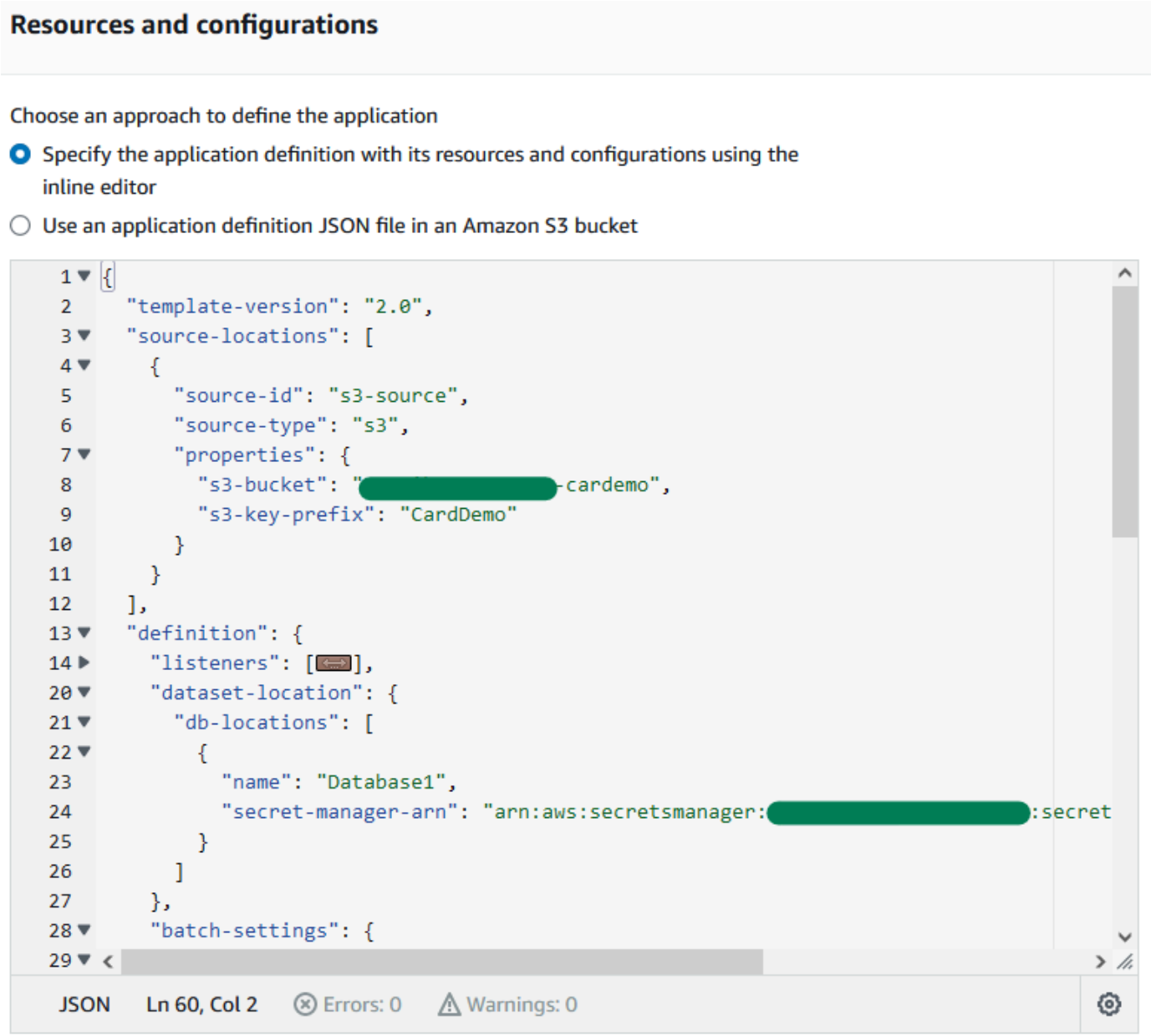
> **Note**
>
> Esse arquivo está sujeito a alterações.

4. Edite o aplicativo JSON no objeto de propriedades dos locais de origem da seguinte forma:

   a. Substitua o valor `s3_bucket` de pelo nome do bucket do Amazon S3 que você criou na Etapa 1.

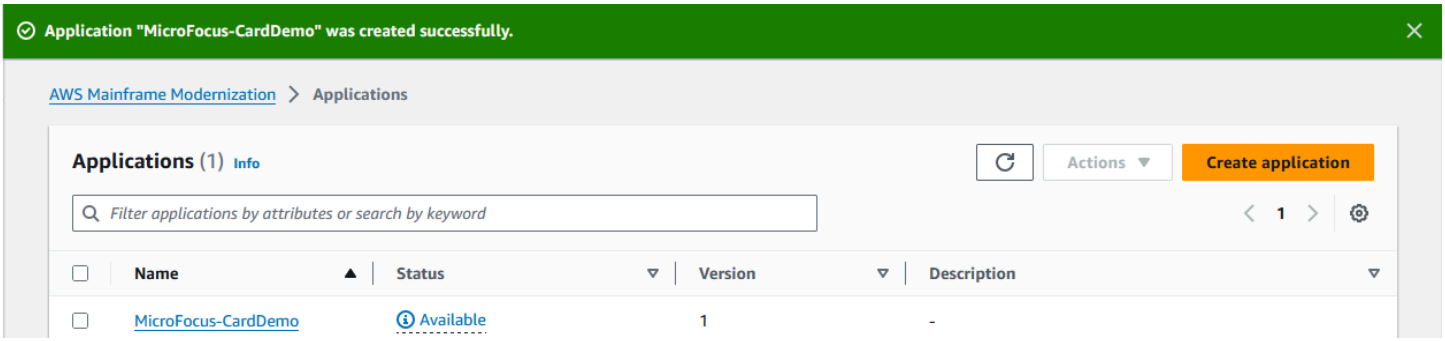
   b. Substitua o valor `s3-key-prefix` de pela pasta (prefixo de chave) na qual você fez o upload dos arquivos de CardDemo amostra. Se você fez o upload do `CardDemo` diretório diretamente para um bucket do Amazon S3, `s3-key-prefix` ele não precisa ser alterado.

   c. Substitua `secret-manager-arn` os dois valores pelo ARN do segredo do banco de dados que você criou na Etapa 4.




Para obter mais informações sobre a definição do aplicativo, consulte [Definição de aplicativo do Micro Focus].

5. Escolha Próximo para continuar.
6. Na página Revisar e criar, revise as informações fornecidas e escolha Criar aplicativo.

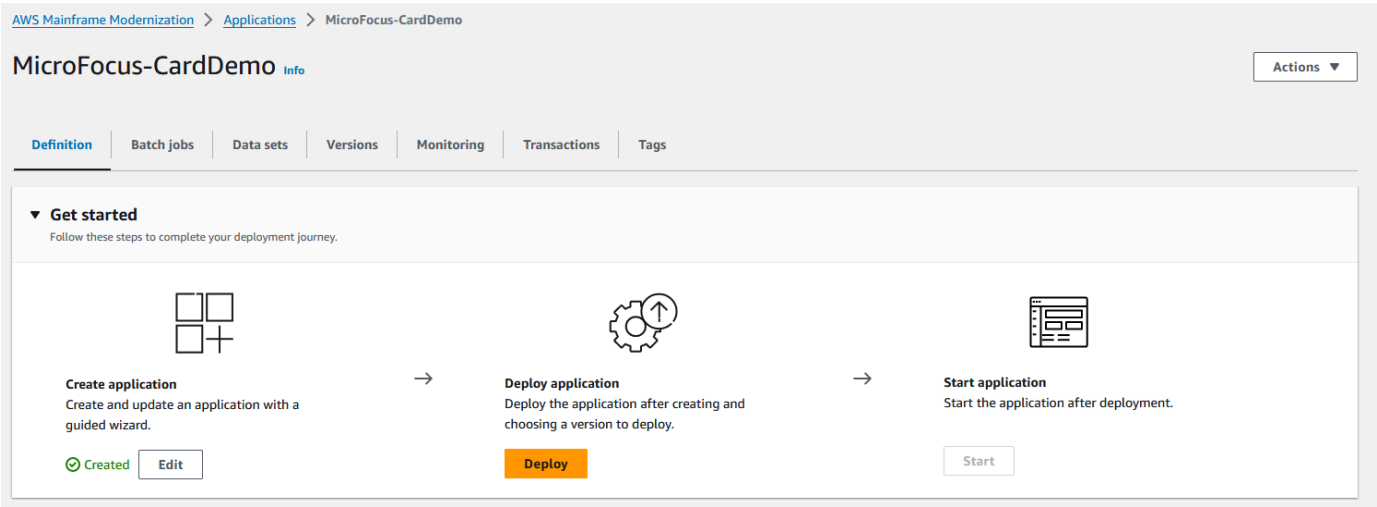



Quando você cria seu aplicativo, aparece um banner que diz`Application` *`name`* `was created successfully`. E o campo Status muda para Disponível.

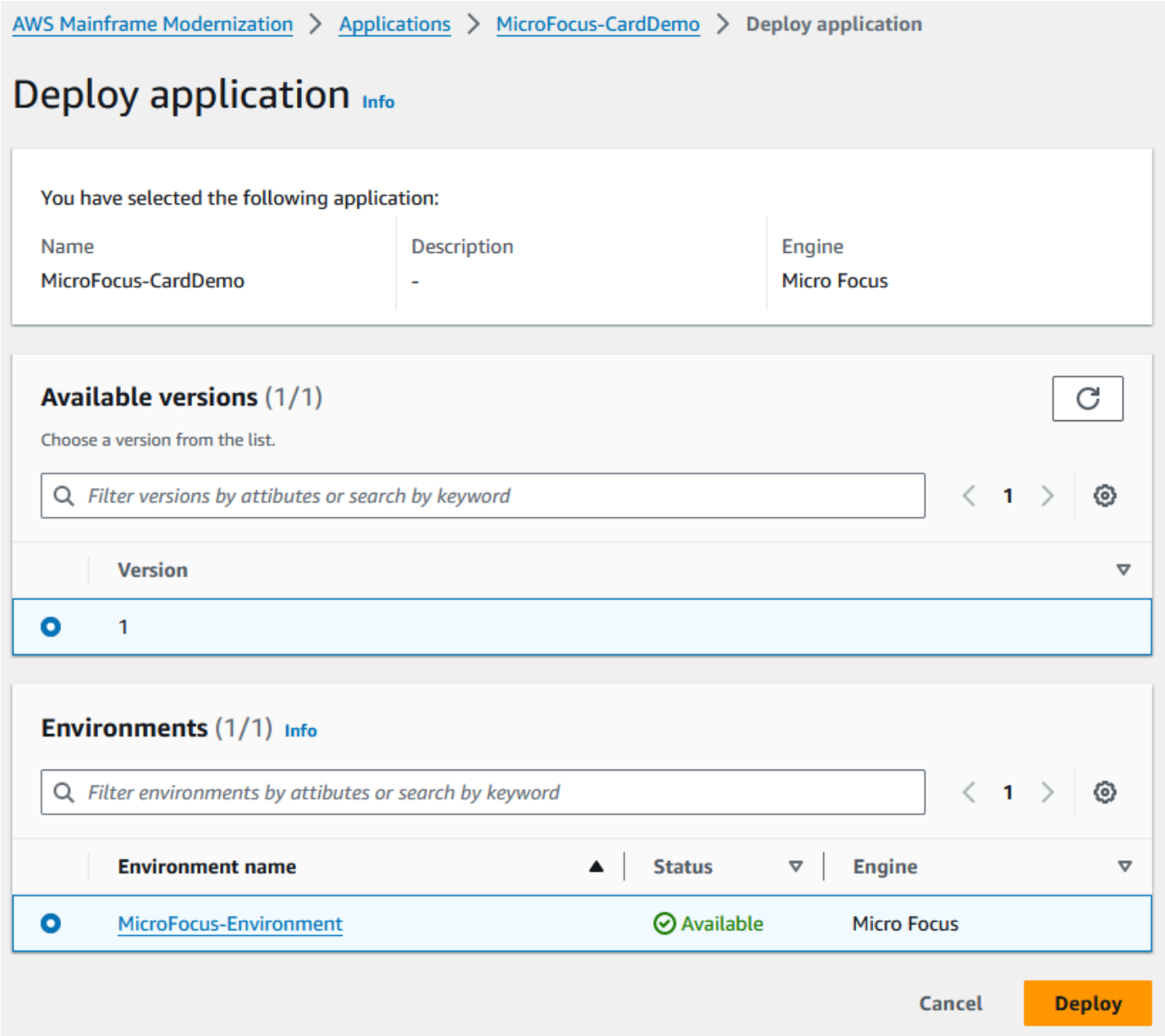
## Etapa 7: implantar um aplicativo

Para implantar uma aplicação

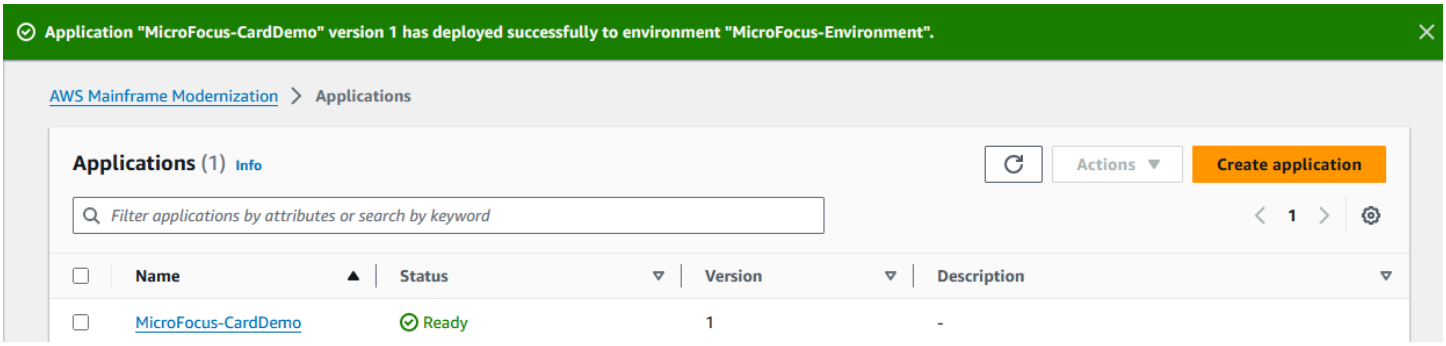
1. No painel de navegação, escolha Aplicativos e, em seguida, escolha`MicroFocus-CardDemo`.
2. Em Implantar aplicativo, escolha Implantar.




3. Escolha a versão mais recente do aplicativo e do ambiente que você criou anteriormente e, em seguida, escolha Implantar.



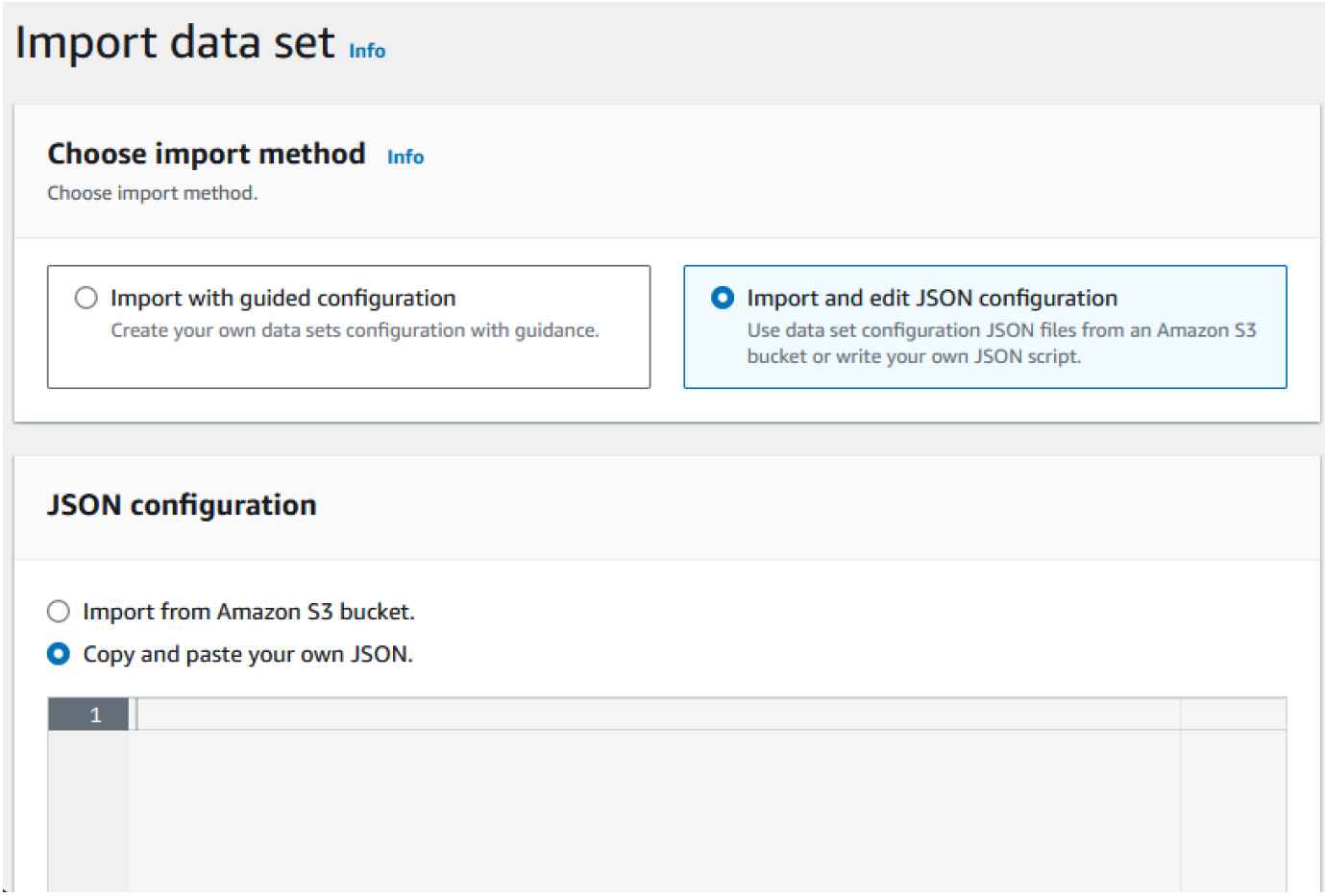

Quando o CardDemo aplicativo é implantado com êxito, o status muda para Pronto.



# Etapa 8: importar conjuntos de dados

Para importar conjuntos de dados

1. No painel de navegação, escolha Aplicativos e, em seguida, escolha o aplicativo.
2. Escolha a guia Conjuntos de dados. Selecione Import (Importar).
3. Escolha Importar e editar a configuração JSON e, em seguida, escolha a opção Copiar e colar seu próprio JSON.




4. Copie e cole o seguinte JSON, mas ainda não escolha “Enviar”. Esse JSON contém todos os conjuntos de dados necessários para o aplicativo de demonstração, mas precisa dos detalhes do bucket do Amazon S3.

```
{
    "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",
```

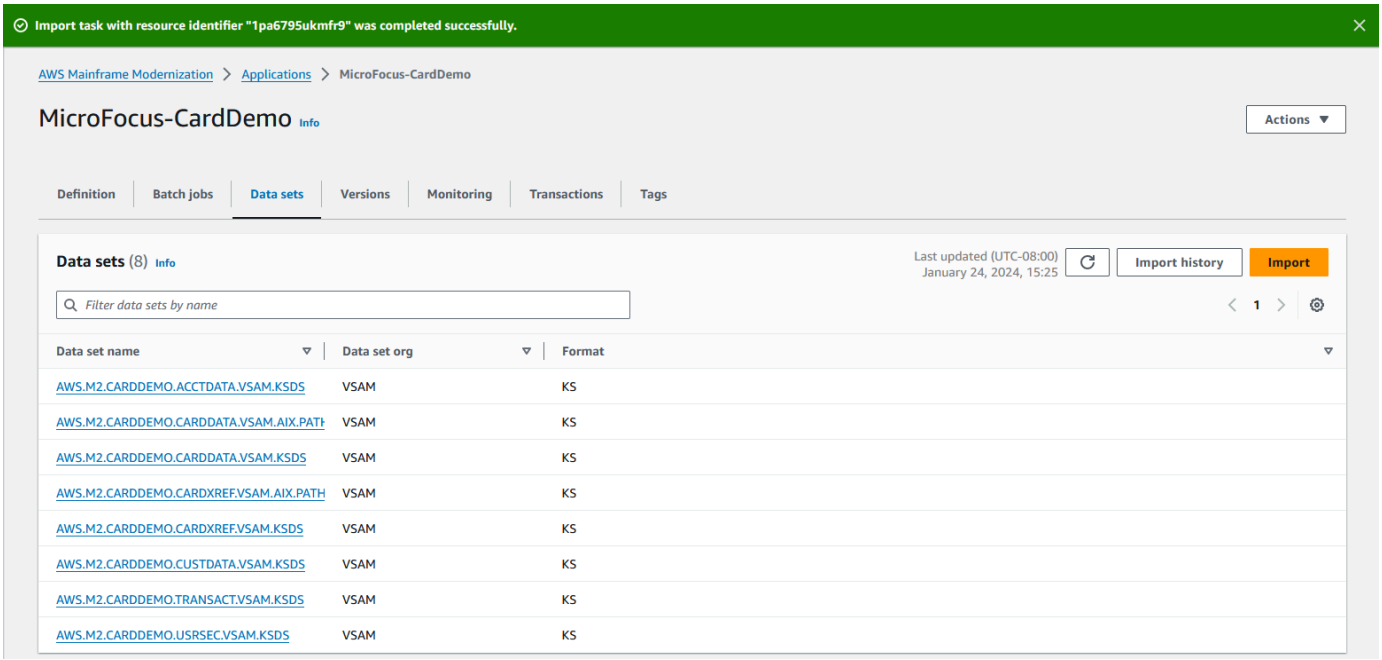
```
                "datasetName": "AWS.M2.CARDDEMO.USRSEC.VSAM.KSDS",
                "relativePath": "DATA",
                "datasetOrg": {
                    "vsam": {
                        "format": "KS",
                        "encoding": "A",
                        "primaryKey": {
                            "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. Substitua cada ocorrência de `<s3-bucket-name>` (há oito) pelo nome do bucket do Amazon S3 que contém a CardDemo pasta, por exemplo,. `your-name-aws-region-carddemo`

> **Note**
>
> Para copiar o URI do Amazon S3 para a pasta no Amazon S3, selecione a pasta e escolha Copiar URI do Amazon S3.

6. Selecione Enviar.

   Quando a importação é concluída, um banner aparece com a seguinte mensagem: `Import task with resource identifier` *`name`* `was completed successfully.` Uma lista dos conjuntos de dados importados é exibida.

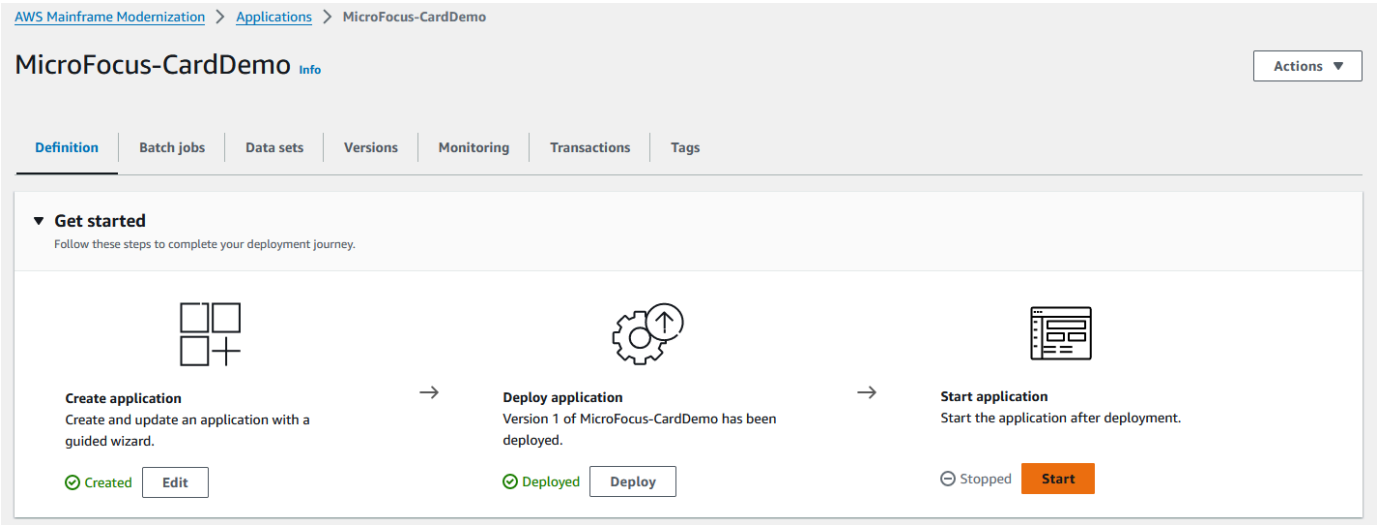



Você também pode visualizar o status de todas as importações de conjuntos de dados escolhendo Histórico de importação na guia Conjuntos de dados.

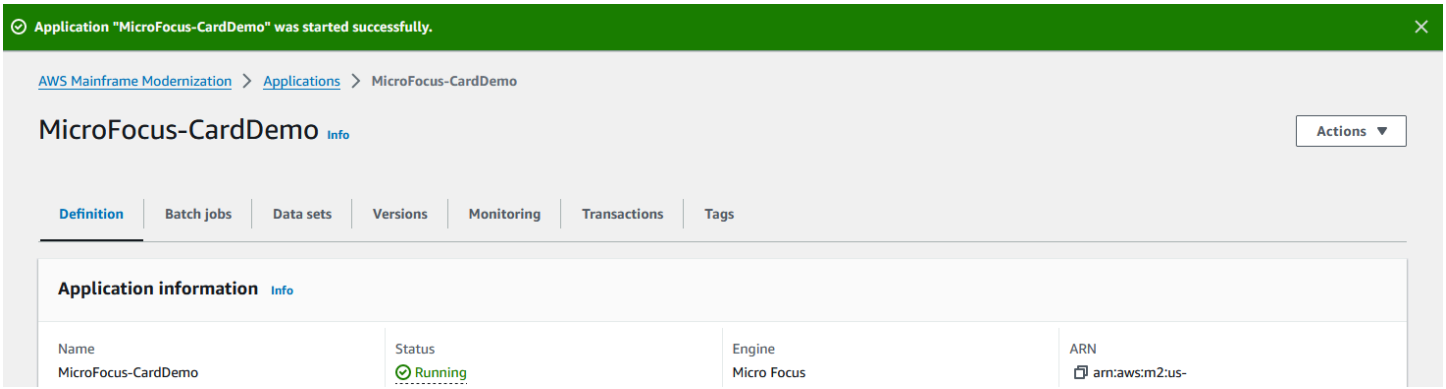
## Etapa 9: iniciar um aplicativo

Para iniciar um aplicativo

1. No painel de navegação, escolha Aplicativos e, em seguida, escolha o aplicativo.
2. Escolha Iniciar aplicativo.



Quando o CardDemo aplicativo começa a ser executado com sucesso, um banner aparece com a seguinte mensagem:`Application` *`name`* `was started successfully`. O campo Status muda para Em execução.




## Etapa 10: Conecte-se ao aplicativo CardDemo CICS

Antes de se conectar, certifique-se de que a VPC e o grupo de segurança que você especificou para o aplicativo sejam os mesmos que você aplicou para a interface de rede a partir da qual você se conectará.

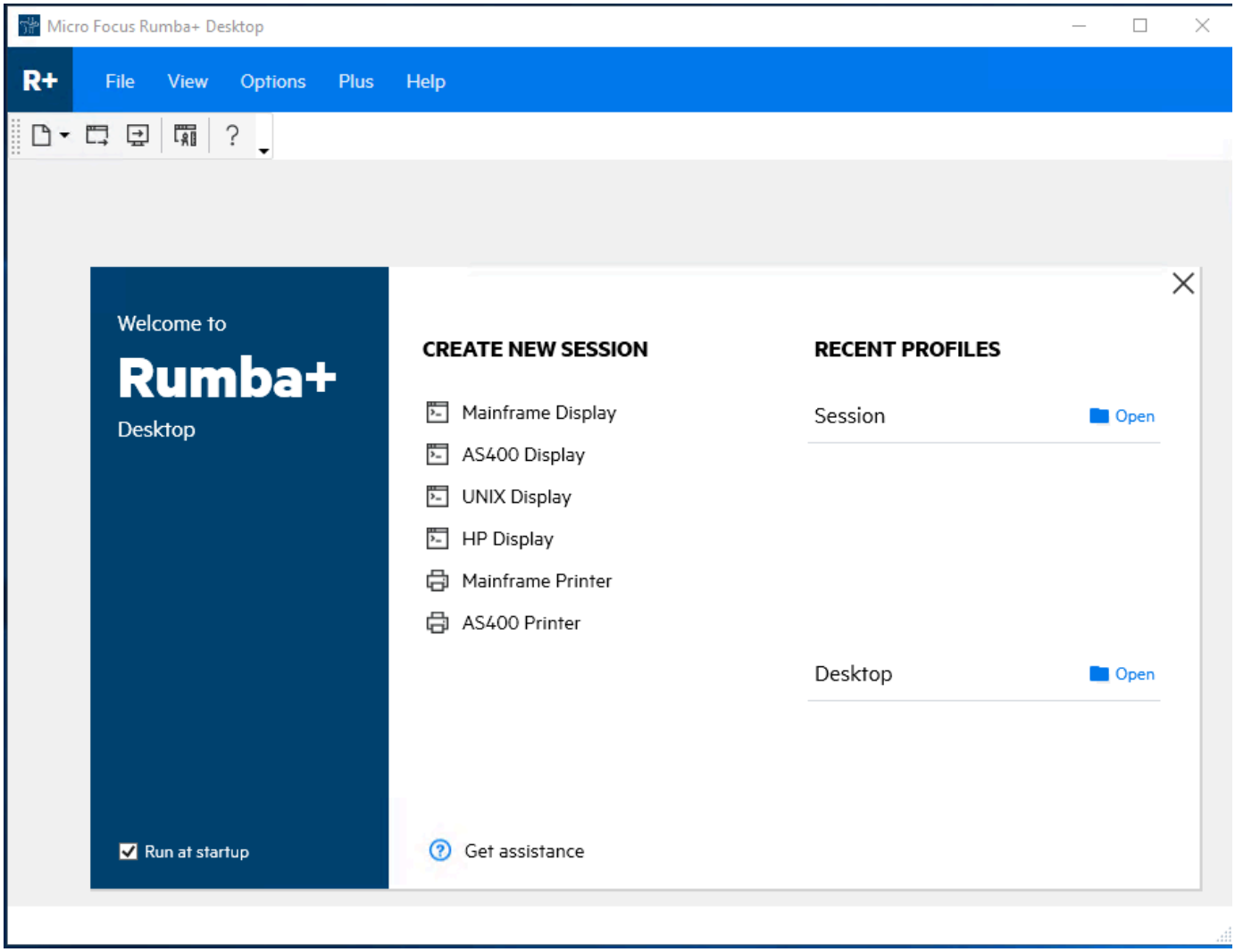
Para configurar a conexão TN3270, você também precisa do nome do host DNS e da porta do aplicativo.

Para configurar e conectar um aplicativo ao mainframe usando o emulador de terminal

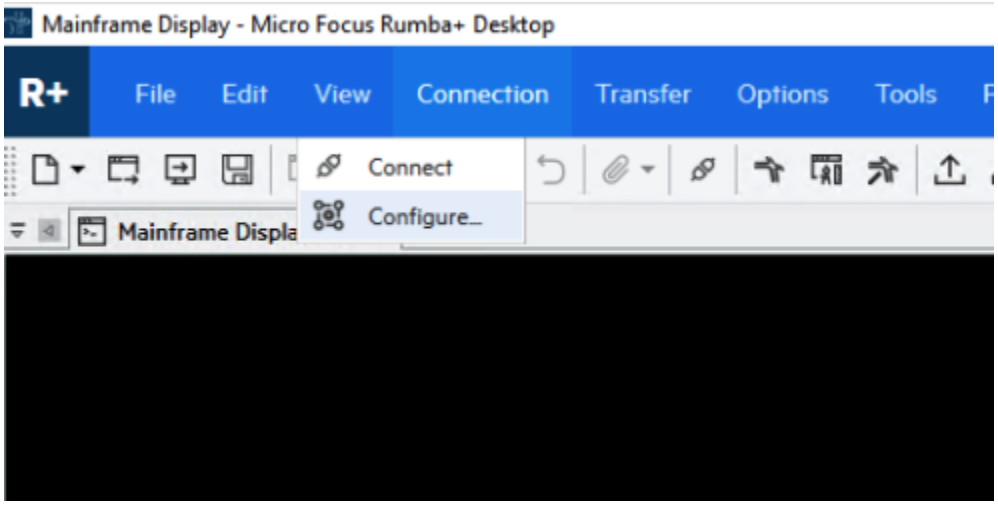
1. Abra o console de modernização do AWS mainframe, escolha Aplicativos e, em seguida, escolha. `MicroFocus-CardDemo`
2. Escolha o ícone de cópia para copiar o nome do host DNS. Além disso, certifique-se de anotar o número das portas.
3. Inicie um emulador de terminal. Este tutorial usa o Micro Focus Rumba+.

   > **Note**
   >
   > As etapas de configuração variam de acordo com o emulador.

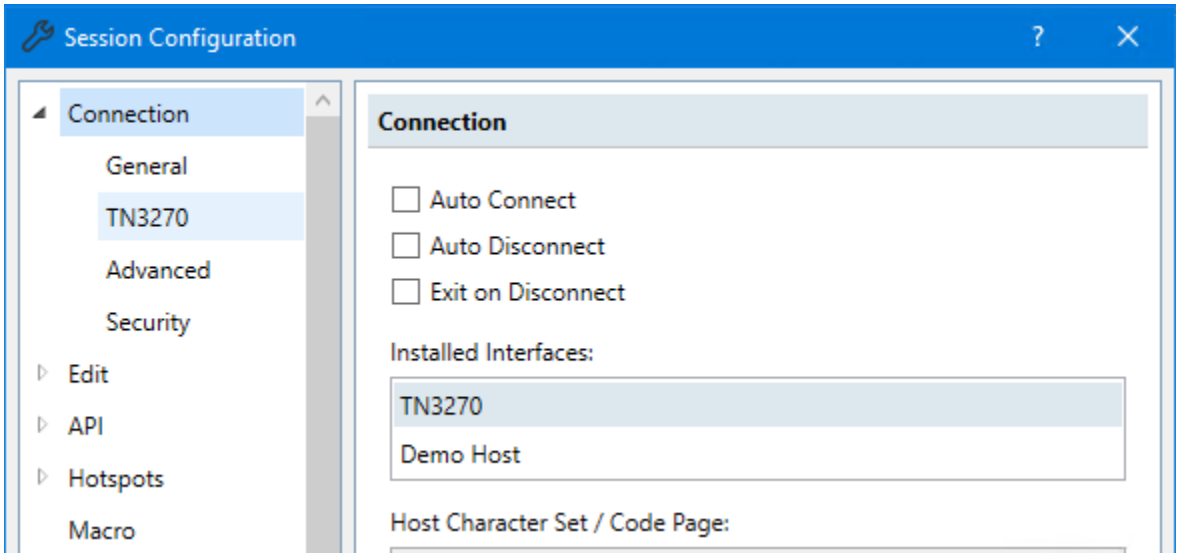
4. Escolha Mainframe Display.



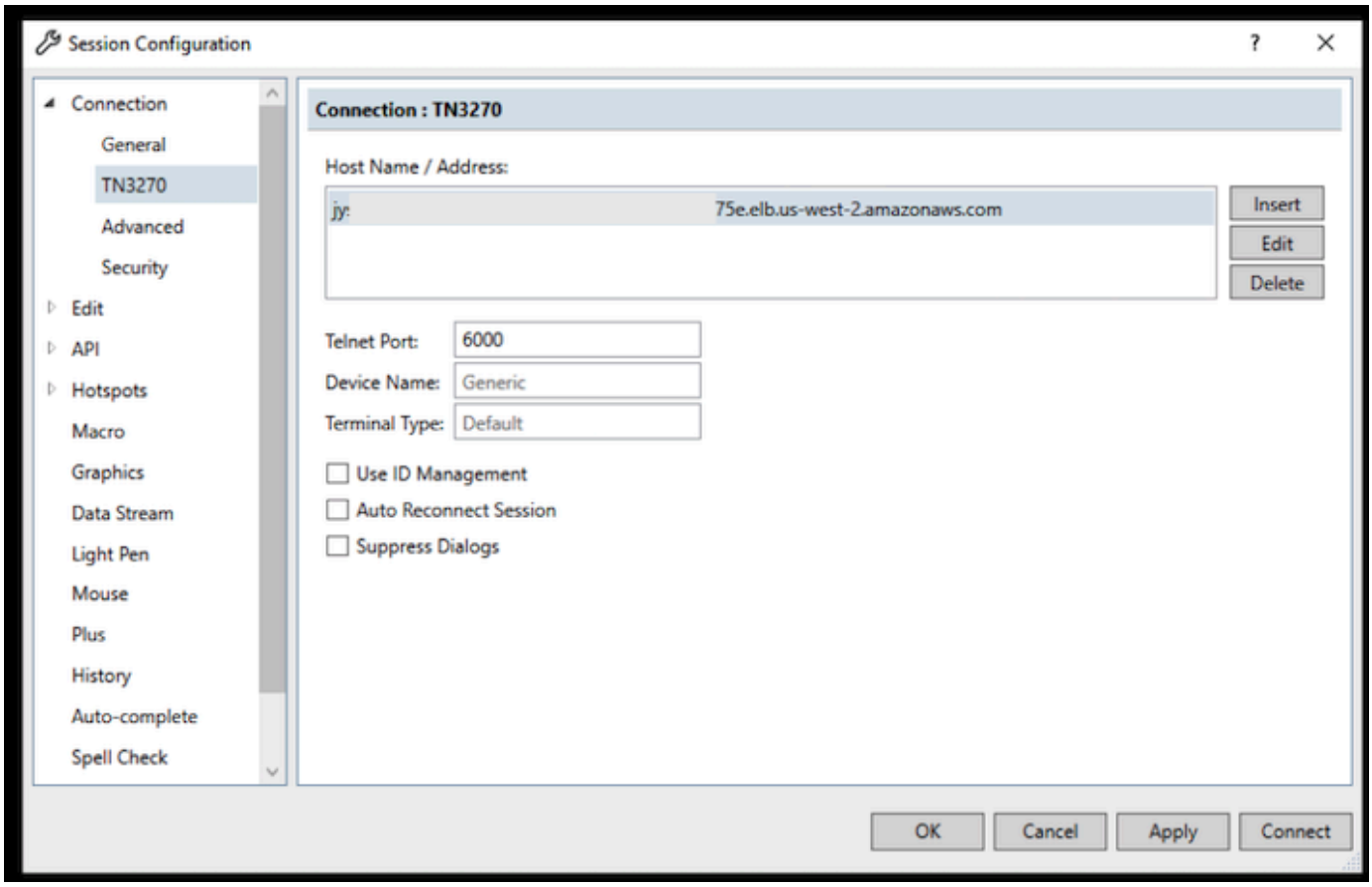

5. Escolha Conexão e, em seguida, escolha Configurar.




6. Em Interfaces instaladas, escolha eTN3270, em seguida, escolha TN3270 novamente no menu Conexão.



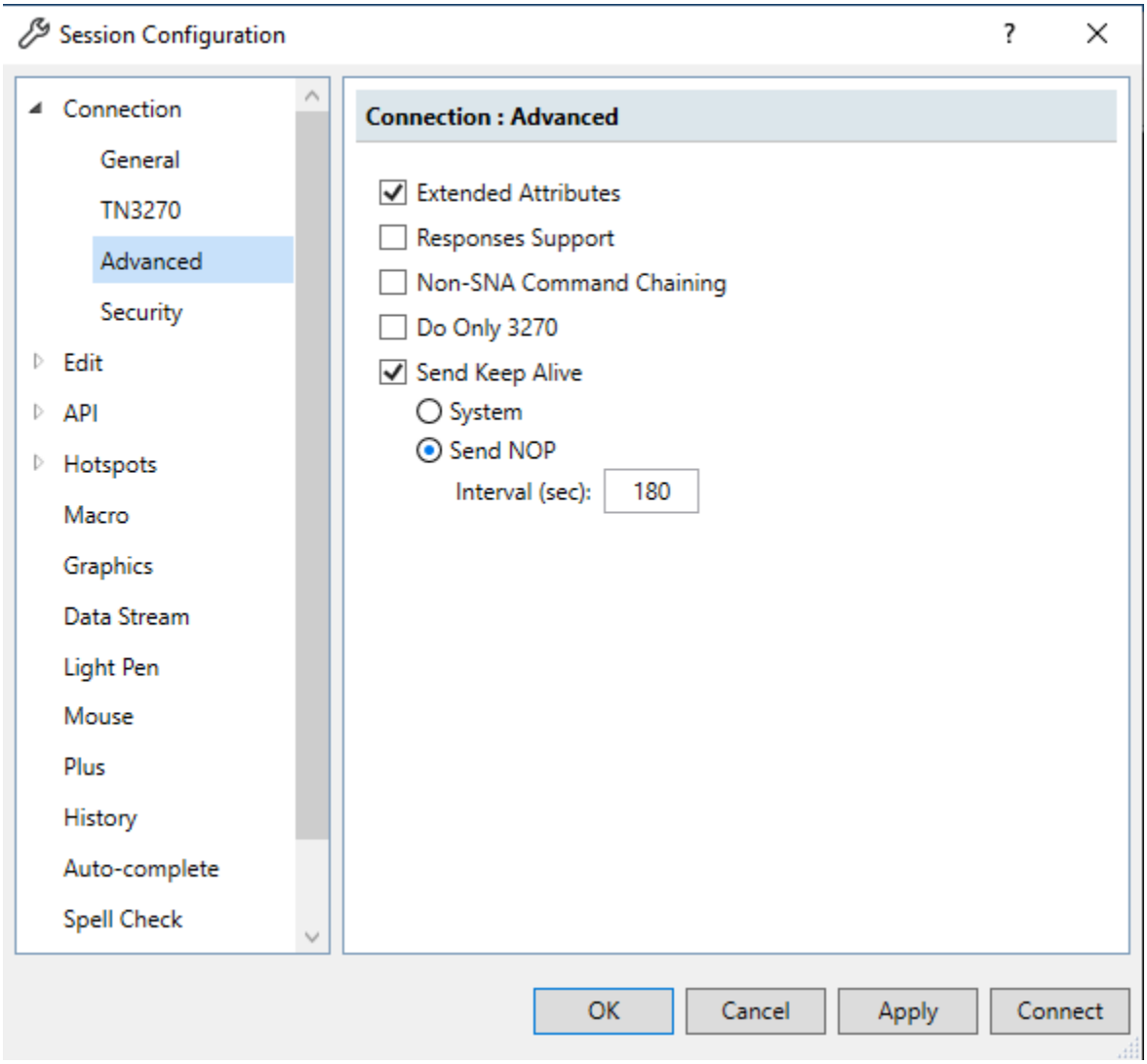

7. Escolha Inserir e cole o DNS `Hostname` para o aplicativo. Especifique `6000` para a porta Telnet.



> **Note**
>
> Se você estiver usando o AWS AppStream 2.0 em um navegador e tiver dificuldades em colar valores, consulte Solução de problemas do usuário AppStream 2.0.

8. Em Conexão, escolha Avançado e, em seguida, escolha Send Keep Alive e Send NOP e insira 180 para o Intervalo.

> **Note**
>
> Configurar a configuração keep alive em seu terminal TN3270 para pelo menos 180 segundos ajuda a garantir que o Network Load Balancer não interrompa sua conexão.



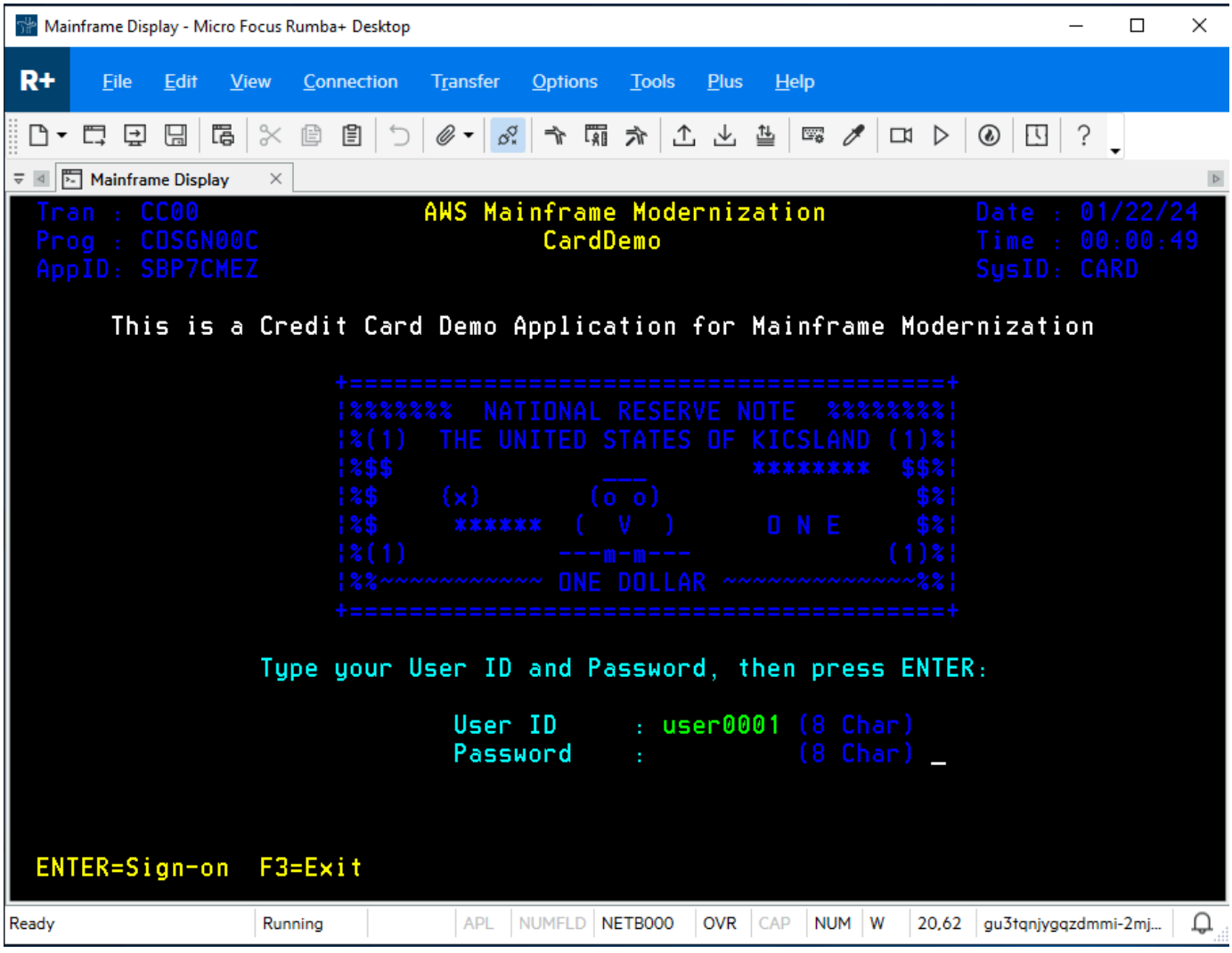

9. Selecione Conectar.

> **Note**
>
> Se a conexão falhar:
>
> - Se você estiver usando AppStream 2.0, confirme se a VPC e o grupo de segurança especificados para o ambiente do aplicativo são os mesmos da frota 2.0. AppStream
> - Use o VPC Reachability Analyzer para analisar a conexão. Você pode acessar o Reachability Analyzer por meio do console.
> - Como etapa de diagnóstico, tente adicionar ou alterar as regras de entrada do Grupo de Segurança do aplicativo para permitir o tráfego para a porta 6000 de qualquer

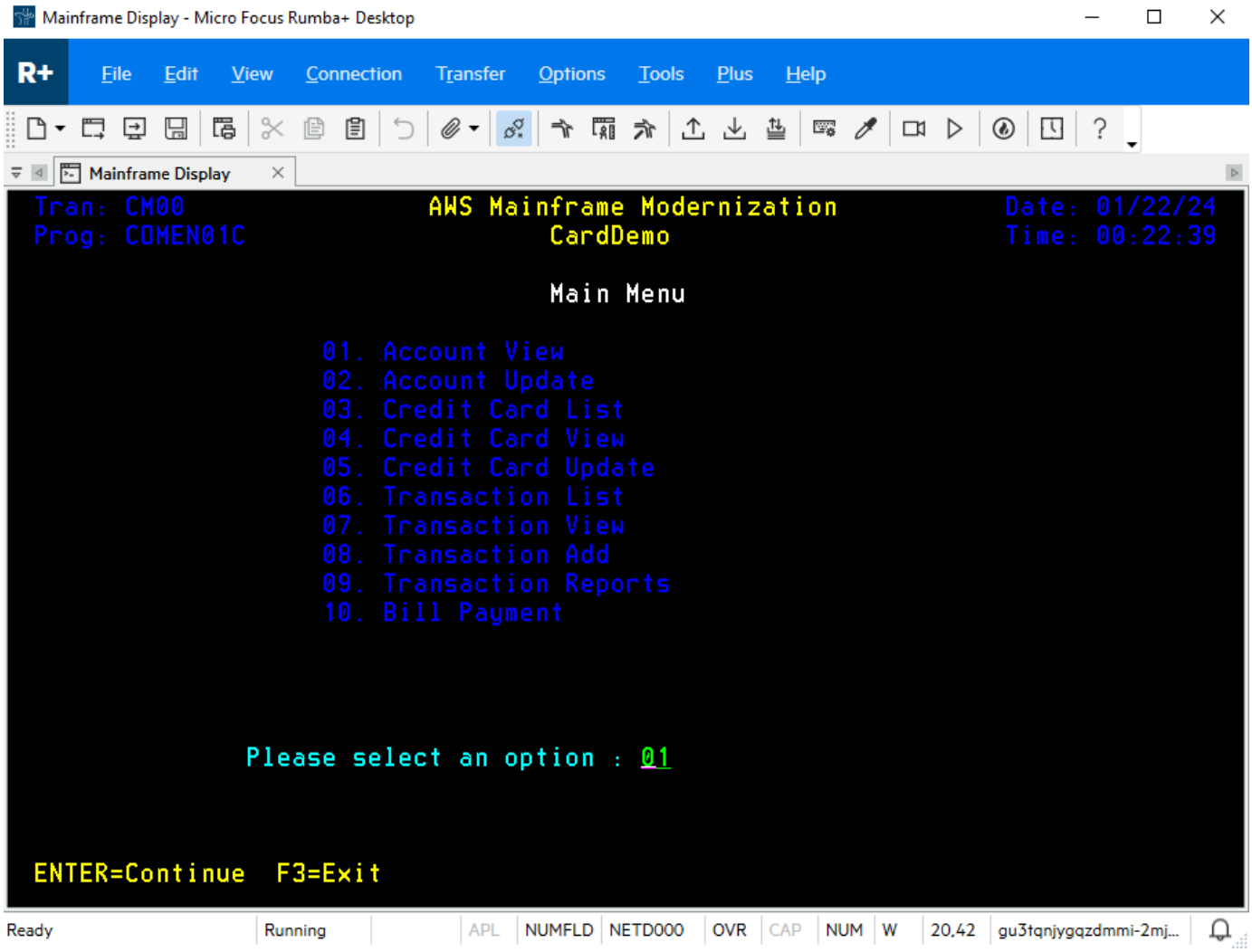
lugar (ou seja, Bloco CIDR 0.0.0.0/0). Se você se conectar com sucesso, saberá que o grupo de segurança estava bloqueando seu tráfego. Altere a fonte do grupo de segurança para algo mais específico. Para obter mais informações sobre grupos de segurança, consulte Noções básicas sobre grupos de segurança.

10. Digite `USER0001` o nome de usuário e `password` a senha.

> **Note**
>
> No Rumba, o padrão para Limpar é ctrl-shift-z, e o padrão para Redefinir é ctrl-r.

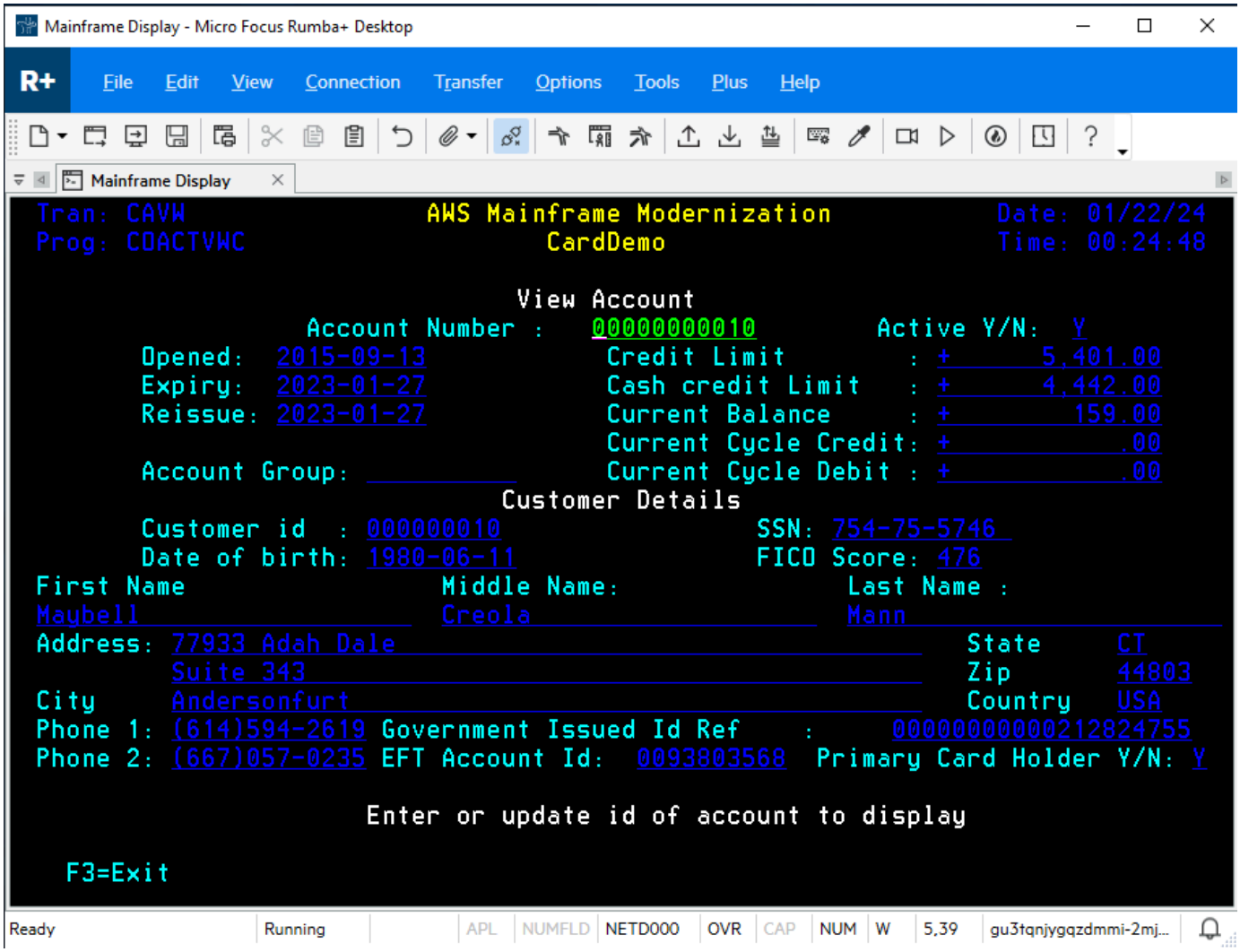



11. Depois de fazer login com sucesso, você pode navegar pelo CardDemo aplicativo.
12. Entre `01` para visualizar a conta.




13. Digite `00000000010` o número da conta e pressione Enter no teclado.

> **Note**
>
> Outras contas válidas são `0000000011` `00000000020` e.




14. Pressione F3 para sair do menu e F3 para sair da transação.

## Limpeza de recursos

Se os recursos criados para este tutorial não forem mais necessários, exclua-os para evitar cobranças adicionais. Para fazer isso, conclua as etapas a seguir:

- Se necessário, interrompa o aplicativo.
- Exclua o aplicativo do . Para ter mais informações, consulte Excluir um aplicativo do AWS Mainframe Modernization.
- Exclua o ambiente de execução. Para ter mais informações, consulte Excluir um ambiente de execução do AWS Mainframe Modernization.

- Exclua os buckets do Amazon S3 que você criou para este tutorial. Para obter mais informações, consulte Chaves de bucket do Amazon S3 no Guia do usuário do Amazon S3.
- Exclua o AWS Secrets Manager segredo que você criou para este tutorial. Para obter mais informações, consulte Excluir um segredo.
- Exclua a chave KMS que você criou para este tutorial. Para obter mais informações, consulte Excluir chaves do AWS KMS.
- Exclua o banco de dados do Amazon RDS que você criou para este tutorial. Para obter mais informações, consulte Excluir a instância do EC2 e a instância de banco de dados no Guia do usuário do Amazon RDS.
- Se você adicionou uma regra de grupo de segurança para a porta 6000, exclua a regra.

## Próximas etapas

Para saber como configurar um ambiente de desenvolvimento para seus aplicativos modernizados, consulte o Tutorial: Configurar AppStream 2.0 para uso com o Micro Focus Enterprise Analyzer e o Micro Focus Enterprise Developer.

# Abordagem de modernização

A migração é complexa e tem muitas variáveis. O AWS Mainframe Modernization oferece uma abordagem evolutiva que proporciona alguns ganhos de curto prazo ao melhorar a agilidade com muitas oportunidades de otimizar e inovar posteriormente. Além disso, o AWS Mainframe Modernization ajuda a simplificar a jornada e ainda respeita as particularidades da empresa e dos negócios de seu cliente. As duas principais abordagens suportadas pelo AWS Mainframe Modernization são a refatoração automatizada ou a reformulação. A escolha depende da situação do seu cliente.

A refatoração automatizada usa ferramentas AWS Blu Age para converter automaticamente código, dados e dependências em linguagem, armazenamento de dados e estruturas modernas, ao mesmo tempo em que garante a equivalência funcional com as mesmas funções de negócios.

O Replatforming usa as ferramentas da Micro Focus para transformar cargas de trabalho de mainframe em serviços ágeis em. AWS

Você pode pensar na jornada de modernização em etapas. A primeira etapa inclui três fases: avaliar, mobilizar, migrar e modernizar. A próxima etapa inclui a fase de operação e otimização, na qual você pode identificar mais oportunidades de inovação.

Tópicos



# Fase de avaliação

No nível mais alto, a fase Avaliação analisa se você está pronto para migrar. Você define um caso de negócios e, em seguida, educa sua equipe com workshops e um dia de imersão (demonstrações e laboratórios) oferecidos pela. AWS Workshops e dias de imersão abordam temas diferentes. Essas tarefas são realizadas fora do AWS Mainframe Modernization.

# Fase de mobilização

Na fase de Mobilização, você inicia seu projeto com um pontapé inicial e, em seguida, executa um processo de descoberta que extrai dados de seus aplicativos de mainframe e os ingere em uma ferramenta de migração. Você identifica os aplicativos que deseja migrar e seleciona alguns aplicativos para testar. Você refina seu caso de negócios, elabora seu plano de migração e decide como deseja lidar com segurança e conformidade, governança de contas e seu modelo operacional. Você configura um centro de excelência em nuvem com as pessoas certas da sua equipe. Você dirige os pilotos e documenta o que aprendeu. Você refina seu plano de migração e seu caso de negócios. Muitas dessas tarefas são realizadas fora do AWS Mainframe Modernization.

# Fase de migração e modernização

A fase de migração e modernização se aplica a cada aplicativo e consiste em várias tarefas, incluindo designar pessoas, executar descobertas aprofundadas, descobrir a arquitetura correta do aplicativo, configurar ambientes de tempo de execução de aplicativos AWS, reformular a plataforma ou refatorar seu código, integrá-lo a outros sistemas e, é claro, testar. No final da fase, você implanta os aplicativos reformulados ou refatorados na produção e transfere para o novo sistema na AWS. A maioria ou todas essas tarefas são realizadas no AWS Mainframe Modernization, em outro serviço do AWS ou em uma ferramenta à qual o AWS Mainframe Modernization fornece acesso.

Se você quiser usar a refatoração automatizada, consulte Blu Insights. AWS O Blu Insights agora está disponível a AWS Management Console partir do login único. Você não precisa mais gerenciar credenciais separadas do AWS Blu Insights. Você pode acessar os recursos do AWS AWS Blu Age Codebase e do Transformation Center diretamente do AWS Management Console.

Para migrar dados do mainframe para AWS, recomendamos o AWS SCT e o AWS Database Migration Service. Para obter mais informações, consulte O que é o AWS Schema Conversion Tool? no Guia do usuário do AWS Schema Conversion Tool e O que é o AWS Database Migration Service? no Guia do usuário do AWS Database Migration Service.

# Opere e otimize a fase

Na fase Operar e Otimizar, você se concentra em monitorar seus aplicativos implantados, gerenciar recursos e garantir que a segurança e a conformidade estejam atualizadas. Você também avalia as oportunidades de otimizar as cargas de trabalho migradas.

# Conceitos

AWS A modernização do mainframe fornece ferramentas e recursos para ajudá-lo a migrar, modernizar e executar cargas de trabalho do mainframe. AWS

Tópicos



## Aplicativo

Uma carga de trabalho de mainframe em execução na modernização do AWS mainframe. Um conjunto de trabalhos em lotes, transações interativas (CICS ou IMS) ou outros componentes compõem um aplicativo. Você define o escopo. Você deve definir e especificar quaisquer componentes ou recursos que a carga de trabalho precise, como transações do CICS ou trabalhos em lotes.

## Definição do aplicativo

A definição ou especificação dos componentes e recursos necessários para um aplicativo (carga de trabalho do mainframe) executado na modernização do AWS mainframe. Separar a definição do

aplicativo em si é importante porque é possível reutilizar a mesma definição para vários estágios (pré-produção, produção), representados por diferentes ambientes de tempo de execução.

# Trabalho em lote

Um programa agendado que é configurado para ser executado sem exigir interação do usuário. No AWS Mainframe Modernization, você precisará armazenar os arquivos JCL do trabalho em lote e os binários do trabalho em lote em um bucket do Amazon S3 e fornecer o local de ambos no arquivo de definição do aplicativo. Quando você executa um trabalho em lotes, a Modernização do AWS Mainframe relata os seguintes valores de status:

Enviando

A tarefa em lote está em processo de envio.

Em espera

O trabalho em lotes está suspenso.

Expedição

A tarefa em lote está em processo de envio.

Executando

O trabalho em lote está em execução no momento.

Cancelando

A tarefa em lote está em processo de cancelamento.

Cancelado

O trabalho em lotes foi cancelado.

Bem-sucedida

A execução do trabalho em lote foi concluída com êxito.

Com falha

O trabalho em lotes falhou.

Foi bem-sucedido com o aviso

A execução do trabalho em lote foi concluída com êxito, com um pequeno erro relatado. O código de condição do trabalho retornado como parte da GetBatchJobExecution resposta indica a causa do erro.

## Configuração

As características de um ambiente ou aplicativo. As configurações do ambiente consistem em tipo de mecanismo, versão do mecanismo, padrões de disponibilidade, configurações opcionais do sistema de arquivos e muito mais.

As configurações do aplicativo podem ser estáticas ou dinâmicas. As configurações estáticas mudam somente quando você atualiza um aplicativo implantando uma nova versão. As configurações dinâmicas, que geralmente são uma atividade operacional, como ativar ou desativar o rastreamento, mudam assim que você as atualiza.

## Conjunto de dados

Um arquivo contendo dados para uso por aplicativos.

## Ambiente

Uma combinação nomeada de recursos AWS computacionais, um mecanismo de tempo de execução e detalhes de configuração criados para hospedar um ou mais aplicativos.

## Mainframe Modernization

O processo de migração de aplicativos de um ambiente de mainframe legado para o. AWS

## Jornada de migração

O end-to-end processo de migração e modernização de aplicativos legados, normalmente composto pelas seguintes fases: avaliar, mobilizar, migrar e modernizar e operar e otimizar.

## Ponto de montagem

Um diretório em um sistema de arquivos que fornece acesso aos arquivos armazenados nesse sistema.

## Refatoração automatizada

O processo de modernização de artefatos de aplicativos legados para execução em um ambiente de nuvem moderno. Ele pode incluir conversão de código e dados. Para obter mais informações, consulte AWS Mainframe Modernization Automated Refactor.

## Reestruturação de plataformas

O processo de mover um aplicativo e seus artefatos de uma plataforma de computação para outra. Para obter mais informações, consulte AWS Mainframe Modernization Replatform

## Recurso

Um componente físico ou virtual em um sistema de computador.

## Mecanismo de execução

Software que facilita a execução de um aplicativo.

# Refatorando aplicativos automaticamente com o Blu Age AWS

A refatoração automatizada com o AWS Blu Age fornece uma end-to-end solução para migrar e modernizar seus aplicativos de mainframe. As etapas do processo de refatoração são as seguintes:

- Analise o inventário
- Analise dependências
- Transformar código automaticamente
- Capture e gerencie cenários de teste

Você pode concluir as etapas anteriores na ferramenta Blu Insights, disponível por meio de login único no console de modernização do AWS Mainframe. Para obter mais informações sobre o Blu Insights, consulte a documentação do Blu Insights.

Quando estiver satisfeito com o código-fonte transformado, é hora de mudar para AWS, onde você concluirá as seguintes etapas:

- Crie e implante o aplicativo refatorado.
- Implante e monitore seu aplicativo na modernização AWS do mainframe.

AWS O Blu Age Runtime (não gerenciado) é uma das ofertas do serviço de modernização de AWS mainframe junto com o Blu Age gerenciado. AWS Com o AWS Blu Age gerenciado, você pode implantar seu aplicativo modernizado em um ambiente AWS gerenciado que simplifica sua experiência, para que você não precise gerenciar a infraestrutura subjacente que executa seu aplicativo modernizado. Por outro lado, com o AWS Blu Age Runtime (não gerenciado), você pode implantar seu aplicativo modernizado em sua própria AWS conta, para poder gerenciar sua própria infraestrutura. Com o AWS Blu Age Runtime (não gerenciado), você tem a flexibilidade de operar todos os componentes técnicos necessários para executar seu aplicativo modernizado da maneira que desejar.

AWS O Blu Age Runtime (não gerenciado) está disponível para implantação no Amazon EC2 e no Amazon ECS em. AWS Fargate

Tópicos

# AWS Notas de lançamento do Blu Age

Esta seção contém as notas de lançamento do AWS Blu Age Runtime and Modernization Tools da versão 3.5.0 em diante, a mais recente, organizada por número de versão.

> **Note**
>
> Para notas de lançamento anteriores a este documento, entre em contato com os serviços de entrega da AWS Blu Age. Para obter informações sobre os recursos mais recentes do Blu Insights, consulte Blu Insights Releases.

Tópicos

## Notas de lançamento 3.10.0

Esta versão do AWS Blu Age Runtime and Modernization Tools se concentra nas principais atualizações e melhorias básicas em todo o produto, buscando aumentar o desempenho e a robustez em todas as etapas de transformação e execução. Alguns dos principais recursos e mudanças nesta versão são:

- Atualização da versão do Java 8 para o Java 17, aumentando a segurança e o desempenho e permitindo que os clientes implantem e executem aplicativos implementados em uma linguagem mais moderna e usem versões recentes da estrutura de terceiros.
- Suporte adicional para gerenciar grandes espaços de memória compartilhada entre usuários ou trabalhos, armazenando dados reutilizáveis após a reinicialização do aplicativo ou da instância.
- Acesso mais rápido a grandes conjuntos de dados no Blusam usando um mecanismo de paginação que possibilita a recuperação incremental de um subconjunto de registros.

Para obter mais informações sobre as alterações incluídas nesta versão, consulte as seguintes seções:

## Runtime versão 3.10.0

Esse tempo de execução é baseado em Java17, Spring2.7 e Angular16.

### zOS

Novos atributos

- Blusam - Adicionado suporte para grandes conjuntos de dados por meio de um mecanismo paginado em que os índices são armazenados e carregados usando páginas

### Melhorias

- DataUtils.compare aprimorado para lidar com a conversão de precedência mais baixa de string para número
- Foi adicionado suporte para verificar se não ByteRange é criado com valores impróprios por meio da propriedade YML DataSimplifier. byteRangeBoundsVerifique
- RemoveSosi () aprimorado para suportar a inicialização de um com um GraphicAlphanumericType caractere vazio
- Robustez adicional para operação de trabalho e leitura segura do estado do GDG
- Blusam - Foi adicionado suporte para limpar o Ehcache dos conjuntos de dados Blusam por meio de um novo método chamado .removeCache () CoreBluesamManager
- Blusam - Melhor comportamento de exclusão/renomeação para conjuntos de dados regulares do Blusam
- Redis - Suporte aprimorado para desbloquear conjuntos de dados e limpar o bloqueio de registros
- JICS - Melhorou a mensagem de erro para solicitações com falha
- JCL - Adicionado suporte para concatenação de variáveis ControlM com base no caractere de ponto
- JCL - Adicionado suporte para Write ADVANCING (ADV) para arquivos GDG
- JCL - Suporte aprimorado para o número da geração atual após excluir todos os arquivos GDG
- JCL - Suporte aprimorado para leitura de RDW/RecordSize do catálogo na criação do conjunto de dados
- JCL - Adicionado suporte para atualizar o objeto de recurso (de AbstractSequentialFile) ao abrir o arquivo com o tamanho do registro de saída de dados
- JCL - Melhor desempenho do IDCAMS
- JCL - Suporte aprimorado para PRINT STATEMENT adicionando “CHAR” como alias de “CHARACTER”
- SORT - Suporte aprimorado para operação de cópia de um conjunto de dados de tamanho fixo Blusam para um conjunto de dados com comprimento variável
- SORT - Gramática de classificação aprimorada para lidar com algumas declarações específicas

## COMO 400

### Novos atributos

- Foi adicionado suporte para espaços de usuário e suas APIs relacionadas
- Foi adicionado suporte para o parâmetro TOMSGQ do SNDPGMMSG e implementou filas de mensagens
- CL - Adicionado suporte para os parâmetros FILE e SPLFNAME para o comando OVRPRTF
- CL - Adicionado suporte para lidar com bibliotecas para a tabela de partições correspondente com o comando CPYF
- CL - Adicionado suporte para lidar com o comando CHGCURLIB e considerar a biblioteca atual ao criar consultas
- CL - Adicionado suporte para lidar com o comando cl como parte da chamada stacktrace

Melhorias

- Aprimorado MessageHandlingBuilder para melhor lidar com a entrada de rastreamento da pilha de chamadas
- Execução paralela aprimorada do recurso ContextPreconstruct
- Atributos de exibição aprimorados quando um registro é criado pelo SFLINZ
- SAVOBJ aprimorado para permitir o manuseio de vários arquivos de saída
- Manipulação aprimorada de programas groovy ao adicioná-los programCallStack quando são chamados a partir de um programa Java
- Detecção aprimorada do posicionamento superior do modal de ajuda
- Funcionalidade TopGMQ aprimorada quando o parâmetro TomSGQ é fornecido para SNDPGMMSG
- Busca aprimorada de mensagens predefinidas e funcionalidade do carregador de mensagens
- Manipulação aprimorada do CPYTOIMPF de caracteres delimitadores no conteúdo
- Bloqueio de liberação aprimorado no registro READ

## Recursos transversais

Novos atributos

- Foi adicionada uma tradução para mensagens do sistema no Front-End
- Foi adicionado um novo método ExecutionContext para retornar a pilha de chamadas do programa
- Defina um separador de linha (para simplificador de dados), independentemente do ambiente real

- Foi adicionada a possibilidade de configurar o caminho JSON do modelo SQL

Melhorias

- Melhorou o método de comparação DataUtils. compareAlphInt() quando o preenchimento está envolvido
- Criação de um sinalizador para permitir comportamento personalizado em caso de exceção nas consultas do cursor
- Conversão gráfica aprimorada de LOWVALUES db

Terceiro

- Atualização para mitigar CVE-2024-21634, CVE-2023-34055, CVE-2023-34462, IN1-JAVA-ORG/SPRINGFRAMEWORKSECURITY-5905484, CVE-2023-46120, CVE-2023-6481, CVE-2023-6378, CVE-2023-5072)

## Ferramentas de modernização versão 3.10.0

### zOS

Melhorias

- COBOL - Adicionado suporte para a função ABS
- JCL - Escopo variável aprimorado: anexado ao STEP em vez de JOB
- Injeção aprimorada de parâmetros do cursor para valor baixo/alto
- Análise aprimorada de CSD, principalmente para TRANSAÇÕES remotas

### AS400

Melhorias

- Verificação em branco removida para o indicador de nível de controle
- Adicionado suporte para nome externo para palavras-chave de IMPORTAÇÃO/EXPORTAÇÃO
- Foi adicionado suporte para %LEN em campos
- CL - Adicionado suporte para novos operadores para a linguagem CLLE

- CL - Adicionado suporte para IF aninhado
- COBOL - Melhor manipulação do comando START quando usado com várias teclas
- DSPF - Melhor manipulação da posição do cursor com número de registro
- DSPF - Melhorou a formatação para campos numéricos assinados, somente numéricos e campos em grande escala
- DSPF - Melhorou a determinação do título para o Screen General Help
- DSPF - Suporte aprimorado das especificações de entrada/saída
- DSPF - Melhor manuseio de separadores de agrupamento durante a validação do campo numérico
- Saída de mapeamento/registros DDS aprimorados
- Capacidade aprimorada da palavra-chave REFFLT do arquivo de impressora para resolver campos referenciados
- RPG - Suporte aprimorado para declarações “TODAS gratuitas”
- RPG - Análise de condições aprimorada e suporte adicionado para lidar com CABXX sem TAG de resultado
- RPG - Manipulação aprimorada da especificação de entrada de campos numéricos
- RPG - Melhor tratamento de chamadas de procedimentos dentro das condições IF/ELSEIF/WHEN
- RPG - Melhor manipulação do comando READ quando chamado em um arquivo dspf
- RPG - Melhore o suporte para arquivos referentes a um DDS inexistente
- Melhore o manuseio do REFFLD ao receber um nome de formato de registro físico
- Suporte adicionado para usar 'return' como nome de coluna db

## Recursos transversais

### Novos atributos

- Oracle - Tornou possível definir usuários além de SYS para armazenar funções integradas

### Melhorias

- Versão Java atualizada da v8 para a v17
- Condição SQL aprimorada com o nome da coluna Cluster

- Adicionado suporte para cláusulas ORDER BY a partir da visualização

## Notas de lançamento 3.9.0

Esta versão do AWS Blu Age Runtime and Modernization Tools está focada em vários aprimoramentos transversais em todo o produto, buscando aumentar o desempenho em arquiteturas de alta disponibilidade, além de novos recursos para elevar a execução de tarefas a um novo patamar. Alguns dos principais recursos e mudanças nesta versão são:

- Atualização da versão do Angular 13 para o Angular 16, aumentando a segurança e dando acesso a novos recursos que melhoram o desempenho nas aplicações on-line do cliente.
- Adicione suporte aos recursos de trabalhos cruzados no AS400, com o principal destaque de que os trabalhos podem enviar mensagens de consulta de forma síncrona entre eles, permitindo a dissociação em trabalhos modernizados.
- Melhorias de desempenho no uso do Redis, incluindo otimização do pool de conexões, alta segurança na conexão e mecanismo de bloqueio de conjunto de dados atualizado.

Para obter mais informações sobre as alterações incluídas nesta versão, consulte as seguintes seções:

## Runtime versão 3.9.0

### zOS

Novos atributos

- Programa de classificação: entradas VSAM atualizadas com tamanho fixo
- JHDB DB: tempo limite configurável adicionado

Melhorias

- Suporte aprimorado para o separador de linha transmitir se usado na concatenação de arquivos
- Suporte aprimorado para abrir arquivos sequenciais concatenados. Inicializar DataSetIndex após a abertura do arquivo
- Suporte aprimorado para separador decimal virtual quando a NumericEditedType é afetado por um valor numérico

- Suporte aprimorado para NumericEditedType valores não negativos
- IDCAMS: Os cartões SYSIN agora são lidos usando a propriedade “encoding” definida em .yml application-utility-pgm
- IDCAMS: gramática atualizada para a compatibilidade com o argumento FILE (..) na instrução DEFINE CLUSTER
- INFUTILB: adicionado suporte ao argumento DFSIGDCB para substituir os parâmetros DCB de DD SYSREC
- INFUTIL: suporte aprimorado para o parâmetro “DFSIGDCB YES”
- SPLICE aprimorado para lidar com um grande arquivo de entrada
- DFSORT: melhor tratamento dos campos de observação
- DFSORT: adicionado suporte para o formato numérico de formato livre (assinado/não assinado) (SFF/UFF)
- SORT: adicionado suporte de análise para as instruções OPTION PRINT e OPTION ROUTE
- SORT/ICEMAN: adicionado suporte a operações de divisão incluídas (campo com operador DIV)
- Suporte aprimorado para CICS READ usando uma chave genérica
- Função StringUtils .chargraphic corrigida para remover SOSI de um tipo gráfico
- Melhore o desempenho ativado DataUtils. isDoubleByteCodificação
- JCL: suporte aprimorado para o modo de disposição KEEP para um conjunto de dados temporário. O sistema muda a disposição para PASS
- JCL: manipula parâmetros DCB dinamicamente
- JCL: saídas aprimoradas de SUM FIELDS para valores incorretos
- JCL: CommonDDUtils::getContent agora procura o recordSize no catálogo
- JCL: leia os atributos rdw/recordSize do catálogo na criação do conjunto de dados
- JCL: adicionado suporte a DCB=.MYDD para copiar parâmetros DCB de um DD para outro na mesma etapa do trabalho
- JCL: aprimorado o sistema de herança de tamanho de registro
- JCL: Adicionado bloqueio de conjunto de dados exclusivo (Redis)
- Redis: adicionado suporte a SSL para o modo autônomo
- Redis: adicionada a contagem sincronizada de bloqueios do Redis com bloqueio
- Redis: parâmetros de pool compatíveis para o bloqueio do Redis
- Redis: atualização otimizada de metadados com o Redis

- Redis: suporte aprimorado ao cluster do redis
- Melhoria nos bloqueios abertos com o modo do IO
- Desempenho aprimorado dos bloqueios de conjuntos de dados e eliminação de bloqueios não utilizados
- Caminho aprimorado do conjunto de dados durante o cancelamento do registro do arquivo
- Invalidação aprimorada do cache da janela de pré-busca
- Adicionado suporte para o uso do provedor de fonte de dados do utilitário de thread seguro
- Verificação aprimorada de nulidade do datasetState
- Suporte aprimorado para não reabrir conjuntos de dados já abertos
- Maior robustez para a operação final do trabalho
- Suporte aprimorado para a ordem dos índices para as chaves, permitindo duplicidades
- Suporte aprimorado para ignorar a ordem de serialização da lista
- Adicionado suporte ao recurso de depuração e despejo para ajudar a diagnosticar problemas de ordem dos índices
- Suporte aprimorado para a atualização de metadados
- Suporte aprimorado para leitura em massa Blusam

## COMO 400

Novos atributos

- Cria um registro de contexto da aplicação
- Suporte à palavra-chave DSPF CLRL(NO) Suporte ao monitoramento de bloqueios de registros
- Support para keyed DataQueue
- Suporte a mensagens INQUIRY para trabalhos em lote
- Adicionado suporte ao arquivo de impressora descrito pelo programa para AS400 COBOL
- Manipula o comando RMVJOBSCDE cl
- Melhoria para RUNSQL/DLYJOB
- CHKOBJ: aumento do código de erro herdado para o parâmetro LIB
- SNDPGMMSG: suporte a parâmetros de string
- RTVDTAARA: Substring aprimorada no LDA
- DSPFD: suporte adicionado ao parâmetro FILE para o nome de arquivo específico

- RUNQRY: suporte ao arquivo sql no QRY PARAM
- CRTDUPOB: suporte à cópia de dados entre áreas de dados
- SBMJOB: converte instruções para uso JobQueueManager
- OPNQRYF: suporte adicionado para a biblioteca Qtemp
- CRTDUPOBJ: Lógica aprimorada para copiar o conteúdo da partição
- CRTDUPOBJ: suporte adicionado a Qtemp para visualizações
- RTVSYSVAL: suporte ao valor SYSVAL, QDATFMT no comando CL
- CHKOBJ: suporte adicionado a OUTQ
- RTVJOBA: suporte ao parâmetro SWS
- SNDPGMMSG e RCVMSG: parâmetros adicionais com suporte a MSGF, MSGFLIB, MSGDTA, MSGTYPE, KEYVAR, MSGKEY, MSGID

Melhorias

- Suporte aprimorado para placas de E/S da ESTAÇÃO DE TRABALHO
- Tratamento aprimorado da mensagem definida sobrepondo a mensagem anterior
- Suporte a informações adicionais de mensagens na array-messageline
- Acesso aprimorado ao wrapper da matriz autônoma dentro de EVAL, SortA e figurativos
- Melhorar a limpeza de DAOs quando a aplicação on-line for encerrada
- Adicionado suporte a formatos de data adicionais e melhora no tratamento de entradas de string
- Manipulação aprimorada do CVTDAT do SYSVAL adicionando parâmetros de decodificação e construção da classe auxiliar de valor do sistema a partir do comando CL SbmJob
- O pacote com.netfective.bluage.gapwalk.rt.blu4iv foi removido da verificação de componentes gapwalk-cl-command
- Aprimorado o suporte de mensagens predefinidas à API de fila de mensagens
- Melhorou o suporte retrieveSubfileRecord para registro escrito em outro programa
- Aprimorado o suporte de mensagens imediatas à API de fila de mensagens
- Tratamento aprimorado da área de dados locais ao enviar um trabalho
- Inicia JobQueues automaticamente quando o servidor é iniciado
- Usa a configuração applicationContext para decodificar parâmetros para SBMJOB
- Melhoria nas mensagens de erro fornecidas pelo sistema

- Permite que RTVMSG pesquise arquivos .properties em subdiretórios aninhados
- Lida com a redefinição de entidades vinculadas a ponteiros inválidos
- Melhorado MessageHandlingBuilder para exibir msgID e MsgFile nome como strings para RCVMSG
- Método de withMsgFile nome aprimorado da API de enfileiramento de mensagens
- Aprimorado o mecanismo de bloqueio da área de dados
- RTVMBRD: suporte a letras maiúsculas e minúsculas para o parâmetro FILE
- CRTDUPOBJ: melhoria no tratamento das visualizações
- CPYTOSTMF: melhoria no tratamento da conexão
- CPYF: melhoria no tratamento do nome do diretório ao copiar de um arquivo simples
- RCVF: lida adequadamente com os parâmetros DEV/RCDFMT e com a transformação de RCDFMT para groovy e java
- RCVF: lida com chamadas subsequentes e evita redefinir o cursor
- CPYF: adicionado suporte à gravação a partir de arquivos simples
- CRTDUPOBJ: adicionado o tratamento de novos obj com a biblioteca Qtemp
- CHGDTAARA: aumento do tamanho máximo da área de dados de 256 para 2.000
- SAVOBJ: certifique-se de que os registros salvos estejam na ordem de inserção
- RTVDTAARA: valores recuperados (não devem ser cortados)
- CHKOBJ: retorna as mensagens corretas do monitor quando o membro não existe
- RTVDTAARA: adicionado suporte à substring LDA
- RTVDTAARA: retorna espaços em branco até o tamanho da variável especificada no parâmetro RTNVAR
- RTVDTAARA: suporte a parâmetros inteiros para início e tamanho e suporte ao formato de transformação mais recente
- CHGDTAARA: adicionado suporte a parâmetros que incluem limites inferiores e superiores
- CHKOBJ: lida com o valor VIEW para o tipo de objeto do parâmetro
- CHKOBJ: resultado definido como verdadeiro, independentemente do membro se a visualização existir

## Recursos transversais

### Novos atributos

- Lida com a geração de relatórios para arquivos .txt
- Adicionada a propriedade da fonte de dados currentSchema XA ao gerenciador de segredos
- Adicionar a propriedade yml database.cursor.raise.already.opened.error para permitir que a framework gere o erro 502 do SQLCODE quando o cursor já aberto for aberto

Melhorias

- Adicionamos gapwalk poms ao AWS Blu Age na embalagem do Amazon EC2
- Usa o novo paradigma do manipulador de sinais por padrão
- Adicionar suporte ao bloqueio quando a disposição for MOD ou OLD
- Cache adicionado para armazenar padrões de data e hora do banco de dados
- Função de verificação aprimorada do PackedType
- Melhore as DataUtils funções.setTo para registros com VariableSizeArray
- Lida com a opção MQ SYNCPOINT em relação à unidade de execução
- Framework habilitada para definir SQLCODE na transação de reversão
- Adicionado nome automático da classe do driver de acordo com o segredo da chave do mecanismo
- Tempo limite do programa/da transação
- Restaurar a posição do cursor após a reversão ao acessar o cursor

Terceiro

- Atualize o SnakeyAML, o Redisson e o Amazon SDK, YamlBeans remova (reduza CVE-2022-25857, CVE-2023-24621, CVE-2023-42809, CVE-2023-44487)

# Ferramentas de modernização versão 3.9.0

## zOS

Melhorias

- Suporte aprimorado para XML-TEXT como origem para destino do tipo String
- Fluxo de trabalho aprimorado de STM para UML para oferecer suporte ao padrão de divisão X/(Y/Z)

- JHDB DB: aceita a chamada ROLLBACK antes de qualquer atualização do banco de dados
- JHDB DB: aceita ROLLBACK mesmo se a transação for encerrada (NOP)
- JCL: aprimorada a função de validação de etapas
- SORT: manipula a função SUM com valores decimais negativos de zona
- COBOL: adiciona suporte a escape de aspas simples/duplas em literais de string

## AS400

Melhorias

- Aprimorada a manipulação da função integrada %editc do código de edição X adicionando-se zeros à esquerda
- Aprimorada a manipulação do valor inicial dos campos somente de entrada
- Adicionadas teclas de ação para ajudar os diálogos
- Registro de rodapé da tabela dinâmica que aparece na parte inferior
- Comando START manipulado sem KEY PHASE para arquivos que especificam uma RECORD-KEY real
- Valor padrão adicionado para os tipos float e NumberUtils: :pow
- Adicionado suporte à definição de uma variável usando LIKE(IN)
- Manipulação de loop FOR atualizada para oferecer suporte à omissão de elementos opcionais
- Atualizada a análise de RPG para associar registros ao nome da matriz CTDATA
- Tratamento aprimorado de indicadores para instruções CABxx
- Suporte a parâmetros opcionais na palavra-chave COMMIT
- Suporte aprimorado para palavras-chave FORMAT no LF
- Código de operação LOOKUP gerenciado com indicadores altos e iguais (ou baixos e iguais)
- Manipulação no nome da chave PF declarado entre aspas duplas
- Tratamento aprimorado de EDTCDE X para não suprimir os zeros iniciais
- Suporte aprimorado para MSGCON no arquivo da impressora que não gera etiquetas sem nome
- O campo CONTEÚDO é compartilhado por várias estruturas de dados
- Parâmetro ERRSFL tratado em combinação com SFLMSG/SFLMSGID
- Código principal aprimorado antes do escopo da declaração de proc do rpg gratuito completo
- Adicionada a especificação de controle condicionado de análise

- Suporte aprimorado para o método setErrSfl () no dataholdermapper
- Aprimorada a resolução de tipo para variáveis criadas internamente
- Suporte aprimorado para o código de operação Z-ADD
- Tratamento aprimorado do campo constante com valor de DFT
- Melhorar o suporte ao campo inteiro dentro do status ds do programa
- Atribuição de indicadores tratada nos parâmetros ENTRY
- Melhoria no filtro de palavras-chave propagadas por meio da palavra-chave ref/reffield
- Estrutura de DataArea dados sem nome suportada
- Tratamento aprimorado do tipo de dados do ponteiro
- Tratados os elementos da matriz usados para definir variáveis com acesso à matriz de suporte à palavra-chave LIKE, no campo de saída
- Suporte aprimorado para números assinados, exibindo somente dígitos
- Suporte para a relação lógica na placa O
- Caso de teste para %CHAR em alfanumérico
- Suporte à palavra-chave principal da especificação de controle
- EDTCDE com dois parâmetros no arquivo da impressora
- Análise aprimorada FullFree de RPG
- Aprimorada a tabela dinâmica para garantir que o rodapé seja posicionado corretamente
- Adicionado suporte para inicializar tipos numéricos com a constante figurativa TODOS
- Tratamento aprimorado de vários arquivos lógicos do RPG referenciando o mesmo arquivo físico
- Melhorar a detecção de campos modificados em uma tela moderna
- Sincronização de modal com campos dinâmicos
- Tratamento aprimorado do campo numérico assinado somente de saída
- Melhorar as placas de E/S da ESTAÇÃO DE TRABALHO

## Recursos transversais

### Novos atributos

- Ferramenta de migração de dados: propriedade ebcdicFilesWith VarcharIn VB adicionada para permitir levar em consideração o comprimento de 2 bytes do VARCHAR ao ler bytes
- Implementado uma API comum para registrar erros

- Implementação BluAgeErrorDictionaryUtils e uso de API comum para registrar erros e/ou informações em Cobol2Model, CycleBuilder RPG, Definitions2Model e FieldsProcessor
- Aprimorada a gramática SQL para oferecer suporte a diferentes definições de cláusulas de isolamento

Melhorias

- Atualizada a versão do Angular para v16
- Angular: atualizada a versão ajv de 6 para 8.9

Terceiro

- Atualizada a versão do Groovy para 2.4.15

## Notas de lançamento 3.8.0

Esta versão do AWS Blu Age Runtime and Modernization Tools está focada em vários aprimoramentos transversais em todo o produto para melhorar sua qualidade e segurança, além de melhorias no desempenho do armazenamento em cache e na unificação dos suportes de comandos em uma única distribuição. Alguns dos principais recursos e mudanças nesta versão são:

- Atualização da versão do Spring 2.5 para o Spring 2.7, aumentando o suporte de manutenção, o desempenho e a segurança da plataforma.
- Unificação do suporte de mais de 82 comandos CL como parte da over-the-counter distribuição para facilitar o uso e a implantação de aplicativos modernizados que antes usavam scripts CL.
- Novas APIs disponíveis para operar e interagir melhor com os conjuntos de dados BluSam, como importação integrada para o serviço gerenciado e a capacidade de listar informações de metadados do conjunto de dados.
- Melhorias de desempenho e extensão do uso do Redis, incluindo disponibilidade no modo de cluster, recuperação de dados de alta disponibilidade e padronização do uso de segredos.

Para obter mais informações sobre as alterações incluídas nesta versão, consulte as seguintes seções:

# Runtime versão 3.8.0

## zOS

### Novos atributos

- Manipulando a definição da chave como uma string para DynamicFileBuilder
- DFSORT: Adicionado suporte para vários itens na inicialização gramatical OUTFIL TRAILER1 + DFSORT
- Ferramenta CommonDDUtils: tratamento do tamanho do registro em dados in-stream
- Arquivo indexado: manipulando a opção GENKEY

### Melhorias

- Serviços de carregamento BlusAM externalizados em uma jarra separada
- Adicionado suporte à configuração do local para armazenar arquivos temporários
- Mecanismos aprimorados de cache compartilhado para casos de vários nós
- Uso de cache compartilhado: IDCAMS verifica a otimização
- Melhore a injeção de ROWID para seleção incorporada
- JCL: cada procedimento de trabalho in-stream agora é gerado em um arquivo groovy distinto
- Garanta card-demo-v 2 coberturas nos cartões IDCAMS JCL
- BlusAM: Evite o aquecimento duplicado ao usar várias instâncias
- Diminuição do consumo de memória na hidratação do cache
- Suporte de configuração do pool Jedis
- Separador de linha adicionado para transmitir se usado na concatenação de arquivos
- Support para cartões EBCDIC + blocos de comentários (/*.../) no utilitário IDCAMS
- Consulta de suporte ao banco de dados: suporte para cadeias de bytes duplos na conversão do nível 49 em SQL
- Gramática DFSORT: implementa 17 declarações de controle + integração de 2 delas (OMIT/INCLUDE)
- Melhore as colunas GRÁFICAS fetch INFUTILB
- Support para leitura de arquivo com tabela de tamanho variável

- Support for ZonedType with nibble signed, onde o primeiro bit do último byte é 'E'
- DFSORT/ICETOOL adiciona suporte ao argumento NOMATCH =(..) se um registro não corresponder a nenhuma das constantes de busca CHANGE
- Compatibilidade com o Redis Cluster
- Tratamento do Status do Job (Falha) com base no código de saída do Groovy
- Suporte aprimorado para o CICS SYNCPOINT ROLLBACK.
- Janela de pré-busca para otimizar o uso do cache do Redis
- JCL/GROOVY: herda a propriedade isRDW do conjunto de dados da etapa anterior quando DISP=(,PASS)
- Manipulação de cópia parcial de dados com matriz de tamanho variável

## COMO 400

### Novos atributos

- Support para placas de E/S para arquivos de exibição
- Support para informações adicionais de mensagens para as palavras-chave DSPF ERRMSGID e CHKMSGID
- Support para várias mensagens de erro na tela de front-end
- Suporte adicionado ou aprimorado de 82 comandos CL no gapwalk-cl-command aplicativo

### Melhorias

- Suporte aprimorado para DELETE e READ sob controle de compromisso
- ConvertDate dentro do %dec embutido
- Cabeçalhos de segurança XSS aplicados
- Maior robustez e consistência da geração de STM (melhor manuseio de: linha de continuação em RPG de formato livre, vírgulas para parte decimal, blocos de formato livre na definição/declaração)
- DataHolderMapper Geração aprimorada
- Robustez adicionada e mudança de escopo em DataAreaFactory
- Melhorou a mudança de foco na tecla tab
- Melhor desempenho na geração de relatórios do Jasper
- Tela decimal aprimorada com preenchimento 0s

- Suporte aprimorado para o campo ROW/COL no INFDS
- Melhore o suporte para campos modificados na tela
- Foram adicionados getters para nome e caminho do relatório gerado
- Melhorado no comprimento da fila de dados
- Configuração automática aprimorada de Job Queues para atender aos novos padrões no Spring Boot 2.7
- Atualizações aprimoradas da estação de trabalho para várias sessões simultâneas

## Recursos transversais

### Novos atributos

- Support to No Invalid Data Tolerance for Packaged
- Paginação/filtragem adicionada para listar os endpoints do conjunto de dados

### Melhorias

- Estratégia aprimorada de transformação de consultas ORACLE na comparação de colunas com uma string vazia
- Manipulando BLOB DB2 com programas utilitários DSNTEP e INFUTILB. O BLOB DB2 agora está modernizado para postgres do tipo BYTEA.
- Melhoria da exclusão do último item do cursor
- Suporte aprimorado para excluir arquivos RRDS
- Melhor desempenho secreto do AWS Blusam
- Manipulação aprimorada de conexões de banco de dados na estrutura SQL
- Chaves padronizadas do gerenciador AWS secreto de várias fontes de dados
- Correções de regressão de desempenho
- Função de verificação aprimorada para PackedType
- Melhor manuseio de LOW-VALUE para PackedType
- Pacote de segurança Spring atualizado para conexão cognito
- Não aplicar codificação e decodificação de codeshiftpoint em bancos de dados de destino do DB2

### Terceiro

- Atualização do Spring Boot de 2.5 para 2.7

# Ferramentas de modernização versão 3.8.0

## zOS

### Novos atributos

- JCL: Manipulação de fluxo com retorno de carro “\ r”

### Melhorias

- Registro aprimorado para evitar a divisão por zero ao modernizar uma cláusula DIVIDE com ON SIZE ERROR
- JCL: Suporte aprimorado para chamar um procedimento em um procedimento
- Support para a palavra-chave OF no comando FORMATTIME CICS quando há campos ambíguos
- JCL: suporte para o caractere Â⌘ em variáveis
- JCL: computação RC com base nas etapas anteriores
- Comparando bytes em vez de strings quando PL1 SUBSTR é usado
- Melhoria da inicialização de matrizes multidimensionais a partir de uma única fonte
- Análise aprimorada do COBOL quando envolve uma única consulta SQL em um bloco IF

## AS400

### Novos atributos

- Support para instrução IF aninhada em CL
- Suporte aprimorado para a declaração ENDDO em formato livre de RPG

### Melhorias

- Suporte aprimorado para nível de controle de condicionamento
- Retorno aprimorado do protótipo com LIKE
- Suporte aprimorado para lidar com funções %months, %year, %days
- Support for help feature para toda a tela

- Manipulação de espaços em branco figurativos transmitidos como parâmetro
- Melhoria na expressão EVAL com o operador “”
- Manipulando o comando START sem KEY PHASE
- Melhoria no manuseio da palavra-chave LIKEREC
- Melhoria em subcampos sem nome
- Melhoria no procedimento de devolução de um tipo não assinado
- Suporte aprimorado para a operação RESET (RPG gratuito), integrações de %CHAR e %DEC
- Melhoria na função integrada %LOOKUPXX
- Suporte aprimorado para a palavra-chave LIKEDS no procedimento sem protótipo
- Manipulando o tipo de matriz de palavras-chave Dim (VAR, AUTO)
- Suporte aprimorado para o XFOOT
- COBOL: suporte aprimorado para campos RENAMES
- CL: suporte enquanto condição (verdadeira)
- Melhorou o tratamento de matrizes autônomas com a palavra-chave LIKE
- Melhoria da função incorporada %INT
- Análise de RPG totalmente gratuita aprimorada
- Suporte aprimorado para matriz na ligação
- CL2GROOVY: Declaração de seleção de suporte
- Melhoria na palavra-chave DSPF “ERRMSGID”
- Melhorou o tratamento da inicialização de bytes com zeros à esquerda
- Melhoria nos valores autorizados para campos numéricos
- Manipulando o extensor H para declaração EVAL de formato livre
- CL para Groovy: Support a substring do LDA
- Suporte aprimorado para RESET em um registro
- Melhorou o tratamento de EDTCDE e EDTWRD com referências
- Mapeamento aprimorado do campo de entrada com campos DDS
- Suporte aprimorado para o caractere MOVEA para a matriz IN
- Melhoria no protótipo com a palavra-chave LIKEDS
- Suporte aprimorado para a palavra-chave DSPF de DSPATR

- Análise aprimorada do cartão D com +/-
- Maior robustez nas chamadas de programas
- Maior robustez no processo de resolução de campo

### Recursos transversais

Melhorias

- FrontEnd: Simule o evento de colagem para entrada IME

Terceiro

- Atualização do Spring Boot de 2.5 para 2.7

## Notas de lançamento 3.7.0

Esta versão do AWS Blu Age Runtime and Modernization Tools inclui principalmente aprimoramentos para oferecer melhor suporte a comandos e utilitários, recursos de integração com o AWS Secrets Manager e novos recursos de monitoramento. Algumas das principais alterações desta versão são:

- Agora, vários componentes de tempo de execução podem usar o AWS Secrets Manager para aumentar a configuração de segurança de aplicativos modernizados, principalmente relacionados a fontes de dados de serviços públicos, Redis para filas TS, BluSam cache e bloqueios.
- Endpoint de monitoramento que permite recuperar métricas de transação, lote e JVM para otimização do uso de recursos e gerenciamento operacional, como status, duração, volume e outros.
- Novos recursos para suportar chamadas IBM MQ em RPG e maior cobertura de transformação do JCL SORT e IDCAMS.

Para obter mais informações sobre as alterações incluídas nesta versão, consulte as seguintes seções:

## Runtime versão 3.7.0

Tópicos

## zOS

### Novos atributos

- Melhore as consultas de análise envolvidas no aplicativo utilitário do programa usando SQL como gramática. (V7-9401)
- Manipule a matriz de tamanho variável indexada quando deslocada (V7-9904)
- Support a coluna INSERT SQL TIME no DB2 com formato de 24:00:00 horas (V7-10023)
- Support a consulta INSERT SQL de matrizes com as opções FOR ROWS e ATOMIC (V7-10105)
- JCL SORT - aprimorado TranscodeTool para suportar OUTREC com IFTHEN (V7-10124)
- JCL SORT - adicione suporte para a palavra-chave DATE no comando OUTREC (V7-10125)
- JCL - adicione suporte aos procedimentos In-Stream (V7-10223)

### Melhorias

- Um conjunto de dados marcado com a disposição “PASS” deve estar disponível em todas as etapas do trabalho (V7-9504)
- Support JCL atributo SCHENV (V7-9570)
- Support SEND com opção CTLCHAR (V7-9714)
- COBOL - Manipule diferentes conjuntos de caracteres separadores de linha em declarações ACCEPT (V7-9875)
- Evite reversões múltiplas (V7-9958)
- Permitir o uso da disposição MOD para anexar no final dos arquivos GDG (V7-10031)
- Otimização: refatoração PutAll (V7-10063)
- PutAll refatoração: adição de paginação (V7-10063)
- Torne o tempo limite de leitura do cliente Jedis configurável (V7-10063)
- UseSsl suporte para o modo autônomo (V7-10114)
- Support EIBDS após abrir o arquivo com sucesso (V7-10147)

- Support EIBDS após uma solicitação de controle de arquivos (V7-10147)
- Melhore o suporte ao CICS SYNCPOINT (V7-10187)
- BluesamRedisSerializer: problema com a persistência de metadados (V7-10202)
- Support Redis AWS Secrets Manager para filas TS (V7-10204)
- Support JCLBCICS na personalização do tamanho do nome DD (V7-10224)
- Adiciona suporte para caminho absoluto na instrução IDCAMS DELETE (V7-10308)

## COMO 400

### Novos atributos

- Implementação do recurso de ajuda para telas AS400 (V7-9673)

### Melhorias

- Número de registros no INFDS (V7-9377)

## Recursos transversais

### Novos atributos

- Support for Runtime no EC2 para enviar registros para a Amazon CloudWatch (D87990246)
- Novo endpoint adicionado para recuperar métricas sobre lotes, transações e JVM (D88393832)

### Melhorias

- Support: fontes de dados do AWS Secrets Manager para utilitários pgm (V7-9570)
- Foi adicionado suporte ao Db2 para DSNUTILB DISCARD (V7-9798)
- Support para gravação no registrador em vez do fluxo de saída padrão do sistema nos arquivos SYSPRINT e SYSPUNCH padrão (V7-10098)
- Support BluSam Redis cache e bloqueia propriedades de conexão no AWS Secrets Manager (V7-10238)
- Support para conexão SSL no Db2 XA AWS secret (V7-10258)
- Metadados atualizados para IDCAMS REPRO e VERIFY (V7-10281)

- Gerenciamento aprimorado do código de retorno IDCAMS Abend (V7-10307)

# Ferramentas de modernização versão 3.7.0

Tópicos



## zOS

Novos atributos

- PLI - Atribuição aprimorada para seção transversal de matrizes e matrizes bidimensionais (V7-9830)

## COMO 400

Novos atributos

- Manipulação de indicadores de nível de controle (V7-9227)
- Support para o parâmetro EXTNAME *INPUT (V7-9897)
- Reescrita aprimorada do Goto: Suporte para tags localizadas em instruções SELECT OTHER (V7-9973)
- Support a palavra-chave REFSHIT DSPF (V7-10049)

Melhorias

- Melhoria no tratamento da palavra-chave de descrição de arquivo EXTIND (*inUX) (V7-7404)
- Transformação aprimorada de arquivos SQLDDS (V7-7687)
- Objetos de arquivo não são mais gerados para arquivos AS400 (V7-9062)
- Tratamento aprimorado da palavra-chave de descrição de arquivo EXTDESC (V7-9268)
- Manipulação aprimorada do %CHAR embutido (V7-9311)
- Suporte aprimorado para pagedown no último registro sem SFLEND (V7-9322)

- Suporte aprimorado para estruturas de dados prefixadas (V7-9436)
- Support para dimensão definida com %SIZE (V7-9472)
- Support para lidar com o nome do campo PF declarado entre aspas duplas (V7-9557)
- Operação de arquivo aprimorada - não diferencia maiúsculas de minúsculas (V7-9785)
- Support para campo inicializado para *USER (V7-9806)
- Support para o tipo COMP no AS400 (V7-9840)
- Análise aprimorada do COBOL400 em (Não) (V7-9922) InvalidKey
- Tratamento aprimorado da operação SCAN (V7-9971)
- Suporte aprimorado do código de operação GOTO (V7-9973)
- Manipulação aprimorada da operação EXCEPT (V7-9977)
- Suporte aprimorado a prefixos (V7-10000)
- Support para chamadas MQ em RPG (V7-10007)
- %LOOKUP integrado aprimorado (estrutura de dados de matriz com chave) (V7-10022)
- Support for Close *All operation (V7-10036)
- Support para a instrução SQLDDS UPDATE AS ROW CHANGE (V7-10051)
- Melhoria para lidar com o tipo de valor literal Long (V7-10073)
- Gramática RPG aprimorada (o uso da palavra-chave INZ como nome da sub-rotina) (V7-10074)
- Gramática RPG aprimorada para suportar valores numéricos com parte fracionária vazia (V7-10077)
- Suporte aprimorado para campos compartilhados entre CL e arquivo externo (V7-10081)
- Suporte aprimorado para indicadores condicionais do DDS (V7-10084)
- Support para o tipo binário DDS com programas COBOL (V7-10100)
- Melhor colisão de nomes com ligação (V7-10109)
- Support para misturar procedimentos principais e de exportação (V7-10112)
- Suporte aprimorado para DataStructure em um subprocedimento (V7-10113)
- Suporte aprimorado do CLEAR (V7-10126)
- Suporte aprimorado do loop DO (V7-10134)
- Support SQLTYPE em RPG totalmente gratuito (V7-10151)
- Análise aprimorada das condições na palavra-chave DDS (V7-10155)
- Geração DSL aprimorada (V7-10163)

- Melhoria para ProcessIndicators quando a condição é uma expressão binária. (V7-10164)
- GoTos aprimorado com condição Else (V7-10168)
- Support para o tipo Time and Timestamp no DSPF (V7-10173)
- Análise aprimorada da linha de continuação para DDS (V7-10183)
- Suporte COBOL para RENAMES FLD OF RECORD (V7-10195)
- Análise aprimorada de indicadores condicionais em campos DSPF (V7-10221)
- Support a análise da palavra-chave DDS NOALTSEQ (V7-10288)
- Menu Support Help e campos ocultos (V7-10314)
- Verificação aprimorada da integridade das palavras-chave de ajuda do DSPF (V7-10328)
- Não está mais propagando todas as palavras-chave no campo Ref (V7-10347)

### Recursos transversais

Novos atributos

- Migrador de dados - Tratamento de dados CLOB (V7-9665)

Melhorias

- Propagando a propriedade JCL SCHENV da definição JOB para PROC GROOVY por meio de (V7-10225) JobContext
- FrontEnd - Ajustar o tamanho da janela em caso de ausência de borda (V7-10358)

## Notas de lançamento 3.6.0

Esta versão do AWS Blu Age Runtime and Modernization Tools fornece novos recursos para migrações legadas do zOS e do AS400, principalmente orientadas para expandir os mecanismos de suporte do CICS, complementar os recursos da JCL, otimizar o desempenho em recursos simultâneos e de alto volume e adicionar recursos. multi-data-source Algumas das principais alterações desta versão são:

- Aprimoramento do tratamento dinâmico de arquivos da JCL, expansão das instruções atuais e gerenciamento de conjuntos de dados concatenados, execução de várias instruções em um único bloco e transferência de dados de lotes para programas.

- Suporte aprimorado de vários comandos do CICS, incluindo a consulta de vários tipos de recursos do CICS.
- A capacidade de ter bancos de dados diferentes ao usar o Blu Age Runtime Utilities, mais adequado para cenários em que os dados comerciais são distribuídos em várias fontes.

Para obter mais informações sobre as alterações incluídas nesta versão, consulte as seguintes seções:

## Runtime versão 3.6.0

Tópicos



### zOS

Novos atributos

- JCL - DynamicFileBuilder - Gerenciamento aprimorado de manipuladores de arquivos (V7-9408)
- Conversão de formato aprimorada em algumas funções integradas do SQL DB2 ao chamar o utilitário INFUTILB UNLOAD (V7-9554)
- Atribuições de matriz multidimensional PLI aprimoradas (V7-9592)
- Tratamento do redirecionamento do sysout para o arquivo (V7-9992)

Melhorias

- Adicionar acionamento de procedimentos armazenados para DB2 RDBMS (V7-9155)
- SORT manipula a conversão para o formato PDF (V7-9286)
- JCL/GROOVY - Melhore a instrução REPRO para suportar conjuntos de dados DUMMY (V7-9424)
- Melhore o suporte ao CICS UNLOCK (V7-9606)
- Manipule o tamanho do valor padrão para Union (V7-9648)
- O JCL/GROOVY manipula diferentes terminações/disposições em conjuntos de dados concatenados (V7-9653)

- Torne o PageSize configurável para conjuntos de dados blusam (V7-9680)
- DSNUTIL - permite o carregamento de 24:00:00 como HORA válida no DB2LUW (V7-9697)
- Support a comparação de HIGH-VALUES (0xff) em NumberUtils .ne () NumberUtils /.eq () (V7-9731)
- JCL/GROOVY - suporte DO... Palavras-chave THEN nas cláusulas IF-THEN-ELSE do IDCAMS para executar várias instruções em um único bloco (V7-9750)
- JHDB inválido chamado programa fora do JHDB (V7-9782BatchRunner )
- Support caracteres de espaço em branco no cartão de controle SORT OUTFIL (V7-9808)
- Melhore o suporte do CICS READ PREV (V7-9845)
- Melhore o acesso simultâneo aos índices do conjunto de dados (V7-9864)
- Melhore o suporte ao CICS REWRITE (V7-9873)
- COBOL - suporte para SYSIN multilinha em instruções ACCEPT para passar dados do lote (JCL) para um programa (COBOL) (V7-9875)
- Groovy - Melhor manuseio da etapa de criação ConcatenatedFileConfiguration de arquivos (V7-9876)
- IDCAMS UTILITY - Tratamento da instrução DEFINE PATH (V7-9878)
- SORT BUILD - Ajuste a opção TRAN e manipule espaços em branco implícitos (V7-9925)
- Melhore o CICS DELETE com suporte à opção GENERIC (V7-9939)
- Melhore o suporte ao CICS STARTBR e ENDBR (V7-9952)
- Melhore o desempenho próximo no acesso simultâneo (V7-9953)
- Melhore o tratamento do status do arquivo na inicialização (V7-9991)
- Groovy - Permitir a chamada de getDisposition ()/()/getNormalTermination() em (getAbnormalTerminationV7-10012) ConcatenatedFileConfiguration

## COMO 400

Novos atributos

- Support indicadores externos em palavras-chave COMMIT (V7-6035)
- Redefinir o loop ReadC após a gravação SFLCTL (V7-8061)
- Support o indicador LR em CALL (V7-9250)
- Adicione um novo tipo de campo dinâmico (dividido) para lidar com o campo de entrada em várias linhas (V7-9370)

- Support arquivo primário/secundário (V7-9390)
- As áreas de dados locais agora são passadas para o trabalho chamado ao enviar um trabalho (V7-9775)
- Suporte do QTEMP para área de dados e suporte à criação de valor da área de dados. (V7-9916)
- Controle de compromisso - suporte para ativar/desativar o controle de compromisso (V7-9956)
- Support indicadores externos em palavras-chave COMMIT

Melhorias

- Melhore a exibição do valor 0 e o EDTWRD (V7-8933)
- Support da palavra-chave DSPF “CHKMSGID” (V7-9125)
- Transação de confirmação de SQL após o encerramento do lote (V7-9232)
- Melhore o suporte das palavras-chave EXPORT e IMPORT para campo e estrutura de dados (V7-9265)
- Support em letras minúsculas DateHelper (V7-9461)
- Support a conversão de*CYMD para *ISO (numérico) (V7-9488)
- Melhore o identificador do %len embutido para um campo variável (lado esquerdo e direito de uma expressão) (V7-9733)
- Melhore o suporte para funções integradas '%LOOKUPXX' XX (“LE”, "LT”, "GE”, "GT”) (V7-10064)

## Recursos transversais

Novos atributos

- CICS - Melhore a transação do Inquire para o status da opção (V7-9712)
- JCL - Melhore a carga do sysprint com o arquivo de saída do sistema (V7-9797)
- CICS - Melhore o INQUIRE TSQUEUE (V7-9823)
- CICS - Melhore o terminal Inquire para a opção ID de usuário (V7-9906)

Melhorias

- Melhore o controle da comparação com o espaço em branco (V7-8047)
- Melhore o registro para Jics e BluSam (V7-8847)

- Support BMS, atributos estendidos SOSI e símbolo programado F8 para campos dinâmicos (V7-8857)
- Lidar com estouro de buffer no parâmetro do programa (V7-9138)
- Melhore a simultaneidade de gravação de threads para o registro de bloqueios Blusam (V7-9505)
- Support a configuração de várias fontes de dados para Utility-PGM (V7-9570)
- Modo somente de bloqueio de nível de registro Blusam (V7-9626)
- Garanta que a persistência dos metadados resista à reinicialização do servidor (V7-9748)
- Melhore a limpeza do DAO em caso de exceção (fechamento do navegador) (V7-9790)
- Support DummyFile para INFUTILB SYSPUNCH (V7-9799)
- Aprimorar o suporte para valores negativos em NumericEditedType (V7-9935)

# Ferramentas de modernização versão 3.6.0

Tópicos



## zOS

Novos atributos

- JCL - Melhore o registro para o final do procedimento (V7-8509)
- PL1 - Melhore a geração de bolsas para o tipo de dados PakedLong (V7-8917)
- JCL - Melhore o registro para o final do procedimento quando o arquivo contém o marcador “final”// (V7-9509)
- PL1 - Melhore o suporte para GET EDIT com fluxo de ponto fixo e SYSIN (V7-9593)
- DB2 - Melhore o suporte para o tipo VARGRAPHIC DB2 (V7-9809)
- CICS - Melhore o comando QUERY SECURITY para a opção LOGMESSAGE (V7-9969)
- PL1 - Melhore a geração de bolsas para carga/gráfico embutido (V7-9989)

Melhorias

- PL1- Melhore o suporte para a palavra-chave INCLUDEX (V7-9588)
- PL/I - Trate a palavra-chave CHARGRAPHIC como um parâmetro válido de qualquer chamada de método (V7-9589)
- Melhorando a resolução da variável host PL1 quando nomeada com caracteres específicos @ # $ §. (V7-9654)
- COBOL - Support das palavras-chave C01... C12 e S01... S05 como parâmetro da instrução WRITE ADVANCING na etapa de análise (V7-9669)

## COMO 400

### Novos atributos

- Support a transformação SQL-DDS no Analyzer (V7-7687)
- Automatize a detecção de arquivos SQL-DDS (V7-7687)
- Implementação do pré-processamento SQL-DDS (V7-7687)
- Suporta a palavra-chave ALIGN (V7-9254)
- Support ExtName para DSPF e matriz multi-dim (V7-9663)
- InvalidKey Declarações de suporte sobre COBOL WRITE (V7-9793)

### Melhorias

- Melhoria no opcode TESTB (V7-8865)
- Melhore o suporte do DECFMT em foco (V7-8933)
- Manipulação do indicador resultante no MOVE (V7-9224)
- Melhore o suporte da palavra-chave TEMPLATE para campo e estrutura de dados (V7-9278)
- Melhoria do LIKEDS (DS definido usando LIKEDS é automaticamente qualificado) (V7-9302)
- COBOL - Melhore a geração da estrutura de indicadores (V7-9423)
- O parâmetro const no protótipo não é somente para leitura (V7-9437)
- Melhore a palavra-chave EDTCDE com o código de edição “Y” (V7-9443)
- Support a geração do campo*ROUTINE em PSDS e INFDS (V7-9487)
- Melhore o campo de regravação XXX para autônomo (o valor padrão é perdido durante a regravação) (V7-9522)

- Melhore o suporte de palavras-chave DSPF (V7-9658)
- Manipulando o valor padrão de ZEROES no binário (V7-9666)
- Support: ponteiro implícito (V7-9719)
- Melhore o tratamento da chamada embutida %size com um parâmetro (V7-9730)
- Melhore o tratamento de referências de estrutura de dados em chamadas integradas (%ELEM) (V7-9736)
- Melhore o tratamento do comprimento do sinal para o campo com a referência LIKE na especificação de definição (V7-9738)
- Melhoria no REWRITE (V7-9791)
- Melhoria da geração de índices a partir de arquivos DDS (V7-9803)
- Melhore a robustez dos mapeadores com valor numérico inválido (V7-9813)
- Melhore a geração de arquivos SQLModel e AllIndexes (V7-9818)
- Melhore o suporte qualificado do DS (V7-9863)
- Melhore o suporte do LOOKUP (com um campo autônomo COMO um DS no parâmetro) (V7-9961)
- Melhore o LIKE no indicador (V7-9985)
- Manipulando o indicador resultante no MVR (V7-9995)
- Support o personagem N com tilde (V7-10021)
- Melhore a geração moderna de arquivos DDL a partir de arquivos legados SQLDDS (V7-10067)

## Recursos transversais

### Novos atributos

- Personalize a localização do recurso com uma propriedade yml (D88816105)
- COBOL - Support da instrução EXIT PERFORM para sair de um PERFORM embutido sem usar um GO TO/PERFORM... POR MEIO DE (V7-9582)
- Especificar a codificação legada padrão a ser considerada nos metadados globais. (V7-9883)

### Melhorias

- Melhore a geração de máscaras (V7-9602)

- Melhore o aquecimento do contexto (V7-9621)
- Torne a rosca Charset CUSTOM930 segura. (V7-9674)
- Melhoria no MOVEA (V7-9773)

## Notas de lançamento 3.5.0

Esta versão do AWS Blu Age Runtime and Modernization Tools fornece novos recursos para migrações legadas do zOS e do AS400, principalmente orientadas à otimização de conjuntos de dados e mensagens, bem como recursos Java estendidos como um ativo resultante do processo de transformação. Algumas das principais alterações desta versão são:

- Capacidade de migrar programas de CL para Java, além do recurso preexistente de scripts groovy, para facilitar sua integração com outros programas modernizados e para simplificar a curva de aprendizado do cliente unificando a linguagem de programação resultante.
- Redução do tempo e otimização do desempenho das cargas de conjuntos de dados no Redis com o novo recurso de massa de dados.
- Capacidade de operar e transmitir conjuntos de dados dentro das etapas do trabalho para modernizar os comportamentos tradicionais dos conjuntos de dados.
- Extensão da migração de SQL para suportar arquivos de entrada VB e migração simplificada do Java 11.
- Vários novos mecanismos para uma integração mais rápida com o IBM MQ, incluindo cabeçalhos adicionais, suporte estendido a GET/PUT e recuperação automática de metadados da fila.
- Endpoint REST para metadados de conjuntos de dados e importar conjuntos de dados de buckets do S3.

Para obter mais informações sobre as alterações incluídas nesta versão, consulte as seguintes seções:

## Runtime versão 3.5.0

Tópicos

## zOS

### Novos atributos

- JCL SORT - Lidar com a nova sobreposição de palavras-chave (V7-9409)
- ZOS COBOL - melhore o suporte de caracteres flutuantes (V7-9404)
- Porta do RedisJics TSQueue para RedisTemplate & ListOperations (V7-9212)
- ZOS JCL - aprimora o caminho do diretório temporário com o diretório de arquivos, se definido por meio de UserDefinedParameters (V7-9012)
- Manipule a FUNÇÃO ORD-MAX com ALL (todos os itens da matriz) (V7-9366)
- Chaves prefixadas e legíveis por humanos agora são usadas ao armazenar filas TS no Redis (V7-9212)
- Adicionar o endpoint “obter conjunto de dados” para a API do blusam
- JCL - ADICIONE suporte para trabalho em lote com nome envolvendo caracteres especiais como # (V7-9136)
- A busca do TSModel agora é executada de forma robusta sob demanda (V7-9212)

### Melhorias

- Suporte INCLUDE não versionado em arquivos LNK (V7-6022)
- MQ - Suporte aprimorado de codificação (V7-9652)
- Melhorando o suporte para bytes duplos ou conjuntos de caracteres mistos para vários tipos de caracteres (V7-9596)
- JCL - Support of FilesDirectory configurado em IDCAMS delete NONVSAM (V7-9609)
- Support o modo em massa para carregamento de conjuntos de dados ESDS e RRDS de arquivos (V7-8639)
- Manipule a abertura de ESDS vazios no modo de entrada. (V7-9287)
- Melhore a instrução DEFINE CLUSTER com suporte à abreviatura ORD/UNORD (V7-9451)
- BluSam Melhorias no desempenho do Redis lock (V7-8639)
- Melhore a instrução DEFINE CLUSTER para suportar RECORDSIZE fornecida no escopo do argumento DATA () (V7-9337)
- Adiciona suporte aos atributos BUFFERSPACE/UNIQUE nas instruções DEFINE CLUSTER (V7-9419)

- Melhore a operação de BluSam leitura para um conjunto de dados de registro de comprimento variável. (V7-9391)
- O ENDEREÇO CICS representa corretamente o CWA ausente como nulo (V7-9491)
- Remova a gravação desnecessária nos bloqueios finais (V7-8639)
- Manipular a injeção de modelo de cache Redis no cache (V7-9510)
- Decodifique corretamente o parâmetro BPXWDYN (V7-9417)
- Melhoria no consumo de exportação do LISTCAT (V7-9201)
- Suporte a caracteres não imprimíveis no nome BluSam TS Queues (V7-9212)
- Manipule a criação de mapa de recebimento para campo com mapset null (V7-9486)
- Melhore a operação de BluesamRelativeFile exclusão e regravação para o modo de acesso dinâmico. (V7-8989)

## COMO 400

### Novos atributos

- Adicione um recurso para gerar arquivos CL como programas Java por meio do pivô DS/STM padrão (V7-9427)
- Support Input File com o modo ADD (V7-9378)
- Melhorou a ordem de classificação e o gerenciamento de recuperação para suportar o comando cl OPNQRYF (Open Query File) e adicionou suporte ao parâmetro SHARE em. OverrideItem (V7-9364)

### Melhorias

- Support SFLNXTCHG em (V7-8061) UpdateSubfile
- Modifique o escopo do contexto CL ao executar o comando CL (V7-9624)
- Manipule o código de retorno do programa BPXWDYN (V7-9417)
- Limpe os monitores locais. (V7-9624)
- Support da palavra-chave DSPF RTNCSRLOC (V7-9389)
- setOnGreaterOrEqual() não configurando Igual a 1 (V7-9342)
- Atualizar o cache de campos ativado UpdateSubfileRecord (V7-9376)
- Melhore o suporte SFLNXTCHG (V7-8061)

## Recursos transversais

### Novos atributos

- Ignore o prefixo G na sequência gráfica literal. (V7-9420)
- ZOS COBOL - Melhore o suporte de Fiedl.initialize () para algumas estruturas especiais (V7-9485)
- Permitir a inicialização do contexto de forma assíncrona para melhorar o desempenho da inicialização do programa (V7-9446)
- SQL Release explicitamente a instrução de preparação aberta e. ResulSet (V7-9422)
- Melhore o JMS MQ - suporte MQRFH2 para MQ PUT /V7-7085 - suporte ao gerenciador de filas padrão (V7-9400)
- Gerenciamento de SQL: habilite conversões do Lambda em parâmetros para comandos SET (V7-9492)
- ZOS MQ JMS - Adicione suporte ao MQCOMIT e ao MQBACK (V7-9399)
- ZOS IBMQ - Melhore o suporte ao MQINQ (V7-9544)
- Manipule a operação CONCAT com byte em vez de string ao usar a codificação de byte duplo. (V7-8932)
- ZOS IBMMQ - Melhore o suporte ao comando PUT com as opções SET_ALL_CONTEXT (V7-9544)

### Melhorias

- Manipule nomes de arquivos gdg com o caractere $ (V7-9066)
- O Diagnóstico SQL retorna 1 como cláusula NUMBER quando a instrução SQL anterior é bem-sucedida. (V7-9410)
- Esboço para campo com comprimento não nulo (V7-7536)
- Support a função PL1 GRAPHIC integrada (V7-9245)
- MQ - Adicionar suporte da versão para configuração de campos MQGMO (V7-9500)
- JMS MQ GET - Melhoria do comprimento de dados da mensagem retornada (V7-9502)
- Defina sqlerrd (3) com o número de itens buscados no contexto ROWSET. (V7-9371)

# Ferramentas de modernização versão 3.5.0

### Tópicos

- zOS
- COMO 400
- Recursos transversais

## zOS

### Novos atributos

- ZOS PLI - Support asterisk index na atribuição com expressão binária (V7-9178)
- JCL para BatchScript - Um “//” marca o fim da execução do trabalho (V7-9304)
- ZOS PLI - suporte aprimorado para caracteres flutuantes e login em tipo numérico editado (V7-8982)
- COBOL - Support da função SUM integrada (V7-9367)
- JCL- opcionalmente, comente o código morto após a declaração nula (//) (V7-9202)
- JCL- Support of operator '|' na declaração de condição (V7-9499)
- PL/I - Comentário das diretivas de pré-compilação na etapa de pré-processamento para evitar exceções de análise (V7-9507)

### Melhorias

- Manipule a definição de fluxo com delimitador (V7-9615)
- Melhorando o tratamento das exportações do LISTCAT. (V7-9201)
- PL/I - Aprimoramento para suportar argumentos 'nulos' implícitos (V7-9204)

## COMO 400

### Novos atributos

- Support da palavra-chave DDS CONCAT (V7-9439)
- Refatore o código java gerado para palavras-chave DSPF. (V7-7700)
- Support Variating keyword em campos dentro de uma definição de estrutura de dados (V7-9029)

### Melhorias

- Melhore a análise do relacionamento lógico E/OU (V7-9352)

- COBOL Melhore o mapeamento entre vo e dsEntity (V7-9449)
- Exibir valor vazio se a entrada numérica estiver focada (V7-9374)
- Variável local no SQL Declare Cursor (V7-9456)
- Problema de escopo com DS vazio (V7-9466)
- Truncar linhas após a coluna 80 antes da análise (V7-9632)
- Melhore o tratamento de referências de campo e chamadas integradas em palavras-chave (DIM, LIKE,...) na especificação de definição (V7-9358)
- Support a comentários SQL (--) (V7-9632)
- FullFree análise, digite Data/Hora/Timestamp (V7-9542)
- Incluir SQLCA da FullFree análise (V7-9333)
- Melhore o Support of Control Level. (V7-9610)
- Lide com a comparação de DS com *BLANKS (V7-9668)
- Melhore o suporte de vários indicadores no DDS (V7-9318)
- Melhoria do suporte de vários programas DSPF (V7-9657)
- Melhore a manipulação do campo com LIKE (caso de estrutura de dados curtida e caso de estrutura de dados curtida em uma matriz) (V7-9213)
- RPG grátis, continuação de Handle no literal (V7-9686)
- Improve Support of end of program records (V7-9452)
- Support da frase LINKAGE na declaração CALL. (V7-9685)
- Código de operação CASXX (CASBB sem grupo CASXX) (V7-9357)
- Melhore a análise FullFree de RPG (V7-9457)
- %LEN integrado não suporta DS como argumento (V7-9267)
- Melhorias do MOVEA quando o fator 2 é *ALL'X... ' (V7-9228)
- Support assign com o campo RENAME (V7-9385)

## Recursos transversais

### Novos atributos

- Ferramenta SQL Migrator: adicione a opção OID para tamanho de registro variável na etapa de carregamento ebcdic. (V7-9380)

- Ferramenta SQL Migrator: suporte para Java 11 na opção OID (V7-9599)

Melhorias

- Melhore o suporte para matrizes aninhadas (V7-9595)
- Substitua o caractere Â¬ por! no caso de Â¬ é suportado pela codificação original. (V7-9465)
- JCL - Support of PASS normal termination para compartilhar conjuntos de dados entre as etapas do trabalho (V7-9504)
- Aplique ON NULL à definição de coluna no ORACLE ao lidar com VARCHAR e tipo de coluna db anulável. (V7-9681)
- Melhore a conformidade com a injeção de molas (V7-9635)

# AWS Conceitos de tempo de execução do Blu Age

Compreender os conceitos básicos do AWS Blu Age Runtime pode ajudá-lo a entender como seus aplicativos são modernizados com a refatoração automatizada.

Tópicos



## AWS Arquitetura de alto nível do Blu Age Runtime

Como parte da solução AWS Blu Age para modernizar programas legados para Java, o AWS Blu Age Runtime fornece um ponto de entrada unificado baseado em REST para aplicativos modernizados e uma estrutura de execução para esses aplicativos, por meio de bibliotecas que fornecem construções legadas e uma padronização da organização do código dos programas.

Esses aplicativos modernizados são o resultado do processo AWS Blu Age Automated Refactor para modernizar programas de mainframe e midrange (referidos no documento a seguir como “legados”) para uma arquitetura baseada na web.

As metas do AWS Blu Age Runtime são a reprodução do comportamento de programas legados (isofuncionalidade), desempenhos (com relação ao tempo de execução dos programas e ao consumo de recursos) e a facilidade de manutenção de programas modernizados por

desenvolvedores Java, por meio do uso de ambientes e expressões idiomáticas familiares, como tomcat, Spring, getters/setters, APIs fluentes...

Tópicos



## AWS Componentes de tempo de execução do Blu Age

O ambiente AWS Blu Age Runtime é composto por dois tipos de componentes:

- Um conjunto de bibliotecas java (arquivos jar) geralmente referenciado como “a pasta compartilhada” e que fornece construções e declarações legadas.
- Um conjunto de aplicativos web (arquivos war) contendo aplicativos web baseados em Spring que fornecem um conjunto comum de estruturas e serviços para programas modernizados.

As seções a seguir detalham a função de ambos os componentes.

### AWS Bibliotecas Blu Age

As bibliotecas do AWS Blu Age são um conjunto de arquivos jar armazenados em uma `shared/` subpasta adicionada ao classpath padrão do tomcat, a fim de disponibilizá-los para todos os programas Java modernizados. O objetivo deles é fornecer recursos que não estejam nem de forma nativa nem facilmente disponíveis no ambiente de programação Java, mas que sejam típicos de ambientes de desenvolvimento legados. Esses recursos são expostos de uma forma que seja o mais familiar possível para os desenvolvedores Java (getters/setters, APIs fluentes e baseadas em classes). Um exemplo importante é a biblioteca Data Simplifier, que fornece construções antigas de layout e manipulação de memória (encontradas nas linguagens COBOL, PL1 ou RPG) para programas Java. Esses jars são uma dependência central do código Java modernizado gerado a partir de programas legados. Para obter mais informações sobre o Data Simplifier, consulte Simplificador de dados.

### Aplicativo Web

Os Arquivos de Aplicativos Web (WARs) são uma forma padrão de implantar código e aplicativos no servidor de aplicativos tomcat. Os fornecidos como parte do tempo de execução do AWS Blu

Age visam fornecer um conjunto de estruturas de execução que reproduzem ambientes legados e monitores de transações (lotes JCL, CICS, IMS...) e os serviços necessários associados.

O mais importante é `gapwalk-application` (geralmente abreviado como “Gapwalk”), que fornece um conjunto unificado de pontos de entrada baseados em REST para acionar e controlar transações, programas e execução de lotes. Para ter mais informações, consulte AWS APIs de tempo de execução do Blu Age.

Esse aplicativo web aloca threads e recursos de execução Java para executar programas modernizados no contexto para o qual foram projetados. Exemplos desses ambientes reproduzidos são detalhados na seção a seguir.

Outros aplicativos da Web adicionam ao ambiente de execução (mais precisamente, ao “Registro de Programas” descrito abaixo) programas que emulam aqueles disponíveis e que podem ser chamados pelos programas legados. Duas categorias importantes são:

- Emulação de programas fornecidos pelo sistema operacional: os lotes controlados pela JCL esperam poder chamar uma variedade de programas de manipulação de arquivos e bancos de dados como parte de seu ambiente padrão. Exemplos incluem `SORT`/`DFSORT` ou `IDCAMS`. Para isso, são fornecidos programas Java que reproduzem esse comportamento e podem ser chamados usando as mesmas convenções dos antigos.
- “Drivers”, que são programas especializados fornecidos pela estrutura de execução ou pelo middleware como pontos de entrada. Um exemplo é `CBLTDLI` de quais programas COBOL executados no ambiente IMS dependem para acessar os serviços relacionados ao IMS (IMS DB, diálogo do usuário por meio do MFS etc.).

#### Registro de programas

Para participar e tirar proveito dessas construções, estruturas e serviços, os programas Java modernizados a partir dos antigos aderem a uma estrutura específica documentada em AWS Estrutura Blu Age de um aplicativo modernizado. Na inicialização, o AWS Blu Age Runtime coletará todos esses programas em um “Registro de Programas” comum para que possam ser invocados (e chamados uns aos outros) posteriormente. O Registro de Programas oferece acoplamento frouxo e possibilidades de decomposição (já que os programas que se chamam não precisam ser modernizados simultaneamente).

### Ambientes de execução

Ambientes e coreografias legados frequentemente encontrados estão disponíveis:

- Os lotes controlados pela JCL, uma vez modernizados para programas Java e scripts Groovy, podem ser iniciados de forma síncrona (bloqueio) ou assíncrona (desanexada). No último caso, sua execução pode ser monitorada por meio de endpoints REST.
- Um subsistema AWS Blu Age fornece um ambiente de execução semelhante ao CICS por meio de:
  - um ponto de entrada usado para iniciar uma transação do CICS e executar programas associados, respeitando a coreografia dos “níveis de execução” do CICS,
  - um armazenamento externo para definições de recursos,
  - um conjunto homogêneo de APIs fluentes em Java que reproduzem declarações, `EXEC CICS`
  - um conjunto de classes conectáveis que reproduzem serviços do CICS, como filas de armazenamento temporário, filas de dados temporárias ou acesso a arquivos (várias implementações geralmente estão disponíveis, como Amazon Managed Service para Apache Flink, Amazon Simple Queue Service ou RabbitMQ para filas TD),
  - para aplicativos voltados para o usuário, o formato de descrição de tela do BMS é modernizado para um aplicativo web Angular e a caixa de diálogo “pseudo-conversacional” correspondente é suportada.
- Da mesma forma, outro subsistema fornece coreografia baseada em mensagens IMS e oferece suporte à modernização de telas de interface do usuário no formato MFS.
- Além disso, um terceiro subsistema permite a execução de programas em um ambiente semelhante ao iSeries, incluindo a modernização de telas especificadas pelo DSPF (Display File).

Todos esses ambientes se baseiam em serviços comuns em nível de sistema operacional, como:

- a emulação da alocação e layout de memória legados (Simplificador de dados),
- Reprodução baseada em threads em Java da execução de “unidades de execução” do COBOL e do mecanismo de passagem de parâmetros (`CALL` instrução),
- emulação de organizações planas e concatenadas de VSAM (por meio do conjunto de bibliotecas Bluesam) e GDG Data Set,
- acesso a armazenamentos de dados, como RDBMS (`EXEC SQL`declarações).

## Apatridia e tratamento de sessões

Um recurso importante do AWS Blu Age Runtime é permitir cenários de alta disponibilidade (HA) e escalabilidade horizontal ao executar programas modernizados.

A base para isso é a apátrida, um exemplo importante disso é o tratamento de sessões HTTP.

Manuseio de sessões

Sendo o Tomcat baseado na web, um mecanismo importante para isso é o tratamento da sessão HTTP (conforme fornecido pelo tomcat e pelo Spring) e o design sem estado. Como tal, o design de apatridia é baseado no seguinte:

- os usuários se conectam por meio de HTTPS,
- os servidores de aplicativos são implantados por trás de um balanceador de carga,
- quando um usuário se conecta pela primeira vez ao aplicativo, ele será autenticado e o servidor do aplicativo criará um identificador (normalmente dentro de um cookie)
- esse identificador será usado como uma chave para salvar e recuperar o contexto do usuário de/para um cache externo (armazenamento de dados).

O gerenciamento de cookies é feito automaticamente pela estrutura AWS Blu Age e pelo servidor tomcat subjacente, isso é transparente para o usuário. O navegador da Internet do usuário gerenciará isso automaticamente.

O aplicativo web Gapwalk pode armazenar o estado da sessão (o contexto) em vários armazenamentos de dados:

- Amazon ElastiCache para Redis
- Cluster do Redis
- no mapa de memória (somente para ambientes autônomos e de desenvolvimento, não adequado para HA).

## Alta disponibilidade e apátrida

De forma mais geral, um princípio de design da estrutura AWS Blu Age é a apatridia: a maioria dos estados não transitórios necessários para reproduzir o comportamento de programas legados não são armazenados nos servidores de aplicativos, mas compartilhados por meio de uma “fonte única de verdade” externa comum.

Exemplos desses estados são as filas de armazenamento temporário ou as definições de recursos do CICS, e os armazenamentos externos típicos deles são servidores ou bancos de dados relacionais compatíveis com Redis.

Esse design, combinado com balanceamento de carga e sessões compartilhadas, faz com que a maioria dos diálogos voltados para o usuário (OLTP, “Processamento Transacional Online”) seja distribuída entre vários “nós” (aqui, instâncias do Tomcat).

Na verdade, um usuário pode executar uma transação em qualquer servidor sem se importar se a próxima chamada de transação será realizada em um servidor diferente. Então, quando um novo servidor é gerado (devido ao escalonamento automático ou para substituir um servidor não íntegro), podemos garantir que qualquer servidor acessível e íntegro possa executar a transação conforme o esperado com os resultados adequados (valor retornado esperado, alteração esperada de dados no banco de dados, etc.).

## AWS Estrutura Blu Age de um aplicativo modernizado

Este documento fornece detalhes sobre a estrutura de aplicativos modernizados (usando ferramentas de refatoração de modernização de AWS mainframe), para que os desenvolvedores possam realizar várias tarefas, como:

- navegando pelos aplicativos sem problemas.
- desenvolvendo programas personalizados que podem ser chamados a partir dos aplicativos modernizados.
- refatorando com segurança aplicativos modernizados.

Presumimos que você já tenha conhecimentos básicos sobre o seguinte:

- conceitos de codificação comuns herdados, como registros, conjuntos de dados e seus modos de acesso aos registros — indexados, sequenciais —, VSAM, unidades de execução, scripts jcl, conceitos do CICS e assim por diante.
- codificação java usando o framework Spring.
- Em todo o documento, usamos `short class names` para facilitar a leitura. Para obter mais informações, consulte AWS Mapeamentos de nomes totalmente qualificados da Blu Age para recuperar os nomes totalmente qualificados correspondentes para os elementos de tempo de execução do AWS Blu Age e Mapeamentos de nomes totalmente qualificados de terceiros para recuperar os nomes totalmente qualificados correspondentes para elementos de terceiros.
- Todos os artefatos e amostras são retirados das saídas do processo de modernização do aplicativo COBOL/CICS de amostra. CardDemo

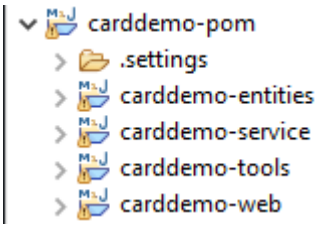
Tópicos

- Organização de artefatos
- Executando e chamando programas
- Escreva seu próprio programa
- Mapeamentos de nomes totalmente qualificados

## Organização de artefatos

AWS Os aplicativos modernizados do Blu Age são empacotados como aplicativos web java (.war), que você pode implantar em um servidor JEE. Normalmente, o servidor é uma instância do Tomcat que incorpora o tempo de execução do AWS Blu Age Velocity, que atualmente é construído com base nas estruturas Springboot e Angular (para a parte da interface do usuário).

A war agrega vários artefatos de componentes (.jar). Cada jar é o resultado da compilação (usando a ferramenta maven) de um projeto java dedicado cujos elementos são o resultado do processo de modernização.



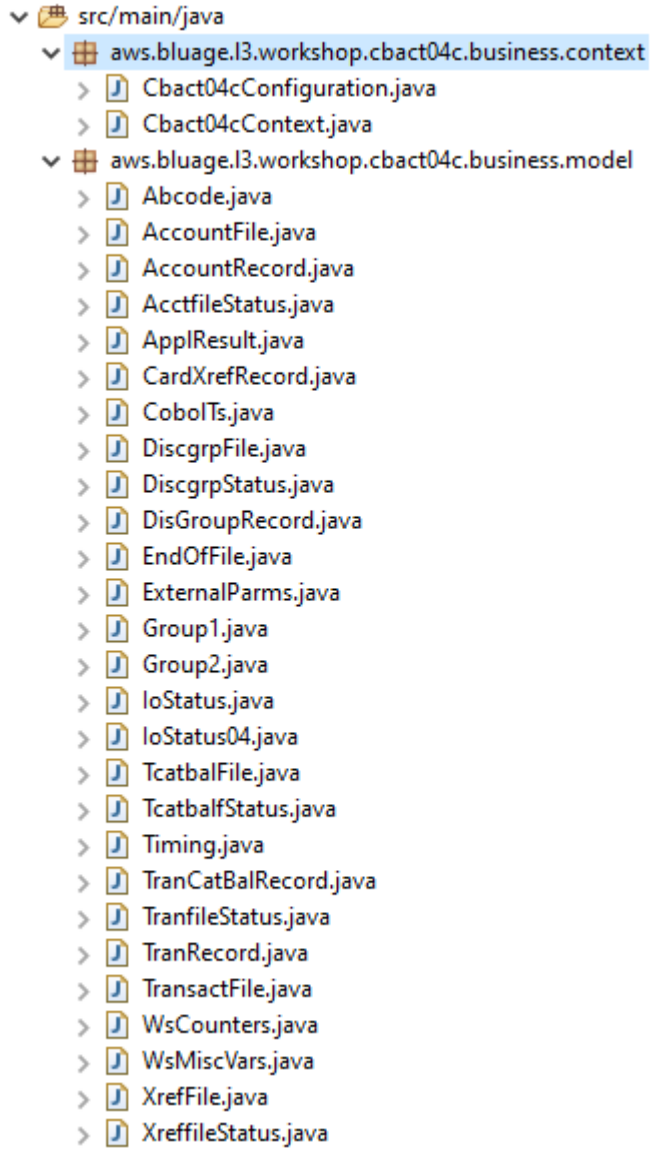

A organização básica se baseia na seguinte estrutura:

- Projeto de entidades: contém elementos do modelo de negócios e do contexto. O nome do projeto geralmente termina com “-entities”. Normalmente, para um determinado programa COBOL legado, isso corresponde à modernização da seção de E/S (conjuntos de dados) e da divisão de dados. É possível ter um projeto de mais de uma entidade.
- Projeto de serviço: contém elementos legados de modernização da lógica de negócios. Normalmente, a divisão de procedimentos de um programa COBOL. Você pode ter mais de um projeto de serviço.
- Projeto utilitário: contém ferramentas e utilitários comuns compartilhados, usados por outros projetos.
- Projeto Web: contém a modernização de elementos relacionados à interface do usuário, quando aplicável. Não é usado para projetos de modernização somente em lote. Esses elementos da interface do usuário podem vir dos mapas do CICS BMS, dos componentes do IMS MFS e de outras fontes de interface do usuário do mainframe. Você pode ter mais de um projeto da Web.

## Conteúdo do projeto Entidades

> **Note**
>
> As descrições a seguir se aplicam somente às saídas de modernização COBOL e PL/I. As saídas de modernização do RPG são baseadas em um layout diferente.

Antes de qualquer refatoração, a organização dos pacotes no projeto da entidade está vinculada aos programas modernizados. É possível fazer isso de duas maneiras diferentes. A forma preferida é usar a caixa de ferramentas de refatoração, que opera antes de você acionar o mecanismo de geração de código. Esta é uma operação avançada, que é explicada nos BluAge treinamentos. Para obter mais informações, consulte Workshop de refatoração. Essa abordagem permite que você preserve a capacidade de regenerar o código java posteriormente, para se beneficiar de novas melhorias no futuro, por exemplo). A outra maneira é fazer refatorações regulares de java, diretamente no código-fonte gerado, usando qualquer abordagem de refatoração de java que você queira aplicar -- por sua conta e risco.




Aulas relacionadas ao programa

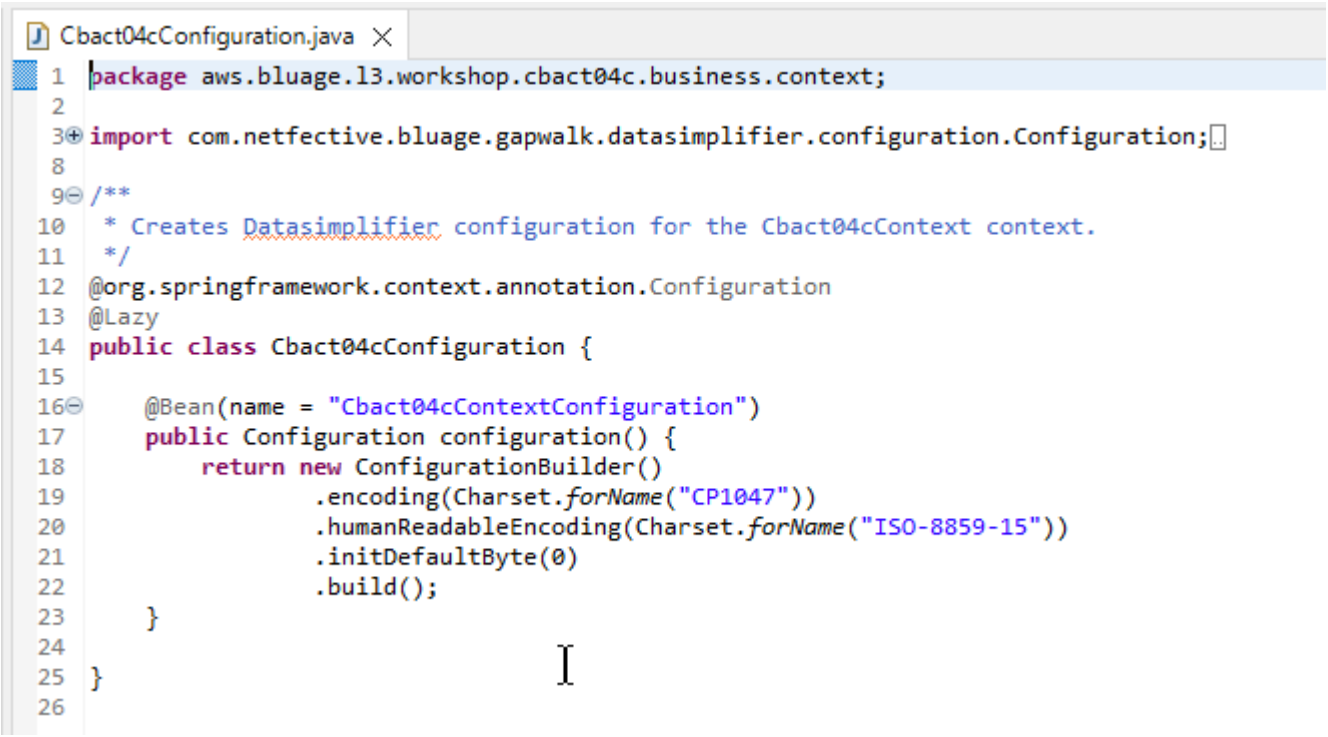
Cada programa modernizado está relacionado a dois pacotes, um pacote business.context e um pacote business.model.

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

  O subpacote business.context contém duas classes, uma classe de configuração e uma classe de contexto.

  - Uma classe de configuração para o programa, que contém detalhes de configuração específicos para um determinado programa, como o conjunto de caracteres a ser usado para representar elementos de dados baseados em caracteres, o valor de byte padrão para preencher elementos

da estrutura de dados e assim por diante. O nome da classe termina com “Configuração”. Ele é marcado com a `@org.springframework.context.annotation.Configuration` anotação e contém um único método que deve retornar um objeto `Configuration` configurado corretamente.



Cbact04cConfiguration.java

```
1  package aws.bluage.l3.workshop.cbact04c.business.context;
2
3  import com.netfective.bluage.gapwalk.datasimplifier.configuration.Configuration;
8
9  /**
10  * Creates Datasimplifier configuration for the Cbact04cContext context.
11  */
12 @org.springframework.context.annotation.Configuration
13 @Lazy
14 public class Cbact04cConfiguration {
15
16     @Bean(name = "Cbact04cContextConfiguration")
17     public Configuration configuration() {
18         return new ConfigurationBuilder()
19                 .encoding(Charset.forName("CP1047"))
20                 .humanReadableEncoding(Charset.forName("ISO-8859-15"))
21                 .initDefaultByte(0)
22                 .build();
23     }
24
25 }
26
```

- Uma classe de contexto, que serve como uma ponte entre as classes de serviço do programa (veja abaixo) e as estruturas de dados (`Record`) e os conjuntos de dados (`File`) do subpacote do modelo (veja abaixo). O nome da classe termina com “Contexto” e é uma subclasse da `RuntimeContext` classe.

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;
```

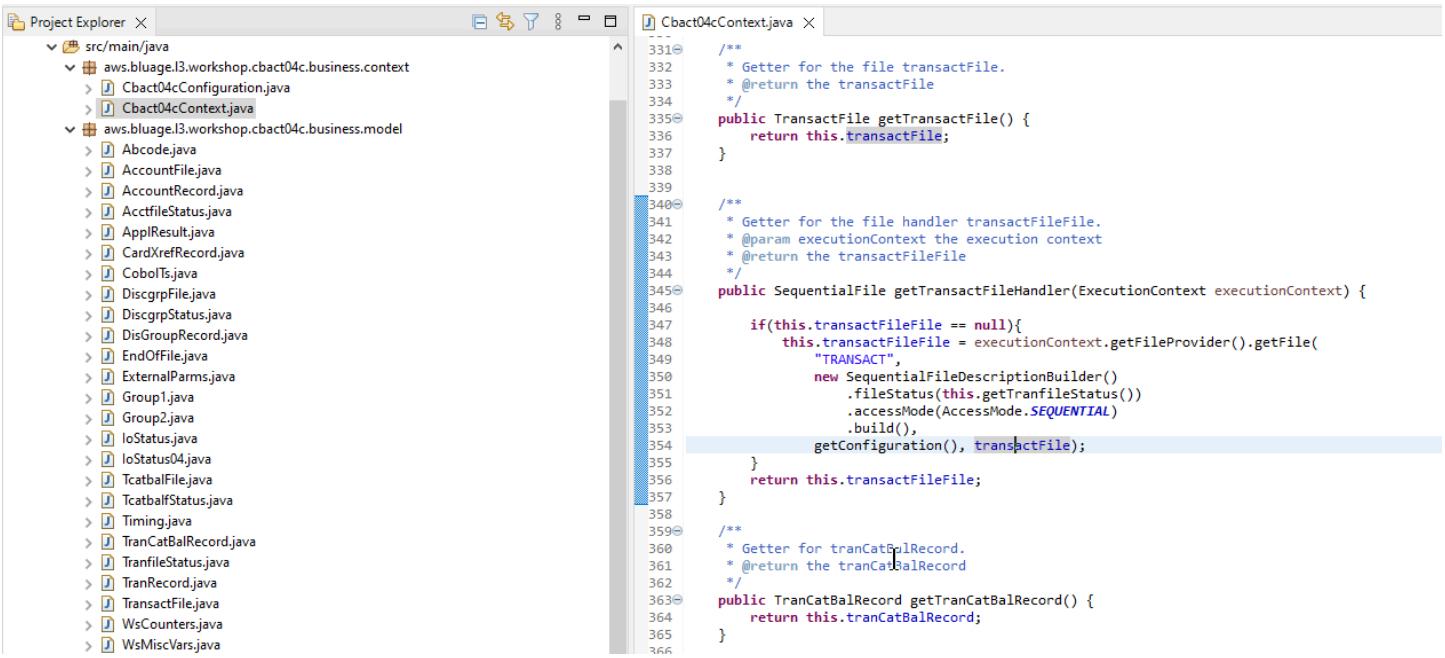
- *`base package`*`.`*`program`*`.business.model`

  O subpacote do modelo contém todas as estruturas de dados que o programa fornecido pode usar. Por exemplo, qualquer estrutura de dados COBOL de nível 01 corresponde a uma classe no subpacote do modelo (estruturas de dados de nível inferior são propriedades de sua própria estrutura de nível 01). Para obter mais informações sobre como modernizamos 01 estruturas de dados, consulte Simplificador de dados.

DiscgrpFile.java

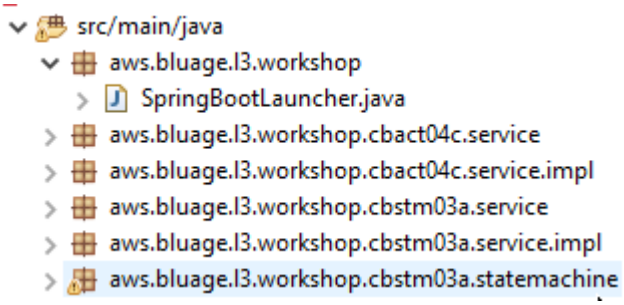
```java
package aws.bluage.l3.workshop.cbact04c.business.model;

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));
```

Todas as classes estendem a `RecordEntity` classe, que representa o acesso a uma representação de registros comerciais. Alguns dos registros têm um propósito especial, pois estão vinculados a um `File`. A vinculação entre a `Record` e a `File` é feita nos FileHandler métodos * correspondentes encontrados na classe de contexto ao criar o objeto de arquivo. Por exemplo, a lista a seguir mostra como o TransactfileFile `File` está vinculado ao TransactFile `Record` (do subpacote do modelo).




Conteúdo do projeto de serviço

Cada projeto de serviço vem com um aplicativo Springboot dedicado, que é usado como a espinha dorsal da arquitetura. Isso é materializado por meio da classe chamada `SpringBootLauncher`, localizada no pacote base das fontes java do serviço:



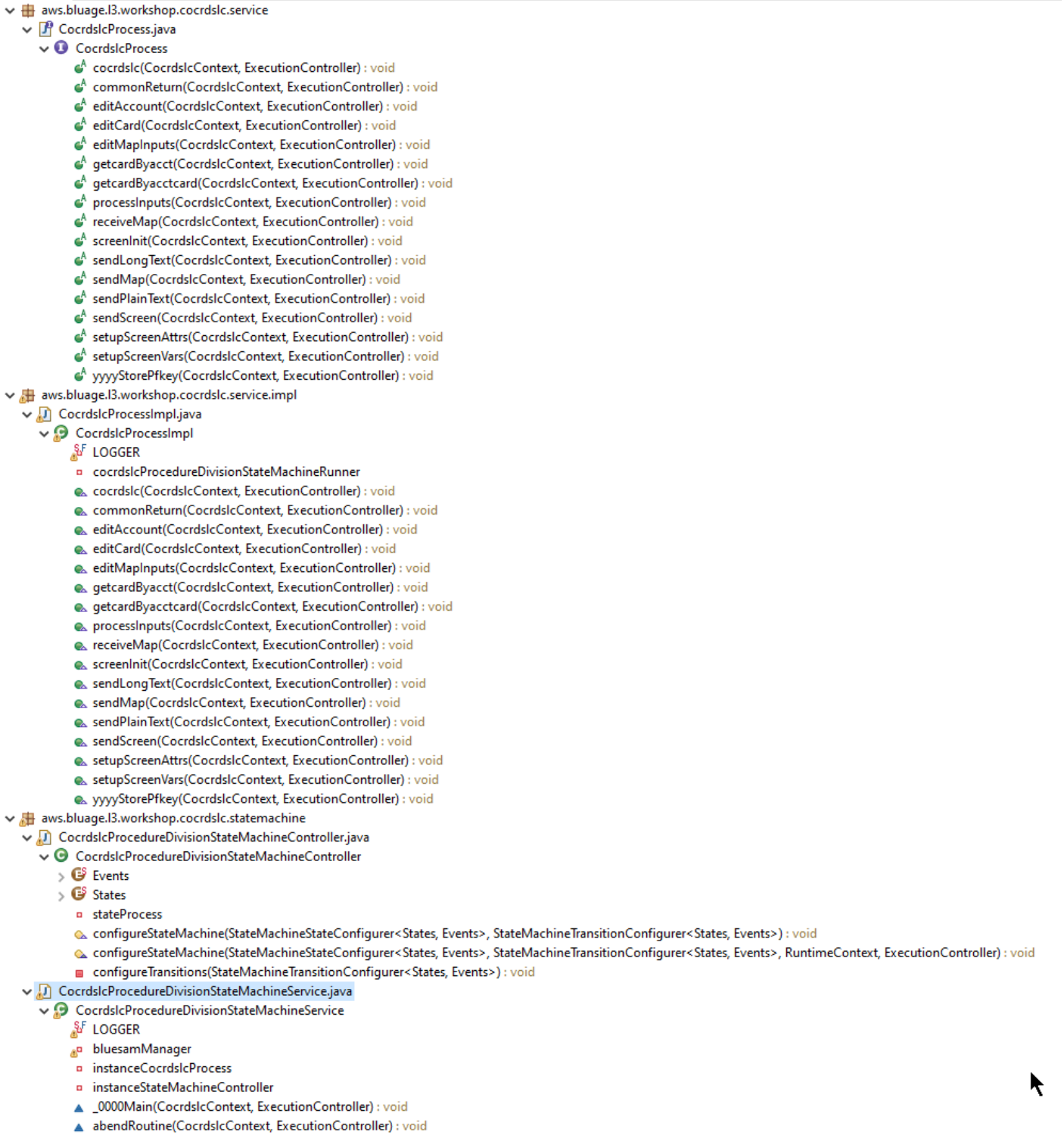

Essa classe é especialmente responsável por:

- criando a cola entre as classes do programa e os recursos gerenciados (fontes de dados/ gerenciadores de transações/mapeamentos de conjuntos de dados/ etc...).
- fornecendo dois `ConfigurableApplicationContext` programas.
- descobrindo todas as classes marcadas como componentes de primavera (`@Component`).
- garantindo que os programas sejam registrados corretamente no `ProgramRegistry` -- veja o método de inicialização responsável por esse registro.

```
/**
 * 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);
}
```

### Artefatos relacionados ao programa

Sem refatoração prévia, os resultados da modernização da lógica de negócios são organizados em dois ou três pacotes por programa antigo:




O estojo mais exaustivo terá três pacotes:

- *base package*`.`*program*`.service`: contém uma interface chamada Program Process, que tem métodos de negócios para lidar com a lógica de negócios, preservando o fluxo de controle de execução legado.
- *base package*`.`*program*`.service.impl`: contém uma classe chamada ProgramaProcessImpl, que é a implementação da interface de processo descrita anteriormente. É aqui que as declarações legadas são “traduzidas” para instruções java, com base na estrutura AWS Blu Age:

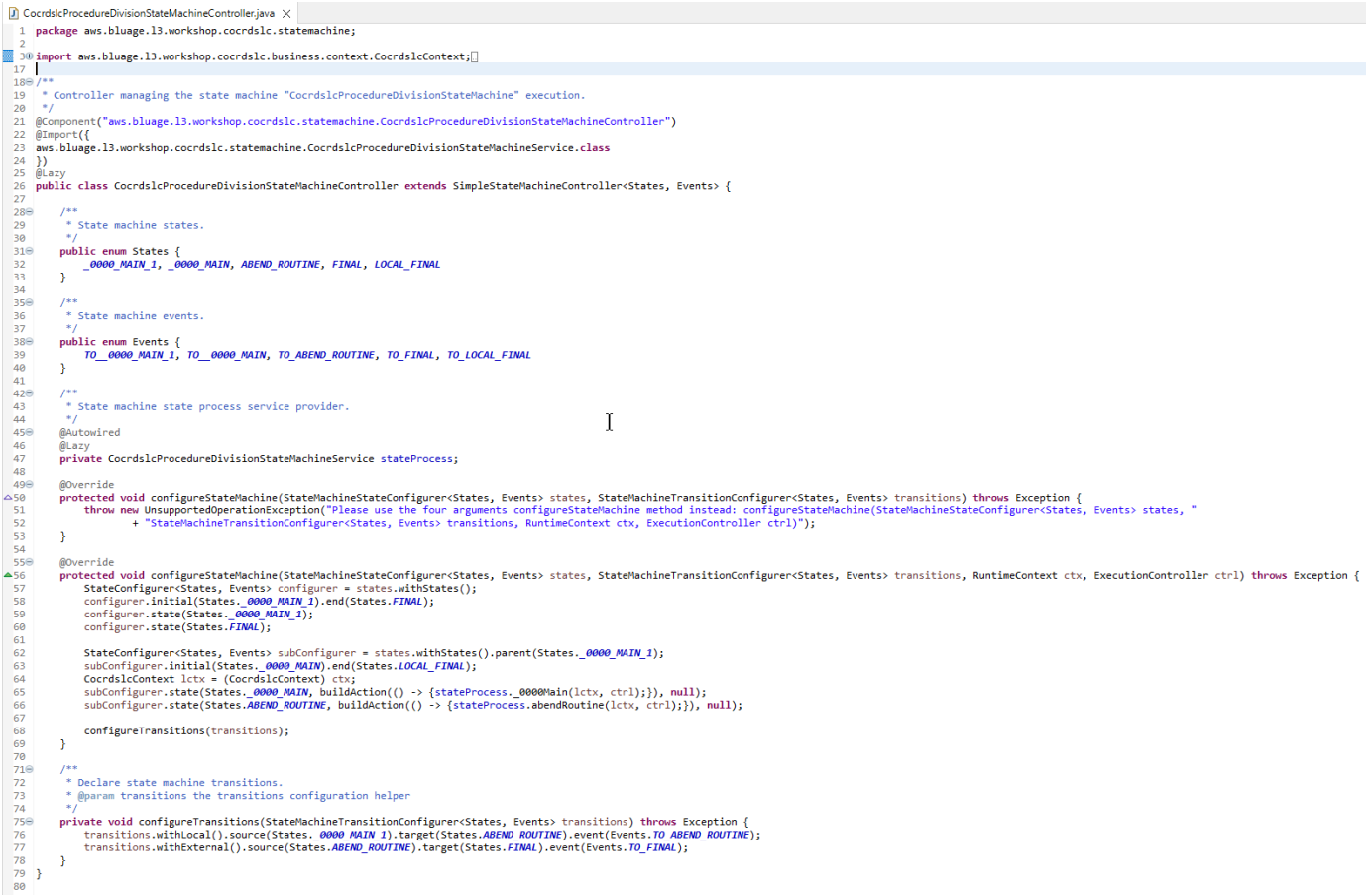
CocrdslcProcessImpl.java

```java
    /**
     * Process operation sendScreen.
     *
     * @param ctx
     * @param ctrl
     */
    @Override
    public void sendScreen(final CocrdslcContext ctx, final ExecutionController ctrl) {
        ctx.getCcWorkAreas().setCcardNextMapset(ctx.getWsLiterals().getLitThismapset());
        ctx.getCcWorkAreas().setCcardNextMap(ctx.getWsLiterals().getLitThismap());
        ctx.getCarddemoCommarea().setCdemoPgmReenter(true);
        SendMapBuilder.newInstance(ctx.getDfheiblk(), ctx)
        .withMap(ctx.getCcWorkAreas().getCcardNextMap())
        .withMapset(ctx.getCcWorkAreas().getCcardNextMapset())
        .withData(ctx.getGroup1().getCcrdslaoReference())
        .withCursor()
        .withErase()
        .withFreeKB()
        .execute();
        ctx.getWsMiscStorage().setWsRespCd(ctx.getDfheiblk().getEibresp());
    }

    /**
     * Process operation processInputs.
     *
     * @param ctx
     * @param ctrl
     */
    @Override
    public void processInputs(final CocrdslcContext ctx, final ExecutionController ctrl) {
        receiveMap(ctx, ctrl);
        editMapInputs(ctx, ctrl);
        ctx.getCcWorkAreas().setCcardErrorMsg(ctx.getWsMiscStorage().getWsReturnMsg());
        ctx.getCcWorkAreas().setCcardNextProg(ctx.getWsLiterals().getLitThispgm());
        ctx.getCcWorkAreas().setCcardNextMapset(ctx.getWsLiterals().getLitThismapset());
        ctx.getCcWorkAreas().setCcardNextMap(ctx.getWsLiterals().getLitThismap());
    }

```

- *base package*`.`*program*`.statemachine`: esse pacote pode nem sempre estar presente. É necessário quando a modernização do fluxo de controle legado precisa usar uma abordagem de máquina de estado (ou seja, usar a StateMachine estrutura Spring) para cobrir adequadamente o fluxo de execução legado.

  Nesse caso, o subpacote statemachine contém duas classes:

- *Program*`ProcedureDivisionStateMachineController`: uma classe que estende uma classe que implementa as interfaces `StateMachineController` (define as operações necessárias para controlar a execução de uma máquina de estado) e `StateMachineRunner` (define as operações necessárias para executar uma máquina de estado), usadas para conduzir a mecânica da máquina de estado Spring; por exemplo, `SimpleStateMachineController` como no caso de exemplo.



```
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);
    }
}
```

O controlador da máquina de estado define os diferentes estados possíveis e as transições entre eles, que reproduzem o fluxo de controle de execução legado para o programa em questão.

Ao criar a máquina de estado, o controlador se refere aos métodos definidos na classe de serviço associada localizada no pacote da máquina de estado e descrita abaixo:

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

- *Program*ProcedureDivisionStateMachineService: essa classe de serviço representa alguma lógica de negócios que precisa ser vinculada à máquina de estado criada pelo controlador da máquina de estado, conforme descrito anteriormente.

  O código nos métodos dessa classe usa os eventos definidos no controlador da máquina de estado:

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);

    }
}
```

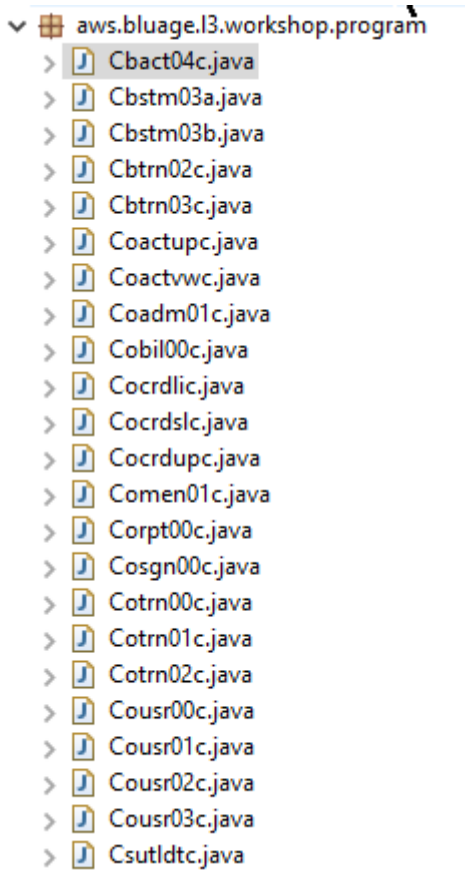
O serviço statemachine também faz chamadas para a implementação do serviço de processo descrita anteriormente:

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);
```

Além disso, um pacote chamado *`base package`*`.program` desempenha um papel importante, pois reúne uma classe por programa, que servirá como ponto de entrada do programa (mais detalhes

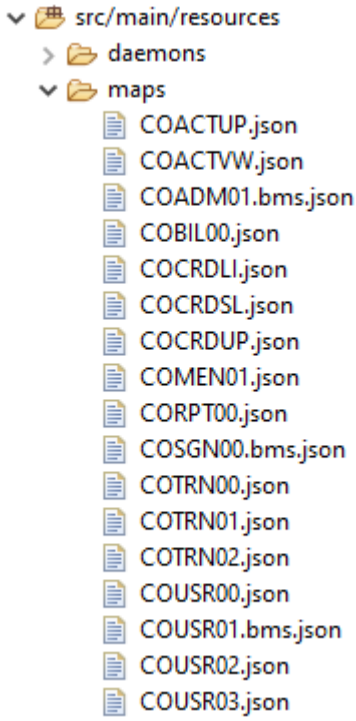
sobre isso posteriormente). Cada classe implementa a `Program` interface, marcador para um ponto de entrada do programa.




Outros artefatos

- Companheiros do BMS MAPs

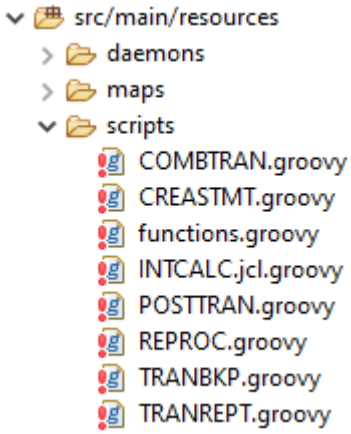
  Além dos artefatos relacionados ao programa, o projeto de serviço pode conter outros artefatos para várias finalidades. No caso da modernização de um aplicativo on-line do CICS, o processo de modernização produz um arquivo json e coloca na pasta map da pasta /src/main/resources:




O tempo de execução do Blu Age consome esses arquivos json para vincular os registros usados pela instrução SEND MAP aos campos da tela.

- Scripts Groovy

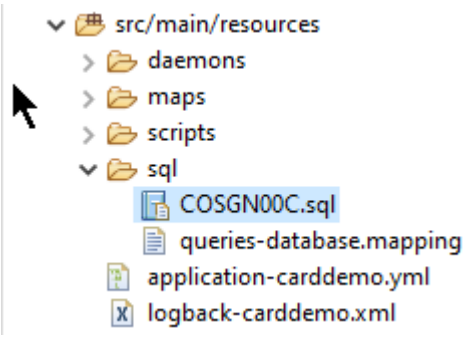
  Se o aplicativo legado tinha scripts JCL, eles foram modernizados como scripts groovy, armazenados na pasta /src/main/resources/scripts (mais sobre esse local específico posteriormente):



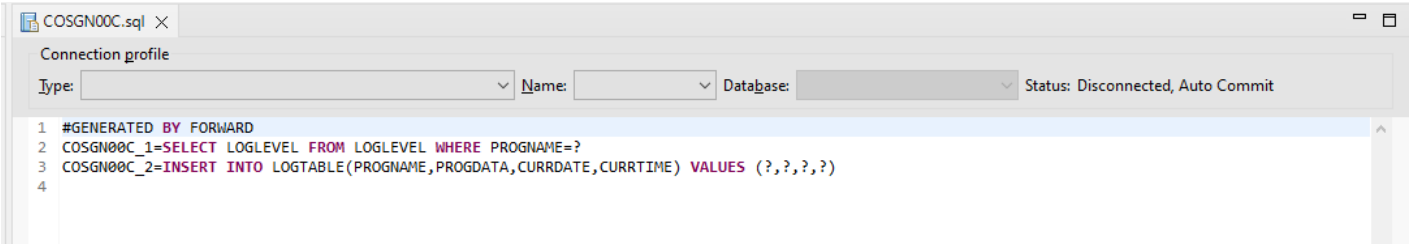

  Esses scripts são usados para iniciar trabalhos em lotes (cargas de trabalho de processamento de dados dedicadas, não interativas e com uso intenso de CPU).

- Arquivos SQL

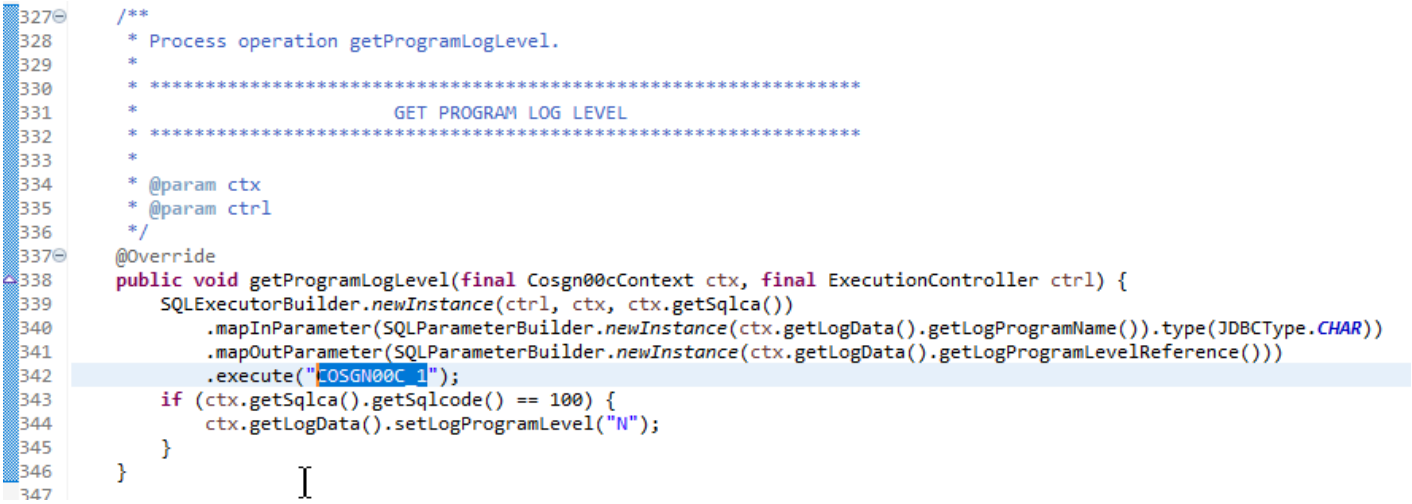
Se o aplicativo legado estava usando consultas SQL, as consultas SQL modernizadas correspondentes foram reunidas em arquivos de propriedades dedicados, com o padrão de nomenclatura program.sql, em que program é o nome do programa que usa essas consultas.




O conteúdo desses arquivos sql é uma coleção de entradas (key=query), em que cada consulta é associada a uma chave exclusiva, que o programa modernizado usa para executar a consulta fornecida:

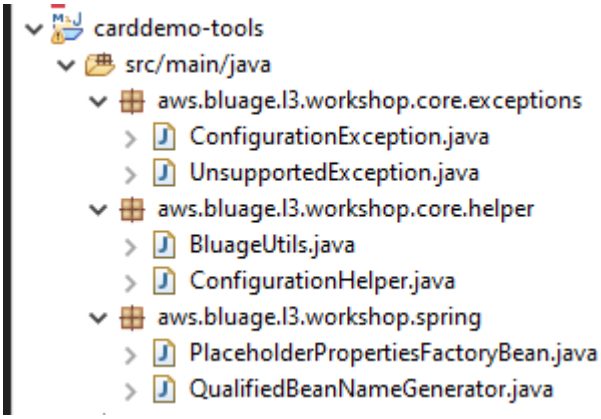



Por exemplo, o programa COSGN00C está executando a consulta com a chave “COSGN00C_1" (a primeira entrada no arquivo sql):



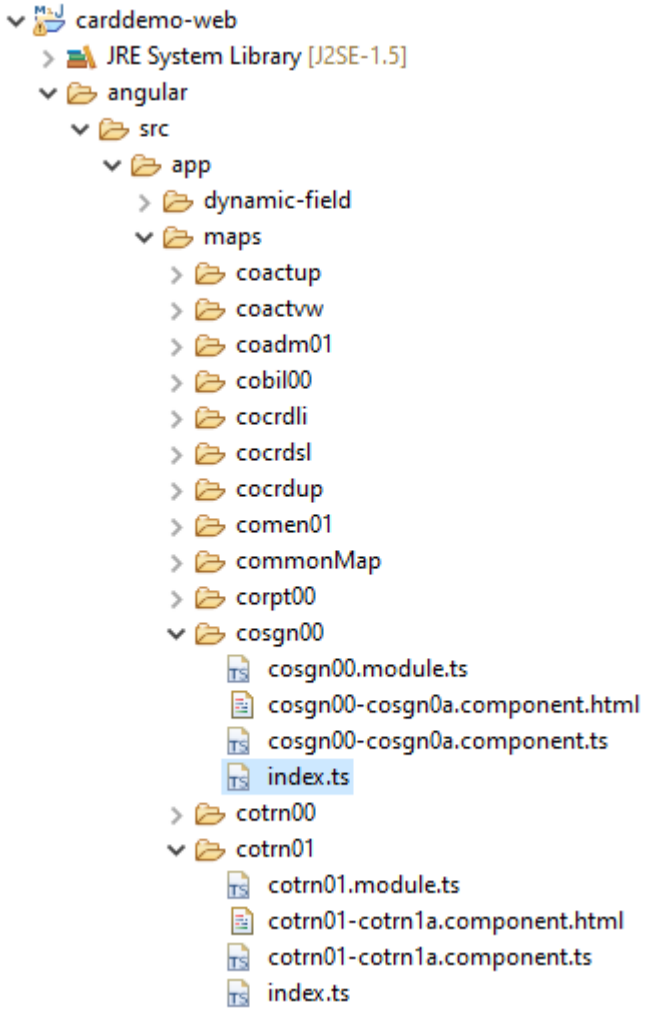
### Conteúdo do projeto de utilitários

O projeto de utilitários, cujo nome termina com “-tools”, contém um conjunto de utilitários técnicos, que podem ser usados por todos os outros projetos.




### Conteúdo do (s) projeto (s) web

O projeto web só está presente ao modernizar elementos legados da interface do usuário. Os elementos modernos da interface de usuário usados para criar o front-end do aplicativo modernizado são baseados no Angular. O aplicativo de exemplo usado para mostrar os artefatos de modernização é um aplicativo COBOL/CICS, executado em um mainframe. O sistema CICS usa MAPs para representar as telas da interface do usuário. Os elementos modernos correspondentes serão, para cada mapa, um arquivo html acompanhado por arquivos Typescript:




O projeto web cuida apenas do aspecto de front-end do aplicativo. O projeto de serviço, que depende dos projetos de utilitários e entidades, fornece os serviços de back-end. O link entre o frontend e o back-end é feito por meio do aplicativo web chamado Gapwalk-Application, que faz parte da distribuição de tempo de execução padrão do Blu Age. AWS

## Executando e chamando programas

Em sistemas legados, os programas são compilados como executáveis autônomos que podem se chamar por meio de um mecanismo CALL, como a instrução COBOL CALL, transmitindo argumentos quando necessário. Os aplicativos modernizados oferecem a mesma capacidade, mas usam uma abordagem diferente, porque a natureza dos artefatos envolvidos difere dos antigos.

No lado modernizado, os pontos de entrada do programa são classes específicas que implementam a `Program` interface, são componentes do Spring (@Component) e estão localizados em projetos de serviço, em um pacote chamado *`base package`*`.program`.

### Registro de programas

Cada vez que o servidor Tomcat que hospeda aplicativos modernizados é iniciado, o aplicativo Springboot do serviço também é iniciado, o que aciona o registro do programa. Um registro dedicado chamado `ProgramRegistry` é preenchido com entradas de programa, cada programa sendo registrado usando seus identificadores, uma entrada por identificador de programa conhecido, o que significa que, se um programa for conhecido por vários identificadores diferentes, o registro conterá tantas entradas quanto identificadores.

O registro de um determinado programa depende da coleção de identificadores retornados pelo método 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;
    }
```

Neste exemplo, o programa é registrado uma vez, sob o nome 'CBACT04C' (veja o conteúdo da coleção ProgramIdentifiers). Os registros do tomcat mostram cada registro do programa. O registro do programa depende apenas dos identificadores declarados do programa e não do nome da classe do programa em si (embora normalmente os identificadores do programa e os nomes das classes do programa estejam alinhados).

O mesmo mecanismo de registro se aplica aos programas utilitários trazidos pelos vários aplicativos web utilitários AWS Blu Age, que fazem parte da distribuição de tempo de execução do AWS Blu Age. Por exemplo, o aplicativo web Gapwalk-Utility-Pgm fornece os equivalentes funcionais dos

utilitários do sistema z/OS (IDCAMS, ICEGENER, SORT etc.) e pode ser chamado por programas ou scripts modernizados. Todos os programas utilitários disponíveis registrados na inicialização do Tomcat são registrados nos registros do Tomcat.

Registro de scripts e daemons

Um processo de registro semelhante, no momento da inicialização do Tomcat, ocorre para scripts groovy que estão localizados na hierarquia de pastas /src/main/resources/scripts. A hierarquia da pasta de scripts é percorrida e todos os scripts groovy descobertos (exceto o script reservado de funções especiais.groovy) são registrados no `ScriptRegistry`, usando seu nome curto (a parte do nome do arquivo de script localizada antes do primeiro caractere de ponto) como chave para recuperação.

> **Note**
>
> - Se vários scripts tiverem nomes de arquivo que resultarão na produção da mesma chave de registro, somente o mais recente será registrado, substituindo qualquer registro encontrado anteriormente para essa chave específica.
> - Considerando a observação acima, preste atenção ao usar subpastas, pois o mecanismo de registro nivela a hierarquia e pode levar a substituições inesperadas. A hierarquia não conta no processo de registro: normalmente, /scripts/a/myscript.groovy e /scripts/b/myscript.groovy farão com que /scripts/b/myscript.groovy sobrescreva /scripts/a/myscript.groovy.

Os scripts groovy na pasta /src/main/resources/daemons são tratados de forma um pouco diferente. Eles ainda estão registrados como scripts regulares, mas, além disso, são lançados uma vez, diretamente no momento da inicialização do Tomcat, de forma assíncrona.

Depois que os scripts são registrados no `ScriptRegistry`, uma chamada REST pode iniciá-los, usando os endpoints dedicados que o aplicativo Gapwalk expõe. Para obter mais informações, consulte Entrada, na documentação do .

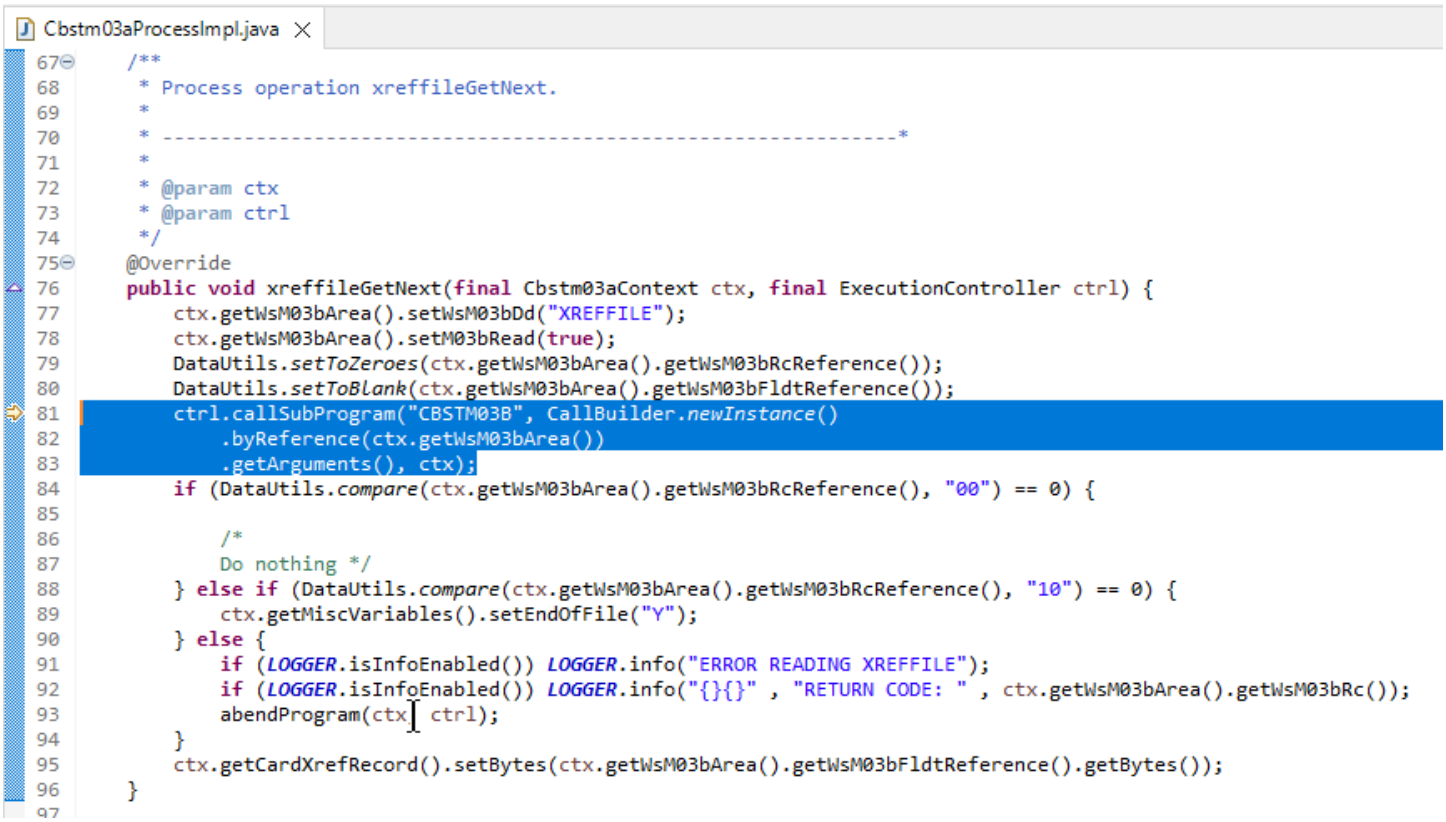
Programas de chamada de programas

Cada programa pode chamar outro programa como subprograma, passando parâmetros para ele. Os programas usam uma implementação da `ExecutionController` interface para fazer isso (na maioria das vezes, isso será uma `ExecutionControllerImpl` instância), junto com um

mecanismo de API fluente chamado de `CallBuilder` para criar os argumentos de chamada do programa.

Todos os métodos de programas usam tanto a `RuntimeContext` quanto an `ExecutionController` como argumentos de método, portanto, um `ExecutionController` está sempre disponível para chamar outros programas.

Veja, por exemplo, o diagrama a seguir, que mostra como o programa CBST03A chama o programa CBST03B como um subprograma, passando parâmetros para ele:




- O primeiro argumento do `ExecutionController.callSubProgram` é um identificador do programa a ser chamado (ou seja, um dos identificadores usados para o registro do programa -- veja os parágrafos acima).
- O segundo argumento, que é o resultado da construção no`CallBuilder`, é uma matriz de `Record`, correspondente aos dados passados do chamador para o chamador.
- O terceiro e último argumento é a `RuntimeContext` instância do chamador.

Todos os três argumentos são obrigatórios e não podem ser nulos, mas o segundo argumento pode ser uma matriz vazia.

O chamador só poderá lidar com os parâmetros passados se tiver sido originalmente projetado para isso. Para um programa COBOL legado, isso significa ter uma seção LINKAGE e uma cláusula USING para que a divisão de procedimentos faça uso dos elementos LINKAGE.

Por exemplo, consulte o arquivo fonte COBOL CBSTM03B.CBL correspondente:

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
```

Portanto, o programa CBSTM03B usa um único `Record` como parâmetro (uma matriz de tamanho 1). Isso é o que `CallBuilder` está construindo, usando o encadeamento de métodos byReference () e getArguments ().

A classe de API `CallBuilder` fluente tem vários métodos disponíveis para preencher a matriz de argumentos a serem passados para um chamador:

- asPointer (RecordAdaptable): adicione um argumento do tipo ponteiro, por referência. O ponteiro representa o endereço de uma estrutura de dados de destino.
- byReference (RecordAdaptable): adicione um argumento por referência. O chamador verá as modificações que o chamador executa.
- ByReference (RecordAdaptable...): variante varargs do método anterior.
- ByValue (Object): adicione um argumento, transformado em um `Record`, por valor. O chamador não verá as modificações que o chamador executa.

- byValue (RecordAdaptable): igual ao método anterior, mas o argumento está diretamente disponível como a. `RecordAdaptable`
- byValueWithLimites (Object, int, int): adicione um argumento, transformado em a`Record`, extraindo a parte da matriz de bytes definida pelos limites fornecidos, por valor.

Finalmente, o método getArguments coletará todos os argumentos adicionados e os retornará como uma matriz de `Record`.

> **Note**
>
> É responsabilidade do chamador garantir que a matriz de argumentos tenha o tamanho necessário, que os itens estejam devidamente ordenados e compatíveis, em termos de layout de memória com os layouts esperados para os elementos de ligação.

Scripts chamando programas

Chamar programas registrados a partir de scripts groovy requer o uso de uma instância de classe implementando a `MainProgramRunner` interface. Normalmente, a obtenção dessa instância é obtida por meio do ApplicationContext uso do 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 = [:]
```

Depois que uma `MainProgramRunner` interface estiver disponível, use o método runProgram para chamar um programa e passar o identificador do programa de destino como parâmetro:

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")
                })
        }
    }
}
```

No exemplo anterior, uma etapa de trabalho chama o IDCAMS (programa utilitário de manipulação de arquivos), fornecendo um mapeamento entre as definições reais do conjunto de dados e seus identificadores lógicos.

Ao lidar com conjuntos de dados, os programas legados geralmente usam nomes lógicos para identificar conjuntos de dados. Quando o programa é chamado a partir de um script, o script deve mapear nomes lógicos com conjuntos de dados físicos reais. Esses conjuntos de dados podem estar no sistema de arquivos, em um armazenamento Blusam ou até mesmo definidos por um fluxo embutido, pela concatenação de vários conjuntos de dados ou pela geração de um GDG.

Use o withFileConfiguration método para criar um mapa lógico para físico dos conjuntos de dados e disponibilizá-lo para o programa chamado.

## Escreva seu próprio programa

Escrever seu próprio programa para que scripts ou outros programas modernizados possam ser chamados é uma tarefa comum. Normalmente, em projetos de modernização, você escreve seus próprios programas quando um programa legado executável é escrito em uma linguagem que o processo de modernização não suporta, ou as fontes foram perdidas (sim, isso pode acontecer) ou o programa é um utilitário cujas fontes não estão disponíveis.

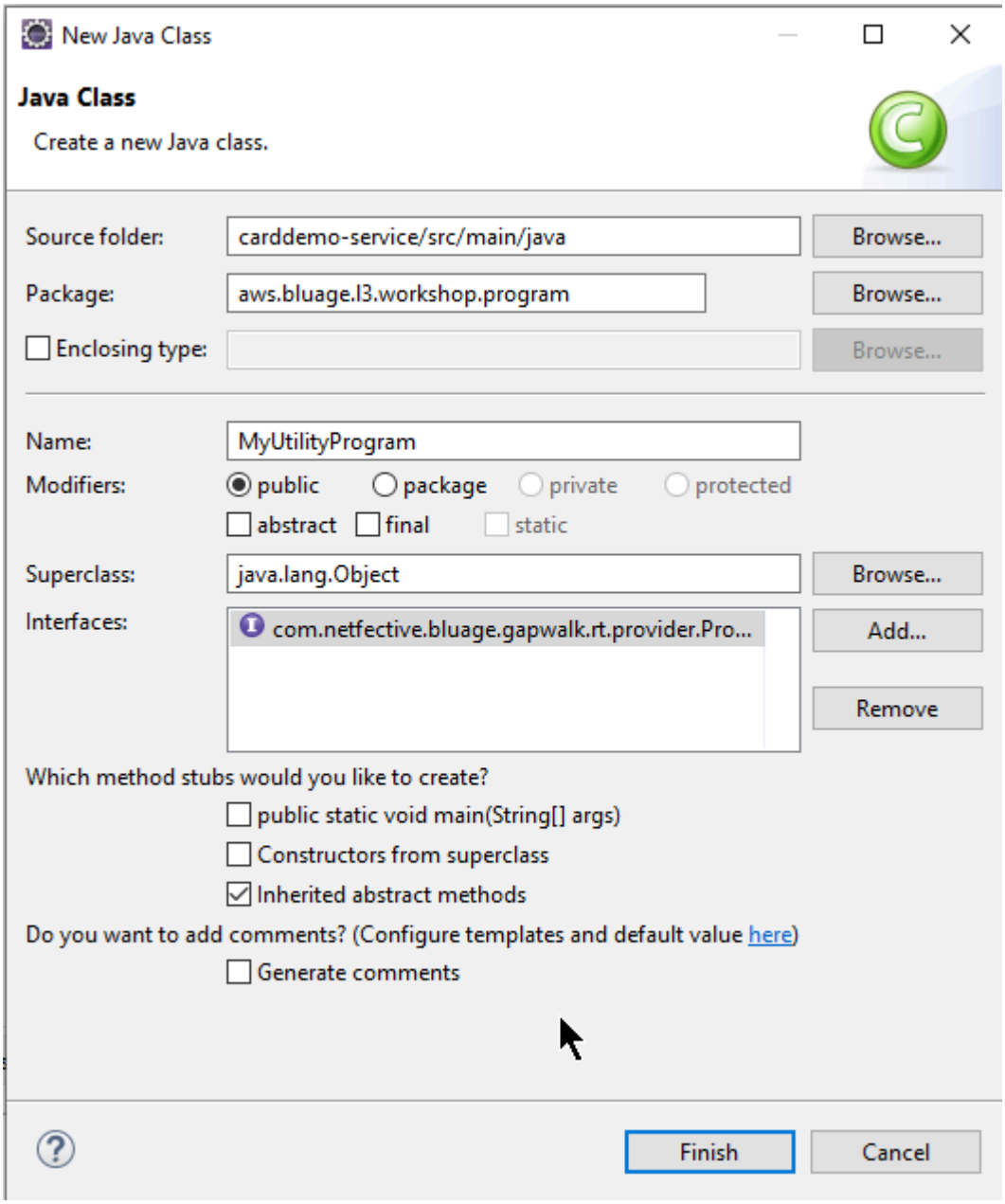
Nesse caso, talvez você precise escrever o programa ausente, em java, sozinho (supondo que você tenha conhecimento suficiente sobre qual deve ser o comportamento esperado do programa, o layout de memória dos argumentos do programa, se houver, e assim por diante). Seu programa java deve estar em conformidade com a mecânica do programa descrita neste documento, para que outros programas e scripts possam executá-lo.

Para garantir que o programa seja utilizável, você deve concluir duas etapas obrigatórias:

- Escreva uma classe que implemente a `Program` interface corretamente, para que ela possa ser registrada e chamada.
- Certifique-se de que seu programa esteja registrado corretamente, para que fique visível em outros programas/scripts.

### Escrevendo a implementação do programa

Use seu IDE para criar uma nova classe java que implemente a interface `Program`:




A imagem a seguir mostra o Eclipse IDE, que se encarrega de criar todos os métodos obrigatórios a serem implementados:

MyUtilityProgram.java

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

import java.util.Set;

public class MyUtilityProgram implements Program {

    @Override
    public ConfigurableApplicationContext getSpringApplication() {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public Set<String> getProgramIdentifiers() {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public Context getContext() {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public void run(ExecutionController ctrl) {
        // TODO Auto-generated method stub

    }

}
```

Integração com o Spring

Primeiro, a classe deve ser declarada como um componente Spring. Anote a classe com a anotação `@Component`:

```java
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 {
```

Em seguida, implemente os métodos necessários corretamente. No contexto desse exemplo, adicionamos o `MyUtilityProgram` ao pacote que já contém todos os programas modernizados.

Esse posicionamento permite que o programa use o aplicativo Springboot existente para fornecer o necessário `ConfigurableApplicationContext` para a implementação do método: getSpringApplication

```
public class MyUtilityProgram implements Program {

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

Você pode escolher um local diferente para seu próprio programa. Por exemplo, você pode localizar o determinado programa em outro projeto de serviço dedicado. Certifique-se de que o projeto de serviço fornecido tenha seu próprio aplicativo Springboot, o que possibilita recuperar o ApplicationContext (que deveria ser um). `ConfigurableApplicationContext`

Dando uma identidade ao programa

Para ser chamado por outros programas e scripts, o programa deve receber pelo menos um identificador, que não deve colidir com nenhum outro programa registrado existente no sistema. A escolha do identificador pode ser motivada pela necessidade de cobrir a substituição de um programa antigo existente; nesse caso, você precisará usar o identificador esperado, conforme encontrado nas ocorrências de CALL encontradas em todos os programas legados. A maioria dos identificadores de programas tem 8 caracteres em sistemas legados.

Criar um conjunto não modificável de identificadores no programa é uma forma de fazer isso. O exemplo a seguir mostra a escolha de “MYUTILPG” como identificador único:

```
@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;
    }
```

Associar o programa a um contexto

O programa precisa de uma `RuntimeContext` instância complementar. Para programas modernizados, o AWS Blu Age gera automaticamente o contexto complementar, usando as estruturas de dados que fazem parte do programa legado.

Se você estiver escrevendo seu próprio programa, também deverá escrever o contexto complementar.

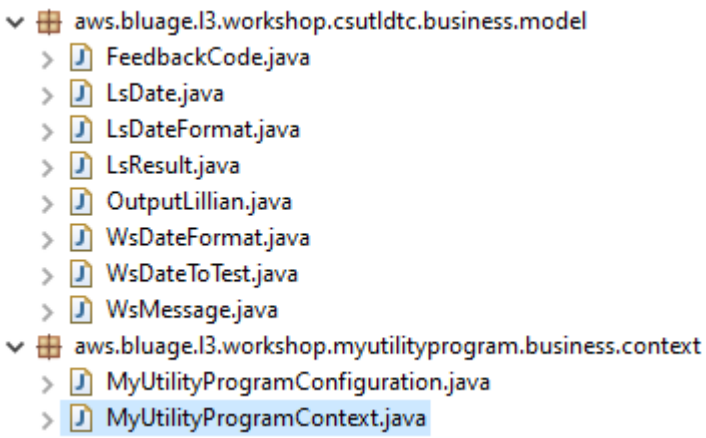
Referindo-se a Aulas relacionadas ao programa, você pode ver que um programa requer pelo menos duas classes complementares:

- uma classe de configuração.
- uma classe de contexto que usa a configuração.

Se o programa utilitário usa alguma estrutura de dados extra, ela também deve ser escrita e usada pelo contexto.

Essas classes devem estar em um pacote que faça parte de uma hierarquia de pacotes que será verificada na inicialização do aplicativo, para garantir que o componente de contexto e a configuração sejam tratados pela estrutura Spring.

Vamos escrever uma configuração e um contexto mínimos, no pacote *`base package`*`.myutilityprogram.business.context`, recém-criado no projeto de entidades:




Aqui está o conteúdo da configuração. Ele está usando uma configuração similar a outros programas -- modernizados -- nas proximidades. Você provavelmente precisará personalizar isso de acordo com suas necessidades específicas.

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();
    }
}

```

Observações:

- A convenção geral de nomenclatura é ProgramNameConfiguração.
- Ele deve usar as anotações @org .springframework.context.annotation.Configuration e @Lazy.
- O nome do bean geralmente segue a ProgramNameContextConfiguration convenção, mas isso não é obrigatório. Certifique-se de evitar colisões de nomes de beans em todo o projeto.
- O único método a ser implementado deve retornar um `Configuration` objeto. Use a API `ConfigurationBuilder` fluente para ajudá-lo a criar uma.

E o contexto associado:

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
    }

}

```

Observações

- A classe de contexto deve estender uma implementação de `Context` interface existente ( `RuntimeContext` ou `JicsRuntimeContext`, que é aprimorada `RuntimeContext` com itens específicos do JICS).
- A convenção geral de nomenclatura é ProgramNameContext.
- Você deve declará-lo como um componente de protótipo e usar a anotação @Lazy.
- O construtor se refere à configuração associada, usando a anotação @Qualifier para direcionar a classe de configuração adequada.
- Se o programa utilitário usar algumas estruturas de dados extras, elas devem ser:
  - Escrito e adicionado ao pacote *`base package`*`.business.model`.
  - referenciado no contexto. Dê uma olhada em outras classes de contexto existentes para ver como referenciar classes de estruturas de dados e adaptar os métodos de contexto (constructor/clean-up/reset) conforme necessário.

Agora que um contexto dedicado está disponível, deixe o novo programa usá-lo:

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);
    }

```

Observações:

- O método getContext deve ser implementado estritamente conforme mostrado, usando uma delegação ao getOrCreate método da `ProgramContextStore` classe e ao Spring conectado automaticamente. `BeanFactory` Um único identificador de programa é usado para armazenar o contexto do programa no `ProgramContextStore`; esse identificador é referenciado como sendo o “identificador principal do programa”.
- A configuração complementar e as classes de contexto devem ser referenciadas usando a anotação spring `@Import`.

## Implementando a lógica de negócios

Quando o esqueleto do programa estiver concluído, implemente a lógica de negócios para o novo programa utilitário.

Faça isso no `run` método do programa. Esse método será executado sempre que o programa for chamado, seja por outro programa ou por um script.

Feliz codificação!

Gerenciando o registro do programa

Por fim, verifique se o novo programa está registrado corretamente no `ProgramRegistry`. Se você adicionou o novo programa ao pacote que já contém outros programas, não há mais nada a ser feito. O novo programa é selecionado e registrado em todos os programas vizinhos na inicialização do aplicativo.

Se você escolheu outro local para o programa, verifique se o programa está registrado corretamente na inicialização do Tomcat. Para se inspirar sobre como fazer isso, veja o método de inicialização das SpringbootLauncher classes geradas no (s) projeto (s) de serviço (consulteConteúdo do projeto de serviço).

Verifique os registros de inicialização do Tomcat. Cada registro do programa é registrado. Se o seu programa for registrado com sucesso, você encontrará a entrada de registro correspondente.

Quando tiver certeza de que seu programa está registrado corretamente, você pode começar a iterar na codificação da lógica de negócios.

## Mapeamentos de nomes totalmente qualificados

Esta seção contém listas do AWS Blu Age e de mapeamentos de nomes totalmente qualificados de terceiros para uso em seus aplicativos modernizados.

AWS Mapeamentos de nomes totalmente qualificados da Blu Age

| Nome curto | Nome totalmente qualificado |
|---|---|
| `CallBuilder` | `com.netfective.bluage.gapwalk.runtime.statements.CallBuilder` |
| `Configuration` | `com.netfective.bluage.gapwalk.datasimplifier.configuration.Configuration` |

| Nome curto | Nome totalmente qualificado |
| --- | --- |
| ConfigurationBuilder | com.netfective.bluage.gapwalk.datasimplifier.configuration.ConfigurationBuilder |
| 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 |

| Nome curto | Nome totalmente qualificado |
| --- | --- |
| SimpleStateMachineController | com.netfective.bluage.gapwalk.rt.statemachine.SimpleStateMachineController |
| StateMachineController | com.netfective.bluage.gapwalk.rt.statemachine.StateMachineController |
| StateMachineRunner | com.netfective.bluage.gapwalk.rt.statemachine.StateMachineRunner |

Mapeamentos de nomes totalmente qualificados de terceiros

| Nome curto | Nome totalmente qualificado |
| --- | --- |
| @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 |

# Simplificador de dados

Em sistemas mainframe e midrange (referidos no tópico a seguir como sistemas “legados”), as linguagens de programação usadas com frequência, como COBOL, PL/I ou RPG, fornecem acesso de baixo nível à memória. Esse acesso se concentra no layout de memória acessado por meio de tipos nativos, como zoneados, compactados ou alfanuméricos, possivelmente também agregados por meio de grupos ou matrizes.

Uma mistura de acessos a uma determinada parte da memória, por meio de campos digitados e como acesso direto a bytes (memória bruta), coexiste em um determinado programa. Por exemplo, os programas COBOL passarão argumentos aos chamadores como conjuntos contíguos de bytes (LINKAGE) ou lerão e gravarão dados de arquivos da mesma maneira (registros), enquanto interpretam esses intervalos de memória com campos digitados organizados em cadernos.

Essas combinações de acesso bruto e estruturado à memória, a dependência de um layout de memória preciso em nível de byte e tipos legados, como zoneados ou compactados, são recursos que não estão disponíveis de forma nativa nem facilmente no ambiente de programação Java.

Como parte da solução AWS Blu Age para modernizar programas legados para Java, a biblioteca Data Simplifier fornece essas construções para programas Java modernizados e as expõe de uma forma que seja o mais familiar possível para os desenvolvedores Java (getters/setters, matrizes de bytes, baseadas em classes). É uma dependência central do código Java modernizado gerado a partir desses programas.

Para simplificar, a maioria das explicações a seguir é baseada em construções COBOL, mas você pode usar a mesma API para modernização do layout de dados PL1 e RPG, já que a maioria dos conceitos é semelhante.

Tópicos



## Classes principais

Para facilitar a leitura, este documento usa os nomes abreviados Java das interfaces e classes da API AWS Blu Age. Para ter mais informações, consulte FQN dos tipos de Java discutidos.

### Representação de baixo nível

No nível mais baixo, a memória (uma faixa contígua de bytes acessível de forma rápida e aleatória) é representada pela interface. `Record` Essa interface é essencialmente uma abstração de uma matriz de bytes de tamanho fixo. Dessa forma, ele fornece setters e getters capazes de acessar ou modificar os bytes subjacentes.

### Representação de dados estruturados

Para representar dados estruturados, como “01 itens de dados” ou “01 cadernos”, conforme encontrado em COBOL DATA DIVISION, são usadas subclasses da `RecordEntity` classe. Normalmente, eles não são escritos à mão, mas gerados pelas ferramentas de modernização do AWS Blu Age a partir das construções legadas correspondentes. Ainda é útil conhecer sua estrutura principal e sua API, para que você possa entender como o código em um programa modernizado as usa. No caso do COBOL, esse código é gerado em Java a partir de sua DIVISÃO DE PROCEDIMENTOS.

O código gerado representa cada “01 item de dados” com uma `RecordEntity` subclasse; cada campo elementar ou agregado que o compõe é representado como um campo Java privado, organizado como uma árvore (cada item tem um pai, exceto o raiz).

Para fins ilustrativos, aqui está um exemplo de item de dados COBOL, seguido pelo código correspondente gerado pelo AWS Blu Age que o moderniza:

```
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);
    }
}
```

Campos elementares

Os campos de classe `Elementary` (ou `Filler`, quando não nomeados) representam uma “folha” da estrutura de dados legada. Eles estão associados a uma extensão contígua de bytes subjacentes (“intervalo”) e geralmente têm um tipo (possivelmente parametrizado) que expressa como interpretar e modificar esses bytes (“decodificando” e “codificando”, respectivamente, um valor de/para uma matriz de bytes).

Todos os tipos elementares são subclasses de `RangeType`. Os tipos comuns são:

| Tipo COBOL | Tipo de simplificador de dados |
| --- | --- |
| `PIC X(n)` | `AlphanumericType` |
| `PIC 9(n)` | `ZonedType` |

| Tipo COBOL | Tipo de simplificador de dados |
| --- | --- |
| `PIC 9(n) COMP-3` | `PackedType` |
| `PIC 9(n) COMP-5` | `BinaryType` |

Agregar os campos

Os campos agregados organizam o layout de memória de seus conteúdos (outros agregados ou campos elementares). Eles próprios não têm um tipo elementar.

Campos de `Group` representam campos contíguos na memória. Cada um dos campos contidos é disposto na mesma ordem na memória, o primeiro campo está em deslocamento em `0` relação à posição do campo do grupo na memória, o segundo campo em deslocamento`0 + (size in bytes of first field)`, etc. Eles são usados para representar sequências de campos COBOL sob o mesmo campo contendo.

Campos de `Union` representam vários campos acessando a mesma memória. Cada um dos campos contidos é disposto em deslocamento em `0` relação à posição do campo de união na memória. Eles são usados, por exemplo, para representar a construção COBOL “REDEFINES” (os primeiros filhos da União sendo o item de dados redefinido, os segundos filhos sendo sua primeira redefinição, etc.).

Os campos de matriz (subclasses de `Repetition`) representam a repetição, na memória, do layout de seu campo filho (seja um agregado em si ou um item elementar). Eles apresentam um determinado número desses layouts infantis na memória, cada um em offset `index * (size in bytes of child)`. Eles são usados para representar construções COBOL “OCCURS”.

Primitivos

Em alguns casos de modernização, “Primitivos” também podem ser usados para apresentar itens de dados “raiz” independentes. Eles são muito semelhantes em uso`RecordEntity`, mas não vêm dele, nem são baseados no código gerado. Em vez disso, eles são fornecidos diretamente pelo tempo de execução do AWS Blu Age como subclasses da `Primitive` interface. Exemplos dessas aulas fornecidas são `Alphanumeric` ou`ZonedDecimal`.

## Vinculação e acesso a dados

A associação entre dados estruturados e dados subjacentes pode ser feita de várias maneiras.

Uma interface importante para esse propósito é `RecordAdaptable` a que é usada para obter `Record` uma “visão gravável” dos dados `RecordAdaptable` subjacentes. Como veremos abaixo, várias classes são implementadas`RecordAdaptable`. Reciprocamente, as APIs e o código do AWS Blu Age que manipulam memória de baixo nível (como argumentos de programas, registros de E/S de arquivos, vírgulas do CICS, memória alocada...) geralmente esperam um como um identificador para essa memória. `RecordAdaptable`

No caso de modernização do COBOL, a maioria dos itens de dados está associada à memória, que será corrigida durante a vida útil da execução do programa correspondente. Para isso, `RecordEntity` as subclasses são instanciadas uma vez em um objeto pai gerado (o contexto do programa) e se encarregarão de instanciar seu subjacente `Record`, com base no tamanho do byte `RecordEntity`

Em outros casos de COBOL, como associar elementos LINKAGE a argumentos do programa ou modernizar a construção SET ADDRESS OF, uma `RecordEntity` instância deve ser associada a um fornecido `RecordAdaptable`. Para isso, existem dois mecanismos:

- se a `RecordEntity` instância já existir, o `RecordEntity.bind(RecordAdaptable)` método (herdado de `Bindable`) pode ser usado para fazer essa instância “apontar” para ela `RecordAdaptable`. Qualquer getter ou setter chamado no `RecordEntity` será então apoiado (leitura ou gravação de bytes) pelos bytes `RecordAdaptable` subjacentes.
- se `RecordEntity` for para ser instanciado, um construtor gerado aceitando a estará disponível `RecordAdaptable`.

Por outro lado, os dados `Record` atualmente vinculados aos dados estruturados podem ser acessados. Para isso, `RecordEntity` implementos`RecordAdaptable`, portanto, `getRecord()` podem ser chamados em qualquer instância desse tipo.

Finalmente, muitos verbos COBOL ou CICS exigem acesso a um único campo, para fins de leitura ou escrita. A `RangeReference` classe é usada para representar esse acesso. Suas instâncias podem ser obtidas a partir de `getXXXReference()` métodos `RecordEntity` gerados (`XXX` sendo o campo acessado) e passadas para métodos de tempo de execução. `RangeReference` normalmente é usado para acessar todo o `RecordEntity` ou `Group`, enquanto sua subclasse `ElementaryRangeReference` representa acessos aos campos `Elementary`.

Observe que a maioria das observações acima se aplica às `Primitive` subclasses, pois elas se esforçam para implementar um comportamento semelhante ao fornecido pelo `RecordEntity` tempo de execução do AWS Blu Age (em vez do código gerado). Para este propósito, todas as subclasses

de `Primitive` implementam `RecordAdaptable` `ElementaryRangeReference` e `Bindable` interfaces de forma a serem utilizáveis no lugar das `RecordEntity` subclasses e dos campos elementares.

## FQN dos tipos de Java discutidos

A tabela a seguir mostra os nomes totalmente qualificados dos tipos Java discutidos nesta seção.

| Nome curto | Nome totalmente qualificado |
| --- | --- |
| `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` |

| Nome curto | Nome totalmente qualificado |
| --- | --- |
| Group | com.netfective.bluage.gapwalk.datasimplifier.data.structure.Group |
| PackedType | com.netfective.bluage.gapwalk.datasimplifier.metadata.type.PackedType |
| 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 |

| Nome curto | Nome totalmente qualificado |
| --- | --- |
| `Union` | `com.netfective.bluage.gapwalk.datasimplifier.data.structure.Union` |
| `ZonedDecimal` | `com.netfective.bluage.gapwalk.datasimplifier.elementary.ZonedDecimal` |
| `ZonedType` | `com.netfective.bluage.gapwalk.datasimplifier.metadata.type.ZonedType` |

# AWS Arquivos de configuração e configuração do Blu Age Runtime

A estrutura Velocity e o código do cliente são aplicativos da web que usam a estrutura Spring Boot. Ele aproveita os recursos do Spring para fornecer configuração, com vários locais possíveis e regras de precedência. Também existem regras de precedência semelhantes para fornecer muitos outros arquivos, como scripts groovy, sql etc.

A estrutura Velocity também contém aplicativos web opcionais adicionais, que podem ser ativados se necessário.

Tópicos

- Noções básicas de configuração de aplicativos
- Precedência do aplicativo
- JNDI para bancos de dados
- Usando AWS segredos
- Outros arquivos (groovy, sql, etc.)
- Aplicativo web adicional
- Habilitando propriedades
- Configurar autenticação para aplicativos do Gapwalk

# Noções básicas de configuração de aplicativos

A forma padrão de lidar com a configuração do aplicativo é por meio do uso de arquivos YAML dedicados a serem fornecidos na `config` pasta do servidor do aplicativo. Há dois arquivos principais de configuração do YAML:

- `application-main.yaml`
- `application-`*`profile`*`.yaml` (onde *`profile`* o valor é configurado durante a geração do aplicativo).

O primeiro arquivo configura a estrutura, ou seja `Gapwalk-application.war`, enquanto o segundo é para opções adicionais específicas para o aplicativo cliente. Isso funciona com o uso de perfis de primavera: o aplicativo Gapwalk usa o perfil `main`, enquanto o aplicativo cliente usa o perfil *`profile`*.

O exemplo a seguir mostra um arquivo YAML principal típico.

```
########################################
##### 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:
```

O exemplo a seguir mostra um arquivo YAML típico do cliente.

```
# 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
```

Para obter informações sobre o conteúdo dos arquivos YAML, consulte Habilitando propriedades

## Precedência do aplicativo

Para esses arquivos de configuração, as regras de precedência do Spring se aplicam. Notavelmente:

- O arquivo `application-main` YAML aparece no arquivo war principal do Gapwalk com valores padrão, e o da `config` pasta o substitui.
- O mesmo deve ser feito para a configuração do aplicativo cliente.
- Parâmetros adicionais podem ser transmitidos na linha de comando no momento da inicialização do servidor. Eles substituiriam os do YAML.

Para obter mais informações, consulte a documentação oficial do Spring Boot.

## JNDI para bancos de dados

A configuração do banco de dados pode ser fornecida com JNDI no arquivo context.xml no Tomcat. Qualquer configuração desse tipo substituiria a YAML. Mas preste atenção, pois usar isso não permitirá agrupar suas credenciais em um gerenciador secreto (veja abaixo).

O exemplo a seguir mostra exemplos de configurações para JICS e BluSam bancos de dados.

```
<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

Seria `jdbc/jics` para o banco de dados JICS e `jdbc/bluesam` (preste atenção ao 'e') para o banco de dados blusam.

url="jdbc:postgresql: //xxxx.rds.amazonaws.com:5432/XXXX” nome de usuário="xxxx” senha="xxxx”

O URL, nome de usuário e senha do banco de dados.

## Usando AWS segredos

Algumas das configurações de recursos que contêm credenciais podem ser ainda mais protegidas usando AWS segredos. A ideia é armazenar dados críticos em um AWS segredo e ter uma referência ao segredo na configuração do YAML para que o conteúdo secreto seja escolhido rapidamente na inicialização do tomcat.

### Segredos para Aurora

A configuração do banco de dados Aurora (para jics, blusam, banco de dados do cliente etc.) usará o segredo do banco de dados integrado, que preencherá automaticamente todos os campos relevantes do banco de dados correspondente.

> **Note**
>
> A chave `dbname` é opcional, dependendo da configuração do seu banco de dados, ela entrará no segredo ou não. Você pode adicioná-lo manualmente ou fornecendo o nome ao arquivo YAML.

### Outros segredos

Outros segredos são para recursos que têm uma única senha (principalmente caches redis protegidos por senha). Nesse caso, o outro tipo de segredo deve ser usado, com uma única chave `password`.

## Referências YAML a segredos

Eles `application-main.yaml` podem referenciar o ARN secreto para vários recursos. Os mais importantes são:

- Credenciais do banco de dados JICS com `spring.aws.jics.db.secret`
- Credenciais do JICS TS Queues Redis com `spring.aws.client.jics.queues.ts.redis.secret`
- Credenciais do banco de dados Blusam com `spring.aws.client.bluesam.db.secret`
- Senha de cache Blusam com `spring.aws.client.bluesam.redis.secret`
- Blusam bloqueia a senha do cache com `spring.aws.client.bluesam.locks.redis.secret`

O exemplo a seguir mostra como declarar esses segredos em um arquivo YAML.

```
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
```

nome do banco de dados: bluesam

> Neste exemplo, o nome do banco de dados não está no segredo e, em vez disso, é fornecido aqui.

O cliente `application-profile.yaml` pode referenciar o ARN secreto do banco de dados do cliente. Isso requer uma propriedade adicional para listar as fontes de dados, mostradas no exemplo abaixo:

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

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

nomes: primário, host

Um exemplo com duas fontes de dados de clientes chamadas primary e host, cada uma com seu banco de dados e credenciais.

nome do banco de dados: mydb

Neste exemplo, o nome do banco de dados “host” não está no segredo e é fornecido aqui, enquanto que para o banco de dados “primário” ele está no segredo.

Nenhuma chave secreta compatível com XA

- motor (postgres/oracle/db2/mssql)
- porta
- dbname
- Esquema atual
- username
- password
- url

Pois `postgres` somente a chave `sslMode` secreta valued (`disable/allow/prefer/require/verify-ca/verify-full`), além da propriedade `spring.aws.rds.ssl.cert-path` yml, permite a conexão com SSL.

Chaves secretas compatíveis com XA

Se o banco de dados do cliente estiver usando XA, as subxa-propriedades são suportadas por meio de valores secretos.

- host
- porta
- dbname
- Esquema atual
- username
- password
- url
- Conexão SSL (verdadeiro/falso)

No entanto, para outras propriedades x (por exemplo, `maxPoolSize` ou`driverType`), a chave YAML normal ainda `spring.jta.atomikos.datasource.XXXX.unique-resource-name` deve ser fornecida.

O valor secreto substitui as propriedades YAML.

## Outros arquivos (groovy, sql, etc.)

Os outros arquivos usados pelo projeto do cliente usam regras de precedência semelhantes às da configuração do Spring. Exemplos:

- Os scripts do Groovy são arquivos `.groovy` na pasta ou subpastas `scripts`.
- Os scripts SQL são arquivos `.sql` na pasta ou subpastas `sql`.
- Os scripts Daemon são arquivos `.groovy` na pasta ou subpastas `daemons`.
- Consultas O arquivo de mapeamento do banco de dados são arquivos nomeados `queries-database.mapping` nas subpastas da pasta `sql`.
- Os modelos do Jasper são arquivos `.jrxml` na pasta ou subpastas `templates`.
- Os catálogos de conjuntos de dados são arquivos `.json` na pasta `catalog`.
- Os arquivos Lnk são arquivos `.json` na pasta `lnk`.

Todos esses locais podem ser substituídos por meio de uma propriedade do sistema ou de uma propriedade yml do cliente.

- Para scripts do Groovy: `configuration.scripts`
- Para scripts SQL: `configuration.sql`
- Para scripts Daemon: `configuration.daemons`
- Para o arquivo de mapeamento do banco de dados de consultas: `configuration.databaseMapping`
- Para modelos Jasper: `configuration.templates`
- Para catálogos de conjuntos de dados: `configuration.catalog`
- Para arquivos Lnk: `configuration.lnk`

Se a propriedade não for encontrada, os arquivos serão retirados do local padrão mencionado acima. A pesquisa será feita primeiro com o diretório de trabalho do tomcat como raiz e, por último, no arquivo war do aplicativo.

## Aplicativo web adicional

A estrutura Velocity contém aplicativos web adicionais em sua pasta `webapps-extra`. Esses aplicativos não são atendidos por padrão pelo servidor tomcat.

A adesão a esses aplicativos web depende do projeto de modernização e é feita movendo o arquivo war desejado da pasta `webapps-extra` para a pasta `webapps`. Depois disso, a guerra será atendida pelo servidor tomcat na próxima inicialização.

Algumas configurações adicionais específicas do projeto também podem ser adicionadas em um arquivo de configuração YAML para cada guerra adicional, conforme feito no `application-main.yml` arquivo e explicado acima. As guerras adicionais são:

- `gapwalk-utility-pgm.war`: contém suporte para programas utilitários do ZOS e usa `application-utility-pgm.yaml` como configuração.
- `gapwalk-cl-command.war`: contém suporte para programas utilitários AS/400 e usa `application-cl-command.yaml` como configuração.
- `gapwalk-hierarchical-support.war`: contém suporte a transações IMS/MFS e usa como configuração `application-jhdb.yaml`

# Habilitando propriedades

Nos aplicativos Spring Boot, `application-main.yml` é o arquivo de configuração no qual definimos diferentes tipos de propriedades, como porta de escuta, conectividade do banco de dados e muito mais.

Tópicos



## Notação YML

Na documentação a seguir, uma propriedade como a `parent.child1.child2=true` é escrita da seguinte forma no formato YAML.

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

## Início rápido / Casos de uso

Os casos de uso a seguir mostram exemplos das chaves e valores aplicáveis.

- Application-main.yml padrão

```
----
#### 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
----
```

- Use arquivos de comprimento variável com os comandos 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)
```

- Forneça o valor do indicador de bytes nulos no utilitário 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
```

## Propriedades disponíveis para o aplicativo principal

Esta tabela fornece uma visão exaustiva dos parâmetros de chave/valores.

| Chave | Tipo | Valor padrão | Descrição |
| --- | --- | --- | --- |
| `logging.config` | Path | caminho de classe: logback-main.xml | Chave padrão para a referência ao arquivo de configuração de logback. Outras chaves de registro padrão também estão disponíveis. |
| `spring.jta.enabled` | boolean | false | Chave Padrão. Se o modo de suporte da fonte de dados não for static-xa, a configuração automática das transações do Spring JTA deverá ser desativada. |
| `datasource.jicsDs` + `-driver-class-name` + `-url` + `-username` + `-password` + `-type` | Fonte de dados Spring padrão com subchaves | | Contém as informações de conexão do banco de dados do Jics. Como alternativa, o uso de segredos da AWS é fortemente encorajado, conforme explicado em xref:.. / configuration/configuration.adoc [Configuração]. |

| Chave | Tipo | Valor padrão | Descrição |
| --- | --- | --- | --- |
| `datasource.bluesamDs + -driver-class-name + -url + -username + -password + -type` | Fonte de dados Spring padrão com subchaves | | Contém as informações de conexão do banco de dados Blusam. Como alternativa, o uso de segredos da AWS é fortemente encorajado, conforme explicado em xref:.. /configuration/configuration.adoc [Configuração]. |
| `bluesam.disabled` | boolean | false | Se deve desativar completamente o blusam. |
| `bluesam.cache` | string | | Se não estiver definido, o cache blusam não será usado. Os valores possíveis (implementações de cache) são ehcache e redis. |
| `forcedDate` | string | | Força a data até a data fornecida, se houver uma. |
| `frozenDate` | booliano | true | Especifica se a data deve ser congelada. Aplica-se somente se `forcedDate` também estiver definido. |

| Chave | Tipo | Valor padrão | Descrição |
| --- | --- | --- | --- |
| `externalSort.threshold` | tamanho dos dados (por exemplo, 12 MB) | | O limite de classificação: quando mudar para a classificação externa (mesclagem). |
| `jics.parameters.datform` | string | MMDDY | O formato de data. |
| `jics.initList` | string | | A lista inicialize jics, separada por vírgulas. Se presente, ele define nomes de listas separados por vírgula a serem ativados na inicialização do tomcat entre as listas do CICS. Valor de exemplo:`$UUU,DFH$IVPL,PEZ1` . Isso será transmitido em cascata para os grupos contidos nessas listas e suas definições de recursos subjacentes, que serão então visíveis para o tempo de execução. Vazio por padrão. |
| `jics.parameters.applid` | string | VELOCITY | Eles se aplicam para identificar o aplicativo no JICS (pelo menos 4 caracteres, sem tamanho máximo). |

| Chave | Tipo | Valor padrão | Descrição |
| --- | --- | --- | --- |
| `jics.parameters.sysid` | string | CICS | A identificação do sistema (SYSID). |
| `jics.parameters.eibtrmid` | string | PRAZO | O identificador do terminal (máximo de 4 caracteres, mínimo 1). |
| `jics.parameters.userid` | string | | O ID do usuário (máximo de 8 caracteres, sem mínimo). Quando nenhum valor é fornecido (em branco por padrão), o ID da sessão HTTP é usado como o ID do usuário. |
| `jics.parameters.username` | string | MEU NOME DE USUÁRIO | O nome de usuário (máximo de 10 caracteres, mínimo 1). |
| `jics.parameters.netname` | string | MEU NOME DE REDE | O nome da rede (máximo de 8 caracteres, 1 mínimo). |
| `jics.parameters.opid` | string | XXX | A identificação do operador de 3 caracteres. |
| `jics.parameters.jobname` | string | NOME DO MEU TRABALHO | O nome do trabalho. |
| `jics.parameters.sysname` | string | SYSNAME | O nome do sistema AS400 (sysname). |

| Chave | Tipo | Valor padrão | Descrição |
| --- | --- | --- | --- |
| `jics.parameters.cwa.length` | número | 0 | O comprimento da área de trabalho comum (cwa). |
| `jics.parameters.charset` | string | CP037 | O conjunto de caracteres usado globalmente pelo JICS. |
| `jics.parameters.tsqimpl` | string | barra azul | Implementação da fila de armazenamento temporário (TSQ) do JICS (os valores permitidos sãobluesam/memory/redis) |
| `jics.queues.ts.redis.hostname` | string | 127.0.0.1 | O nome do host do servidor jics cache redis. |
| `jics.queues.ts.redis.port` | número | 6379 | A porta do servidor redis de cache jics. |
| `jics.queues.ts.redis.password` | string | redis | A senha do servidor redis do cache jics. |
| `jics.queues.ts.redis.username` | string | | O nome de usuário do servidor redis de cache jics. O padrão é em branco (sem nome de usuário). |

| Chave | Tipo | Valor padrão | Descrição |
| --- | --- | --- | --- |
| `jics.queues.ts.redis.mode` | string | autônomo | O modo de cache do jics. Os valores possíveis são `standalone` ou `cluster`. O padrão é `standalone`. |
| `lockTimeout` | número | 500 | O tempo limite do bloqueio, em milissegundos. |
| `sqlCodePointShift` | número | | Opcional. A mudança de ponto do código sql. Muda o ponto de código dos caracteres de controle que podemos encontrar ao migrar dados rdbms legados para um rdbms moderno. Por exemplo, você pode especificar `384` para corresponder ao `\u0180` caractere Unicode. |
| `sqlIntegerOverflowAllowed` | boolean | false | Especifica se é permitido o estouro de números inteiros SQL, ou seja, se é permitido colocar valores maiores na variável do host. |

| Chave | Tipo | Valor padrão | Descrição |
| --- | --- | --- | --- |
| `database.cursor.overflow.allowed` | booliano | true | Especifica se é permitido que o cursor transborde. Defina como `true` para realizar uma próxima chamada no cursor, seja qual for sua posição. Defina como `false` para verificar se o cursor está na última posição antes de realizar uma próxima chamada no cursor. Ative somente se o cursor for ROLÁVEL (SENSÍVEL ou INSENSÍVEL). |
| `reportOutputPath` | string | `/reports` | O caminho de saída do relatório. |
| `spring.session.store-type` | string | nenhuma | O cache da sessão para ambientes de alta disponibilidade. Os valores possíveis são none ou `redis`. O padrão énone. |

| Chave | Tipo | Valor padrão | Descrição |
| --- | --- | --- | --- |
| `stopExecutionWhenProgNotFound` | booliano | true | Especifica se a execução deve ser interrompida se um programa não for encontrado. Se definido como`true`, interrompe a execução se um programa não for encontrado. |
| `forceHR` | boolean | false | Especifica se deve ser usado o SYSPRINT legível por humanos, no console ou na saída do arquivo. |
| `rollbackOnRTE` | boolean | false | Especifica se a transação de unidade de execução implícita deve ser revertida em exceções de tempo de execução. |
| `sctThreadLimit` | longo | 5 | O limite de threads para acionar scripts. |

| Chave | Tipo | Valor padrão | Descrição |
| --- | --- | --- | --- |
| `dataSimplifier.onInvalidNumericData` | string | reject | Como reagir ao decodificar dados numéricos inválidos. Os valores permitidos são `reject` / `toleratespaces` / `toleratespaceslowvalues` / `toleratemost` . O padrão é `reject`. |
| `filesDirectory` | string | | O diretório para arquivos de entrada/saída de lotes. |
| `ims.messages.extendedSize` | boolean | false | Especifica se o extendedSize deve ser definido nas mensagens ims. |
| `defaultKeepExistingFiles` | boolean | false | Especifica se o valor anterior padrão do conjunto de dados deve ser definido. |
| `jics.db.ddlScriptLocation` | string | | A localização do script Jics ddl. Permite que você inicie o esquema do banco de dados jics usando um script.sql. Em branco por padrão. Por exemplo, `./jics/sql/jics.sql` . |

| Chave | Tipo | Valor padrão | Descrição |
| --- | --- | --- | --- |
| `jics.db.schemaTestQueryLocation` | string | | Localização do arquivo sql que deve conter uma consulta exclusiva que retorna o número de objetos no esquema jics (se houver). |
| `jics.db.dataScriptLocation` | string | | Localização do script initJics.sql, preparado pelo Analyzer a partir da análise das exportações de CSD do mainframe. |
| `jics.db.dataTestQueryLocation` | string | | Localização de um script sql contendo uma única consulta sql que deve retornar uma contagem de objetos (por exemplo: contagem do número de registros na tabela do programa jics). Se a contagem for igual a 0, o banco de dados será carregado usando o script`jics.db.dataScriptLocation`, caso contrário, o carregamento do banco de dados será ignorado. |

| Chave | Tipo | Valor padrão | Descrição |
| --- | --- | --- | --- |
| `jics.data.dataJsonInitLocation` | string | | |
| `jics.xa.agent.timeout` | número | | |
| `query.useConcatCondition` | boolean | false | Especifica se a condição da chave é criada por concatenação de chaves ou não. |
| `system.qdecfmt` | string | | |
| `disposition.checkexistence` | boolean | false | Especifica se deve liberar uma verificação da existência do arquivo para o conjunto de dados com DISP SHR ou OLD. |
| `useControlMVariable` | boolean | false | Especifica se a especificação Control-m deve ser usada para substituição de variáveis. |
| `card.encoding` | string | CP1145 | Codificação do cartão: para ser usada com`useControlMVariable`. |

| Chave | Tipo | Valor padrão | Descrição |
| --- | --- | --- | --- |
| `mapTransfo.prefixes` | string | &,@,%% | Lista de prefixos a serem usados ao transformar variáveis ControlM. Cada um separado por vírgula. |
| `checkinputfilesize` | boolean | false | Especifica se um cheque deve ser liberado se o tamanho do arquivo for múltiplo do tamanho do registro. |
| `stepFailWhenAbend` | booliano | true | Especifica se a suspensão deve ser levantada se uma etapa falhar ou concluir a execução. |
| `bluesam.fileLoading.commitInterval` | número | 100000 | O intervalo de confirmação do bluesam. |
| `uppercaseUserInput` | booliano | true | Especifica se a entrada do usuário deve estar em maiúsculas. |
| `jhdb.lterm` | string | | Permite que você force um ID de terminal lógico comum no caso de uma emulação IMS. Se não for definido, o sessionID será usado. |

| Chave | Tipo | Valor padrão | Descrição |
| --- | --- | --- | --- |
| `jhdb.identificationCardData` | string | | Usado para codificar alguns “dados do cartão de identificação do operador” no campo MID designado pelo parâmetro CARD. Em branco por padrão, sem restrição de entrada. |
| `encoding` | string | ASCII | A codificação usada em projetos (não em arquivos groovy). Espera uma codificação válida`CP1047`,`IBM930`,`ASCII`, |
| `cl.configuration.context.encoding` | string | CP297 | A codificação dos arquivos CL. Espera uma codificação válida`CP1047`,`IBM930`,`ASCII`, O valor padrão é`CP297` |
| `cl.zonedMode` | string | EBCDIC_STRICT | O modo para codificar ou decodificar comandos da linguagem de controle (CL). Os valores permitidos são`EBCDIC_STRICT` /`EBCDIC_MODIFIED` /`AS400`. |

| Chave | Tipo | Valor padrão | Descrição |
| --- | --- | --- | --- |
| `ims.programs` | string | | Lista de programas IMS a serem usados. Separe cada parâmetro com um ponto e vírgula (`;`) e cada transação com uma vírgula (). `,` Por exemplo: `PCP008,PCT008;PCP054,PCT054;PCP066,PCT066;PCP068,PCT068;` . |
| `jhdb.configuration.context.encoding` | string | CP297 | A codificação JHDB (Java Hierarchical Database). Espera uma string de codificação válida`CP1047`,`IBM930`,`ASCII`, |
| `jhdb.metadata.extrapath` | string | arquivo:. /configuração/ | Um parâmetro de configuração que especifica uma pasta raiz extra específica do tempo de execução para as pastas psbs e dbds. |

| Chave | Tipo | Valor padrão | Descrição |
| --- | --- | --- | --- |
| `jhdb.checkpointPersistence` | string | nenhuma | O modo de persistência do ponto de verificação. Os valores permitidos sãonone /add /end. Use add para manter os pontos de verificação quando um novo for criado e adicionado ao registro. Use o ponto end de verificação para persistir no desligamento do servidor. Quaisquer outros valores desativam a persistência. Observe que sempre que um novo ponto de verificação for adicionado ao registro, todos os pontos de verificação existentes serão serializados e o arquivo será apagado. Não é um acréscimo aos dados existentes no arquivo. Portanto, dependendo do número de pontos de verificação, isso pode ter alguns efeitos no desempenho. |

| Chave | Tipo | Valor padrão | Descrição |
| --- | --- | --- | --- |
| `jhdb.checkpointPath` | string | arquivo:. /configuração/ | Se `jhdb.checkpointPersistence` não for `none`, esse parâmetro permitirá que você configure o caminho de persistência do ponto de verificação (local de armazenamento do arquivo checkpoint.dat); todos os dados de pontos de verificação contidos no registro são serializados e armazenados em um arquivo (checkpoint.dat) localizado na pasta fornecida. Observe que somente os dados do ponto de verificação (ScriptID, StepID, posição do banco de dados e área do ponto de verificação) são afetados por esse backup. |
| `jhdb.navigation.cachenexts` | número | 5000 | A duração do cache (em milissegundos) usada na navegação hierárquica para um RDBMS. |

| Chave | Tipo | Valor padrão | Descrição |
| --- | --- | --- | --- |
| `jhdb.use-db-prefix` | booliano | true | Especifica se um prefixo de banco de dados deve ser ativado na navegação hierárquica para um RDBMS. |
| `jhdb.query.limitJoinUsage` | booliano | true | Especifica se o parâmetro limite de uso de junção deve ser usado em gráficos RDBMS. |
| `taskExecutor.corePoolSize` | número | 5 | Quando uma transação em um terminal é iniciada por meio de um script groovy, um novo tópico é criado. Use esse parâmetro para configurar o tamanho do pool principal. |
| `taskExecutor.maxPoolSize` | número | 10 | Quando uma transação em um terminal é iniciada por meio de um script groovy, um novo tópico é criado. Use esse parâmetro para configurar o tamanho máximo do pool (número máximo de threads paralelos). |

| Chave | Tipo | Valor padrão | Descrição |
| --- | --- | --- | --- |
| `taskExecutor.queueCapacity` | número | 50 | Quando uma transação em um terminal é iniciada por meio de um script groovy, um novo tópico é criado. Use esse parâmetro para configurar o tamanho da fila. (= número máximo de transações pendentes quando `taskExecutor.maxPoolSize` atingido) |
| `taskExecutor.allowCoreThreadTimeOut` | boolean | false | Especifica se os encadeamentos principais devem atingir o tempo limite no JCIS. Isso permite o crescimento e a redução dinâmicos, mesmo em combinação com uma fila diferente de zero (já que o tamanho máximo do pool só aumentará quando a fila estiver cheia). |

| Chave | Tipo | Valor padrão | Descrição |
| --- | --- | --- | --- |
| `jics.runUnitLauncherPool.enable` | boolean | false | Especifica se o pool do lançador de unidades de execução deve ser ativado no JICS. |
| `jics.runUnitLauncherPool.size` | número | 20 | O tamanho do pool do lançador da unidade de execução no JICS. |
| `jics.runUnitLauncherPool.validationInterval` | número | 1000 | O intervalo de validação do pool do lançador de unidades de execução no JICS, expresso em milissegundos. |
| `spring.aws.application.credentials` | string | nulo | Carregue as credenciais da AWS do arquivo de perfis de credenciais no JICS. |
| `jics.queues.sqs.region` | string | eu-west-1 | A região da AWS para o AWS Simple Queue Service, usada no JICS. |
| `mq.queues.sqs.region` | string | eu-west-3 | A região da AWS para o serviço AWS SQS MQ. |

| Chave | Tipo | Valor padrão | Descrição |
| --- | --- | --- | --- |
| `quartz.scheduler.stand-by-if-error` | boolean | false | Especifica se a execução do trabalho deve ser acionado se o agendador de trabalhos estiver no modo de espera. Se verdadeiro, quando ativada, a execução do trabalho não é acionada. |
| `databaseStatistics` | boolean | false | Especifica se devem permitir que os construtores de SQL coletem e exibam informações estatísticas. |
| `dbDateFormat` | string | aaaa-MM-dd | O formato da data alvo do banco de dados. |
| `dbTimeFormat` | string | HH:mm:ss | O formato de hora alvo do banco de dados. |
| `dbTimestampFormat` | string | aaaa-MM-dd HH:mm:ss.SSS | O formato de carimbo de data/hora de destino do banco de dados. |

| Chave | Tipo | Valor padrão | Descrição |
| --- | --- | --- | --- |
| `dateTimeFormat` | string | ISO | dateTimeFormat Descreve como inserir o tipo de data e hora do banco de dados em entidades simplificadoras de dados. Os valores permitidos sãoISO /EUR /EUR /USA /LOCAL |
| `localDateFormat` | string | | Lista de formatos de data locais. Separe cada formato com\. |
| `localTimeFormat` | string | | Lista de formatos de horário local. Separe cada formato com\ |
| `localTimestampFormat` | string | | Lista de formatos de carimbo de data/hora locais. Separe cada formato com\. |
| `pgmDateFormat` | string | aaaa-MM-dd | O formato de data e hora. |
| `pgmTimeFormat` | string | HH.MM.SS | O formato de hora usado para execução de pgm (programas). |
| `pgmTimestampFormat` | string | aaa-mm-dd-hh.mm.ss.ssssss | O formato do registro de data e hora. |

| Chave | Tipo | Valor padrão | Descrição |
| --- | --- | --- | --- |
| `cacheMetadata` | booliano | true | Especifica se os metadados do banco de dados devem ser armazenados em cache. |
| `forceDisableSQLTrimStringType` | boolean | false | Especifica se o corte de todos os parâmetros da string sql deve ser desativado. |
| `fetchSize` | número | | O valor fetchSize para cursores. Use ao buscar dados usando fragmentos por meio de utilitários de carregamento/descarregamento. |
| `check-groovy-file` | booliano | true | Especifica se o conteúdo dos arquivos groovy deve ser verificado antes do registro. |
| `qtemp.uuid.length` | número | 9 | O comprimento de identificação exclusivo do QTEMP. |
| `qtemp.dblog` | boolean | false | Se deve habilitar o registro do banco de dados QTEMP. |

| Chave | Tipo | Valor padrão | Descrição |
| --- | --- | --- | --- |
| `qtemp.cleanup.threshold.hours` | número | 0 | Para especificar quando `qtemp.dblog` está ativado. A vida útil da partição db (em horas). |
| `sort.function` | string | | O nome da função de classificação para o banco de dados blu4iv. |

## Propriedades disponíveis para aplicativos web opcionais

Dependendo do seu aplicativo modernizado, talvez seja necessário configurar um ou mais aplicativos web opcionais que representem suporte para dependências como z/OS, AS/400 ou IMS/MFS. As tabelas a seguir contêm listas dos parâmetros de chave/valor disponíveis para configurar cada aplicativo web opcional.

gapwalk-utility-pgm.guerra

Esse aplicativo web opcional contém suporte para programas utilitários do Z/OS.

Esta tabela fornece uma visão exaustiva dos parâmetros-chave/valores desse aplicativo.

| Chave | Tipo | Valor padrão | Descrição |
| --- | --- | --- | --- |
| `logging.config` | Path | caminho de classe: logback-utility.xml | Chave padrão para a referência ao arquivo de configuração de logback. Outras chaves de registro padrão também estão disponíveis. |
| `spring.jta.enabled` | boolean | false | Chave Padrão. Se o modo de suporte |

| Chave | Tipo | Valor padrão | Descrição |
| --- | --- | --- | --- |
| | | | da fonte de dados não for static-xa , a configuração automática das transações do Spring JTA deverá ser desativada. |
| `spring.datasource.primary.jndi-name` | string | jdbc/primary | O nome jndi (Java Naming And Directory Interface) da fonte de dados primária, se estiver usando JNDI. |
| `primary.datasource + -driver-class-name + -url + -username + -password` | Fonte de dados Spring padrão com subchaves | | Contém as informaçõ es de conexão do banco de dados do aplicativo, se não estiver usando o JNDI. Deve ter a mesma configuração do arquivo yml do aplicativo moderniza do.<br><br>Como alternativa, o uso de segredos da AWS é fortemente encorajado, conforme explicado em xref:.. / configuration/con figuration.adoc [Configuração]. |

| Chave | Tipo | Valor padrão | Descrição |
| --- | --- | --- | --- |
| `encoding` | string | ASCII | A codificação usada em programas utilitários. Espera uma codificação válida`CP1047,IBM930,ASCII,` |
| `sysPunchEncoding` | string | ASCII | O conjunto de caracteres de codificação syspunch. Espera uma codificação válida`CP1047,IBM930,ASCII,` |
| `zonedMode` | string | EBCDIC_STRICT | O modo para codificar ou decodificar tipos de dados zoneados. Os valores permitidos são`EBCDIC_STRICT /EBCDIC_MODIFIED /AS400`. |
| `unload.chunkSize` | número | 0 | Tamanho do pedaço usado para o utilitário de descarga. |
| `unload.sqlCodePointShift` | número | 0 | O utilitário de mudança de pontos do código SQL para descarga. Executa o processo de mudança de caracteres. Obrigatório quando seu banco de dados de destino do DB2 é o Postgresql. |

| Chave | Tipo | Valor padrão | Descrição |
| --- | --- | --- | --- |
| `unload.columnFiller` | string | space | O preenchedor de colunas do utilitário de descarga. |
| `unload.varCharIsNull` | boolean | false | Use esse parâmetro no programa INFTILB, se definido como `true`, todos os campos não anuláveis com valores em branco (espaço) retornarão uma string vazia. |
| `unload.useDatabaseConfiguration` | boolean | false | Especifica se a configuração de data ou hora do application-main.yml deve ser usada no utilitário de descarregamento. |
| `unload.format.date` | string | MM/dd/aaaa | Se `unload.useDatabaseConfiguration` estiver ativado, o formato de data a ser usado no utilitário de descarga. |

| Chave | Tipo | Valor padrão | Descrição |
| --- | --- | --- | --- |
| `unload.format.time` | string | HH.MM.SS | Se `unloadUseDatabaseConfiguration` estiver ativado, o formato de hora a ser usado no utilitário de descarga. |
| `unload.format.timestamp` | string | aaa-mm-dd-hh.mm.ss.ssssss | Se `unloadUseDatabaseConfiguration` estiver ativado, o formato de carimbo de data/hora a ser usado no utilitário de descarga. |
| `unload.nbi.whenNull` | hexadecimais | 6F | O valor do Indicador de Byte Nulo (nbi) a ser adicionado quando o valor do banco de dados for nulo. |
| `unload.nbi.whenNotNull` | hexadecimais | 00 | O valor do Indicador de Byte Nulo (nbi) a ser adicionado quando o valor do banco de dados não for nulo. |

| Chave | Tipo | Valor padrão | Descrição |
| --- | --- | --- | --- |
| `unload.nbi.writeNullIndicator` | boolean | false | Especifica se o indicador nulo deve ser gravado no arquivo de saída de descarga. |
| `unload.fetchSize` | número | 0 | Permite ajustar o tamanho da busca ao manipular cursores no utilitário de descarregamento. |
| `treatLargeNumberAsInteger` | boolean | false | Especifica se os números grandes devem ser tratados como`Integer`. Eles são tratados como `BigDecimal` padrão. |
| `load.batchSize` | número | 0 | O tamanho do lote do utilitário de carga. |
| `load.format.localDate` | string | dd.mm.aaaa\ dd/mm/aaaa\ aaa-mm-dd | O formato de data local do utilitário de carregamento a ser usado. |
| `load.format.localTime` | string | HH:mm:ss\ HH.mm.ss | O formato de hora local do utilitário de carregamento a ser usado. |
| `load.format.dbDate` | string | aaaa-MM-dd | O formato do banco de dados do utilitário de carga a ser usado. |

| Chave | Tipo | Valor padrão | Descrição |
| --- | --- | --- | --- |
| `load.format.dbTime` | string | HH:mm:ss | O tempo de uso do banco de dados do utilitário de carregamento. |
| `load.sqlCodePointShift` | número | 0s | A mudança de pontos do código SQL para o utilitário de carregamento. Executa o processo de mudança de caracteres. Obrigatório quando seu banco de dados de destino do DB2 é o Postgresql. |
| `forcedDate` | string | | Força a data até a data fornecida, se houver uma. |
| `frozenDate` | booliano | true | Especifica se a data deve ser congelada. Aplica-se somente se `forcedDate` também estiver definido. |

| Chave | Tipo | Valor padrão | Descrição |
| --- | --- | --- | --- |
| `jcl.type` | string | mvs | Tipo de arquivo.jcl. Os valores permitidos são `jcl` / `vse`. Os comandos PRINT/ REPRO do utilitário IDCAMS retornam 4 se o arquivo estiver vazio para um jcl que não seja vse. |
| `hasGraphic` | boolean | false | Se o utilitário INFUTILB precisa lidar com colunas GRAPHIC DB2. |

gapwalk-cl-command.guerra

Esse aplicativo web opcional contém suporte para programas utilitários AS/400.

Esta tabela fornece uma visão exaustiva dos parâmetros-chave/valores desse aplicativo.

| Chave | Tipo | Valor padrão | Descrição |
| --- | --- | --- | --- |
| `logging.config` | Path | caminho de classe: logback-utility.xml | Chave padrão para a referência ao arquivo de configuração de logback. Outras chaves de registro padrão também estão disponíveis. |
| `spring.jta.enabled` | boolean | false | Chave Padrão. Se o modo de suporte da fonte de dados não for static-xa |

| Chave | Tipo | Valor padrão | Descrição |
| --- | --- | --- | --- |
| | | | , a configuração automática das transações do Spring JTA deverá ser desativada. |
| `spring.datasource.primary.jndi-name` | string | jdbc/primary | O nome jndi (Java Naming And Directory Interface) da fonte de dados primária, se estiver usando JNDI. |
| `primary.datasource + -driver-class-name + -url + -username + -password` | Fonte de dados Spring padrão com subchaves | | Contém as informações de conexão do banco de dados do aplicativo, se não estiver usando o JNDI. Deve ter a mesma configuração do arquivo yml do aplicativo modernizado.<br><br>Como alternativa, o uso de segredos da AWS é fortemente encorajado, conforme explicado em xref:.. / configuration/configuration.adoc [Configuração]. |

| Chave | Tipo | Valor padrão | Descrição |
| --- | --- | --- | --- |
| `encoding` | string | ASCII | A codificação usada em programas utilitários. Espera uma codificação válida`CP1047,IBM930,ASCII,` |
| `zonedMode` | string | EBCDIC_STRICT | O modo para codificar ou decodificar tipos de dados zoneados. Os valores permitidos são`EBCDIC_STRICT /EBCDIC_MODIFIED /AS400`. |
| `commands-off` | string | | Lista de comandos a serem desativados, separados por vírgula. Os valores permitidos são: `PGM_BASIC ,RCVMSG,SNDRCVF,CHGVAR,Q` e `SNDDTAQ`. Útil quando você deseja desativar ou substituir um programa existente. `PGM_BASIC` é um programa de velocidade específico projetado para fins de depuração. |

gapwalk-hierarchical-support.guerra

Esse aplicativo web opcional contém suporte a transações IMS/MFS.

Esta tabela fornece uma visão exaustiva dos parâmetros-chave/valores desse aplicativo.

| Chave | Tipo | Valor padrão | Descrição |
| --- | --- | --- | --- |
| `logging.config` | Path | caminho de classe: logback-utility.xml | Chave padrão para a referência ao arquivo de configuração de logback. Outras chaves de registro padrão também estão disponíveis. |
| `spring.jta.enabled` | boolean | false | Chave Padrão. Se o modo de suporte da fonte de dados não for static-xa, a configuração automática das transações do Spring JTA deverá ser desativada. |
| `jhdb.configuration.context.encoding` | string | | A codificação JHDB (Java Hierarchical Database). Espera uma string de codificação válida`CP1047`,`IBM930`,`ASCII`, |
| `jhdb.checkpointPersistence` | string | nenhuma | O modo de persistência do ponto de verificação. Os valores permitidos são`none` /`add` /`end`. Use `add` para manter os pontos de verificação quando um novo for criado e adicionado ao registro. Use o |

| Chave | Tipo | Valor padrão | Descrição |
| --- | --- | --- | --- |
| | | | ponto end de verificação para persistir no desligamento do servidor. Quaisquer outros valores desativam a persistência. Observe que sempre que um novo ponto de verificação for adicionado ao registro, todos os pontos de verificação existentes serão serializados e o arquivo será apagado. Não é um acréscimo aos dados existentes no arquivo. Portanto, dependendo do número de pontos de verificação, isso pode ter alguns efeitos no desempenho. |

## Configurar autenticação para aplicativos do Gapwalk

Esta seção descreve como configurar a autenticação OAuth2 para aplicativos Gapwalk usando um Provedor de Identidade (IdP), como Cognito, Azure AD, Keycloak etc.

Tópicos

- Pré-requisito
- Configuração do Amazon Cognito
- Integração do Amazon Cognito no aplicativo Gapwalk

## Pré-requisito

Neste tutorial, usaremos o Amazon Cognito como IdP e o PlanetDemo como o projeto modernizado.

Você pode usar qualquer outro provedor de identidade externo. As ClientRegistration informações devem ser obtidas do seu IdP e são necessárias para a autenticação do gapwalk. Para obter mais informações, consulte a documentação oficial do seu IdP.

As ClientRegistration informações:

ID do cliente

: o id do ClientRegistration, em nosso exemplo, será PlanetDemo.

segredo do cliente

: o segredo de seu cliente.

endpoint de autorização

: O URI do endpoint de autorização para o servidor de autorização.

endpoint de token

: O URI do ponto final do token para o servidor de autorização.

endpoint jwks

: O URI usado para obter a chave web JSON (JWK) contendo as chaves para validar a assinatura web JSON emitida pelo servidor de autorização.

URI de redirecionamento

: URI para o qual o servidor de autorização redireciona o usuário final se o acesso for concedido.

## Configuração do Amazon Cognito

Primeiro, criaremos e configuraremos um grupo de usuários e usuários do Amazon Cognito que usaremos com nosso aplicativo gapwalk implantado para fins de teste.

> **Note**
>
> Se você estiver usando outro IdP, poderá pular esta etapa.

Criar grupo de usuários

1. Acesse o Amazon Cognito em AWS Management Console e autentique-se usando suas credenciais. AWS
2. Escolha Grupos de usuários.
3. Selecione Criar um pool de usuários.
4. Em Configurar a experiência de login, mantenha o tipo de provedor padrão do grupo de usuários do Cognito. Você pode escolher uma ou várias opções de login do grupo de usuários do Cognito; por enquanto, escolha Nome do usuário e escolha Próximo.
5. Em Configurar requisitos de segurança, mantenha os padrões e desative a autenticação multifator escolhendo Sem MFA e escolha Avançar.
6. Desative a opção Ativar autorregistro como medida de segurança e escolha Próximo.
7. Escolha Enviar e-mail com o Cognito. Escolha Próximo.
8. Em Integrar seu aplicativo, escolha um nome para seu grupo de usuários. Nas páginas de autenticação hospedada, escolha Usar a interface do usuário hospedada do Cognito.
9. Para simplificar, em Domínio, escolha Usar um domínio do Cognito e insira um prefixo de domínio; por exemplo, `https://planetsdemo` O aplicativo de demonstração deve ser adicionado como cliente.
    1. Em Cliente inicial do aplicativo, escolha Cliente confidencial. Insira um nome de cliente do aplicativo, por exemplo `` `planetsdemo` `` e escolha Gerar um segredo do cliente.
    2. Em URL de retorno de chamada permitida, insira a URL para a qual redirecionar o usuário após a autenticação. Um URL http://localhost:8080/planetsdemo temporário também pode ser usado e editado posteriormente.
    3. Mantenha os valores padrão nas configurações avançadas do cliente de aplicativo e nas seções Permissões de leitura e gravação de atributos.
    4. Escolha Próximo.
10. Em Revisar e criar, verifique suas escolhas e selecione Criar grupo de usuários.

Para mais informações, consulte Criar um grupo de usuários.

Criação de usuário

Como o autorregistro está desativado, crie um usuário do Amazon Cognito. Navegue até o Amazon Cognito no AWS Management Console. Escolha o grupo de usuários que você criou e, em Usuários, escolha Criar usuário.

Em Informações do usuário, escolha Enviar um convite por e-mail, insira um nome de usuário e um endereço de e-mail e escolha Gerar uma senha. Escolha Criar usuário.

## Integração do Amazon Cognito no aplicativo Gapwalk

Agora que seu grupo de usuários e usuários do Amazon Cognito estão prontos, acesse o `main-application.yml` arquivo do seu aplicativo modernizado e adicione o seguinte código:

```
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
```

Substitua os seguintes espaços reservados conforme descrito:

1. Acesse o Amazon Cognito em AWS Management Console e autentique-se usando suas credenciais. AWS
2. Escolha Grupos de usuários e escolha o grupo de usuários que você criou. Você pode encontrar seu *`ID do pool`* em ID do grupo de usuários.
3. Escolha a integração de aplicativos, onde você pode encontrar sua *`your-cognito-domain`*, acesse clientes e análises de aplicativos e escolha seu aplicativo.
4. Em Cliente do aplicativo: YourApp, você pode encontrar o *`nome do cliente`*, o *`ID do cliente`* e o *`segredo do cliente`* (Mostrar segredo do cliente).
5. *`region-id`* corresponde ao ID da região em que você criou seu grupo de usuários e usuários do Amazon Cognito `eu-west-3`, por exemplo.
6. Para redirect-uri, insira o URI para o qual redirecionar o usuário após a autenticação.

Você pode implantar seu aplicativo Gapwalk agora e usar o usuário criado anteriormente para fazer login no seu aplicativo.

> **Note**
>
> Se durante o login você receber um erro com a exceção “Atributo ausente 'cognito:username' nos atributos”, remova a seguinte linha:: cognito:username user-name-attribute

# AWS APIs de tempo de execução do Blu Age

O AWS Blu Age Runtime usa vários aplicativos da web para expor endpoints REST, fornecendo maneiras de interagir com os aplicativos modernizados usando clientes REST (por exemplo, chamando trabalhos usando um agendador).

O objetivo deste documento é listar os endpoints REST disponíveis, fornecendo detalhes sobre:

- O papel deles
- A maneira de usá-los adequadamente

A lista de endpoints é organizada em categorias, dependendo da natureza do serviço fornecido e do aplicativo web que expõe os endpoints.

Presumimos que você já tenha um conhecimento básico do uso de endpoints REST usando ferramentas dedicadas, como POSTMAN, Thunder Client, CURL, navegadores da web, etc...) ou escrever seu próprio trecho de código para fazer uma chamada de API.

Tópicos



## Criação de URLs

Cada aplicativo da web abaixo está definindo um caminho raiz, compartilhado por todos os endpoints. Em seguida, cada endpoint adiciona seu próprio caminho dedicado. O URL resultante a ser usado é o resultado da concatenação dos caminhos. Por exemplo, considerando o primeiro endpoint do aplicativo Gapwalk, temos:

- `/gapwalk-application` para o caminho raiz do aplicativo web.
- `/scripts` para o caminho de endpoint dedicado.

O URL resultante a ser usado será `http://server:port/gapwalk-application/scripts`

servidor

aponta para o nome do servidor (aquele que hospeda o determinado aplicativo web).

porta

a porta exposta pelo servidor.

## Aplicação Gapwalk

Os endpoints do aplicativo web Gapwalk usam o caminho raiz `/gapwalk-application`

Tópicos



## Endpoints relacionados a trabalhos em lote (JCLs modernizados e similares)

Os trabalhos em lote podem ser executados de forma síncrona ou assíncrona (veja os detalhes abaixo). Os trabalhos em lote estão sendo executados usando scripts groovy que são o resultado da modernização dos scripts legados (JCL).

Tópicos



### Listar scripts implantados

- Método suportado: GET
- Caminho: `/scripts`
- Argumentos: nenhum

- Esse endpoint retorna a lista de scripts groovy implantados no servidor, como uma string. Esse endpoint deve ser usado principalmente em um navegador da Web, já que a String resultante é uma página HTML, com links ativos (um link por script iniciável -- veja o exemplo abaixo).

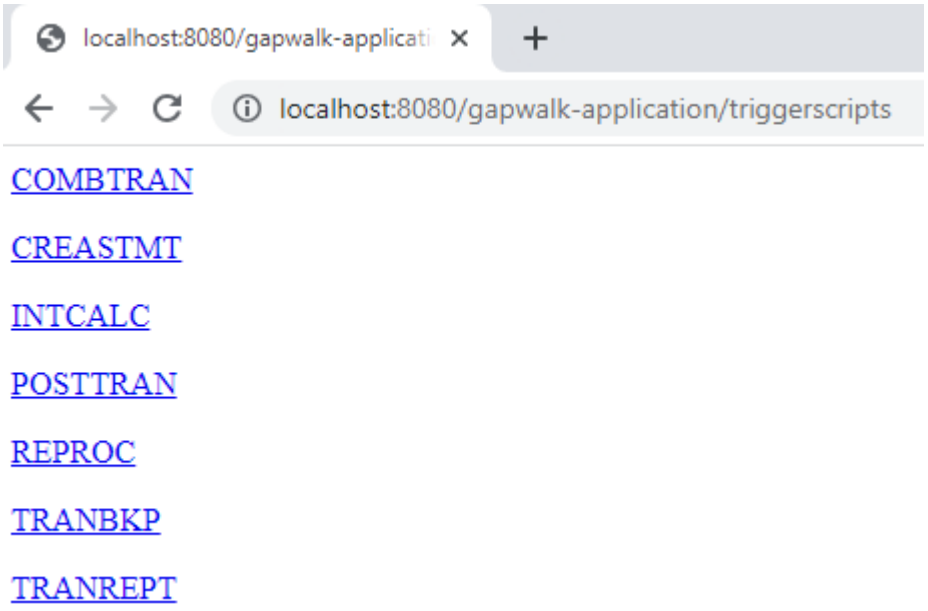
Resposta de exemplo:

```
<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**
>
> Os links representam o URL a ser usado para iniciar cada script listado de forma síncrona.

- Método suportado: GET
- Caminho: `/triggerscripts`
- Argumentos: nenhum
- Esse endpoint retorna a lista de scripts groovy implantados no servidor, como uma string. Esse endpoint deve ser usado principalmente em um navegador da Web, já que a String resultante é uma página HTML, com links ativos (um link por script iniciável -- veja o exemplo abaixo).

  Ao contrário da resposta anterior do endpoint, os links representam o URL a ser usado para iniciar cada script listado de forma assíncrona.




Inicie um script de forma síncrona

Esse endpoint tem duas variantes com caminhos dedicados para uso de GET e POST (veja abaixo).

- Método suportado: GET
- Caminho: `/script/{scriptId:.+}`
- Método suportado: POST
- Caminho: `/post/script/{scriptId:.+}`
- Argumentos:
  - identificador do script a ser lançado
  - opcionalmente: parâmetros a serem passados para o script, usando parâmetros de solicitação (vistos como a`Map<String,String>`). Os parâmetros fornecidos serão adicionados automaticamente às ligações do script groovy invocado.
- A chamada iniciará o script com o identificador fornecido, usando parâmetros extras, se fornecidos, e aguardará a conclusão da execução do script antes de retornar uma mensagem (`String`) que será:
  - “Concluído.” (se a execução do trabalho ocorreu sem problemas).
  - Uma mensagem de erro JSON com detalhes sobre o que deu errado durante a execução do trabalho. Mais detalhes podem ser recuperados dos registros do servidor para entender o que deu errado com a execução do trabalho.

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

Analisando os registros do servidor, podemos descobrir que esse é um problema de implantação (o programa esperado não foi implantado corretamente e, portanto, não pode ser encontrado, fazendo com que a execução do trabalho falhe):

```
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**
>
> As chamadas síncronas devem ser reservadas para trabalhos de curta duração. Trabalhos de longa duração devem ser iniciados de forma assíncrona (consulte o endpoint dedicado abaixo).

Inicie um script de forma assíncrona

- Métodos suportados: GET/POST
- Caminho: `/triggerscript/{scriptId:.+}`
- arguments:
  - identificador do script a ser lançado
  - opcionalmente: parâmetros a serem passados para o script, usando parâmetros de solicitação (vistos como a`Map<String,String>`). Os parâmetros fornecidos serão adicionados automaticamente ao https://docs.groovy-lang.org/latest/html/api/groovy/lang/Binding.html[bindings] do script groovy invocado.

- Ao contrário do modo síncrono acima, o endpoint não espera a conclusão da execução do trabalho para enviar uma resposta. A execução do trabalho é iniciada de uma só vez, se for possível encontrar um thread disponível para fazer isso, e uma resposta é enviada imediatamente ao chamador, com o id de execução do trabalho, um identificador exclusivo que representa a execução do trabalho, que pode ser usado para consultar o status da execução do trabalho ou forçar o encerramento de uma execução de trabalho que deveria estar com defeito. O formato da resposta é:

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

- Como a execução assíncrona do trabalho depende de um número fixo limitado de encadeamentos, a execução do trabalho pode não ser iniciada se nenhum encadeamento disponível for encontrado. Nesse caso, a mensagem retornada se parecerá com:

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

  Consulte o `settriggerthreadlimit` endpoint abaixo para saber como aumentar o limite de segmentos.

Resposta de exemplo:

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

O identificador exclusivo de execução de tarefas permite recuperar rapidamente as entradas de registro relacionadas nos registros do servidor, se necessário. Ele também é usado por vários outros endpoints detalhados abaixo.

Listando scripts acionados

- Métodos suportados: GET
- Caminhos: `/triggeredscripts/{status:.+}`, `/triggeredscripts/{status:.+}/{namefilter}`
- Argumentos:
  - Status (obrigatório): o status dos scripts acionados a serem recuperados. Os valores possíveis são:

  - `all`: mostre todos os detalhes da execução do trabalho, independentemente de os trabalhos ainda estarem em execução ou não.
  - `running`: mostra somente os detalhes dos trabalhos que estão sendo executados no momento.
  - `done`: mostra somente os detalhes dos trabalhos cuja execução acabou.
  - `killed`: mostre somente os detalhes dos trabalhos cuja execução foi encerrada à força usando o endpoint dedicado (veja abaixo).
  - `triggered`: mostra somente os detalhes dos trabalhos que foram acionados, mas ainda não foram lançados.
  - `failed`: mostra somente os detalhes dos trabalhos cuja execução foi marcada como falhada.
  - _namefilter (opcional) _: recupera somente execuções para o identificador de script fornecido.
- Retorna uma coleção de detalhes de execuções de trabalhos como JSON. Para ter mais informações, consulte Estrutura de mensagens do Job Execution Details.

Resposta de exemplo:

```
[
    {
      "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"
    }
  ]
```

Recuperando detalhes da execução do trabalho

- Método suportado: GET
- Caminho: `/getjobexecutioninfo/{jobexecutionid:.+}`
- Argumentos:

  - jobexecutionid (obrigatório): o identificador exclusivo de execução do trabalho para recuperar os detalhes correspondentes da execução do trabalho.
- Retorna: uma string JSON representando um único detalhe da execução do trabalho (consulte Estrutura de mensagens do Job Execution Details) ou uma resposta vazia se nenhum detalhe da execução do trabalho puder ser encontrado para o identificador fornecido.

Listando scripts lançados de forma assíncrona que podem ser eliminados

- Método suportado: GET
- Caminho: `/killablescripts`
- Retorna uma coleção de identificadores de execução de trabalhos que foram iniciados de forma assíncrona e que ainda estão em execução e podem ser eliminados à força (consulte o `/kill` endpoint abaixo).

Listando scripts lançados de forma síncrona que podem ser eliminados

- Método suportado: GET
- Caminho: `/killablesyncscripts`
- Retorna uma coleção de identificadores de execução de trabalhos que foram iniciados de forma síncrona, ainda estão em execução e podem ser eliminados à força (consulte o `/kill` endpoint abaixo).

Matando a execução de um determinado trabalho

- Método suportado: GET
- Caminho: `/kill/{identifier:.+}`
- argumento: identificador de execução do trabalho (obrigatório): o identificador exclusivo de execução do trabalho que aponta para que a execução do trabalho seja eliminada à força.
- Retorna: uma mensagem de texto detalhando o resultado da tentativa de eliminação da execução da tarefa; a mensagem conterá o identificador do script, o identificador exclusivo da execução da tarefa e a data e hora em que a eliminação da execução ocorreu. Se nenhuma execução de trabalho em execução for encontrada para o identificador fornecido, uma mensagem de erro será retornada em vez disso.

⚠ Warning

- O tempo de execução se esforça ao máximo para eliminar bem a execução do trabalho de destino. Portanto, a resposta do endpoint /kill pode levar um pouco de tempo para chegar ao chamador, pois o tempo de execução do AWS Blu Age tentará minimizar o impacto comercial da eliminação do trabalho.
- A eliminação forçada da execução de um trabalho não deve ser feita de ânimo leve, pois pode ter consequências comerciais diretas, incluindo possível perda ou corrupção de dados. Ele deve ser reservado para casos em que a execução de um determinado trabalho tenha ocorrido mal e os meios de remediação de dados estejam claramente identificados.
- A eliminação de um emprego deve levar a investigações adicionais (análise post-mortem) para descobrir o que deu errado e tomar as medidas corretivas adequadas.
- De qualquer forma, a tentativa de eliminar um trabalho em execução será registrada nos registros do servidor com mensagens de nível de aviso.

Listando os pontos de verificação existentes para capacidade de reinicialização

A reinicialização do trabalho depende da capacidade dos scripts de registrar pontos de verificação no `CheckpointRegistry` para rastrear o progresso da execução do trabalho. Se a execução de um trabalho não terminar corretamente e os pontos de verificação de reinicialização tiverem sido registrados, basta reiniciar a execução do trabalho a partir do último ponto de verificação registrado conhecido (sem precisar executar as etapas acima do ponto de verificação).

- Método suportado: GET
- Caminho: `/restarts`
- Retorna a lista de pontos de reinicialização existentes, que podem ser usados para reiniciar um trabalho cuja execução não chegou e terminou corretamente, como uma página html. Se nenhum ponto de verificação foi registrado por nenhum script, o conteúdo da página será “Nenhum ponto de verificação registrado”.

Reiniciando um trabalho (de forma síncrona)

- Método suportado: GET
- Caminho: `/restart/{hashcode}`

- Argumentos: hashcode (inteiro - obrigatório): reinicie uma execução de trabalho abortada anteriormente, usando o hashcode fornecido como valor do ponto de verificação (consulte o `/restarts` endpoint acima para saber como recuperar um valor de ponto de verificação válido).
- Devoluções: veja a descrição da `script` devolução acima.

Reiniciando um trabalho (de forma assíncrona)

- Método suportado: GET
- Caminho: `/triggerrestart/{hashcode}`
- Argumentos: hashcode (inteiro - obrigatório): reinicie uma execução de trabalho abortada anteriormente, usando o hashcode fornecido como valor do ponto de verificação (consulte o `/restarts` endpoint acima para saber como recuperar um valor de ponto de verificação válido).
- Devoluções: veja a descrição da `triggerscript` devolução acima.

Definindo o limite de threads para execuções de tarefas assíncronas

A execução assíncrona do trabalho depende de um pool dedicado de threads na JVM. Esse pool tem um limite fixo em relação ao número de threads disponíveis. O usuário tem a capacidade de ajustar o limite de acordo com as capacidades do host (número de CPUs, memória disponível, etc...). Por padrão, o limite de segmentos é definido como 5 segmentos.

- Método suportado: GET
- Caminho: `/settriggerthreadlimit/{threadlimit:.+}`
- Argumento (inteiro): o novo limite de encadeamento a ser aplicado. Deve ser um número inteiro estritamente positivo.
- Retorna uma mensagem (`String`) fornecendo o novo limite de encadeamento e o anterior, ou uma mensagem de erro se o valor limite de encadeamento fornecido não for válido (não é um número inteiro estritamente positivo).

Resposta de exemplo:

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

Contando as execuções de trabalhos acionadas atualmente em execução

- Método suportado: GET

- Caminho: `/countrunningtriggeredscripts`
- Retorna uma mensagem indicando o número de trabalhos em execução iniciados de forma assíncrona e o limite de threads (ou seja, o número máximo de trabalhos acionados que podem ser executados simultaneamente).

Resposta de exemplo:

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

> **Note**
>
> Isso pode ser usado para verificar, antes de iniciar um trabalho, se o limite de encadeamentos não foi atingido (o que impediria que o trabalho fosse iniciado).

Limpar informações sobre execuções de trabalhos

As informações de execução do trabalho permanecem na memória do servidor enquanto o servidor estiver ativo. Pode ser conveniente limpar as informações mais antigas da memória, pois elas não são mais relevantes; esse é o propósito desse endpoint.

- Método suportado: GET
- Caminho: `/purgejobinformation/{age:.+}`
- Argumentos: um valor inteiro estritamente positivo que representa a idade em horas das informações a serem eliminadas.
- Retorna uma mensagem com as seguintes informações:
  - Nome do arquivo de limpeza em que as informações de execução do trabalho eliminado estão sendo armazenadas para fins de arquivamento.
  - Número de informações de execução do trabalho eliminadas.
  - Número de informações restantes sobre a execução do trabalho no memorando

## Métricas para endpoints do

JVM

Esse endpoint retorna as métricas disponíveis relacionadas à JVM.

- Método suportado: GET
- Caminho: `/metrics/jvm`
- Argumentos: nenhum
- Retorna uma mensagem com as seguintes informações:
  - threadActiveCount: Número de segmentos ativos.
  - jvmMemoryUsed: Memória usada ativamente pela Máquina Virtual Java.
  - jvmMemoryMax: Memória máxima permitida para a Máquina Virtual Java.
  - jvmMemoryFree: Memória disponível que não está sendo usada atualmente pela Java Virtual Machine.

Sessão

Esse endpoint retorna métricas relacionadas às sessões HTTP abertas atualmente.

- Método suportado: GET
- Caminho: `/metrics/session`
- Argumentos: nenhum
- Retorna uma mensagem com as seguintes informações:
  - SessionCount: Número de sessões de usuário ativas atualmente mantidas pelo servidor.

Lote

- Método suportado: GET
- Caminho: `/metrics/batch`
- Argumentos:
  - startTimestamp (opcional, número): carimbo de data/hora inicial para filtragem de dados.
  - endTimestamp (opcional, número): carimbo de data/hora final para filtragem de dados.
  - página (opcional, número): Número da página para paginação.
  - PageSize (opcional, número): Número de itens por página na paginação.
- Retorna uma mensagem com as seguintes informações:
  - conteúdo: lista de métricas de execução em lote.
  - Número da página: número da página atual na paginação.
  - PageSize: Número de itens exibidos por página.

- TotalPages: Número total de páginas disponíveis.
- numberOfElements: Contagem de itens na página atual.
- last: bandeira booleana para a última página.
- primeiro: bandeira booleana para a primeira página.

TRANSACTION

- Método suportado: GET
- Caminho: `/metrics/transaction`
- Argumentos:
  - startTimestamp (opcional, número): carimbo de data/hora inicial para filtragem de dados.
  - endTimestamp (opcional, número): carimbo de data/hora final para filtragem de dados.
  - página (opcional, número): Número da página para paginação.
  - PageSize (opcional, número): Número de itens por página na paginação.
- Retorna uma mensagem com as seguintes informações:
  - conteúdo: lista de métricas de execução de transações.
  - Número da página: número da página atual na paginação.
  - PageSize: Número de itens exibidos por página.
  - TotalPages: Número total de páginas disponíveis.
  - numberOfElements: Contagem de itens na página atual.
  - last: bandeira booleana para a última página.
  - primeiro: bandeira booleana para a primeira página.

## Outros endpoints

Use esses endpoints para listar programas ou serviços registrados, descobrir o status de saúde e gerenciar transações do JICS.

Tópicos

- Lista de programas registrados
- Listando serviços registrados
- Status de integridade

- Lista de transações JICS disponíveis
- Inicie uma transação JICS
- Iniciar uma transação JICS (alternativa)

Lista de programas registrados

- Método suportado: GET
- Caminho: `/programs`
- Retorna a lista de programas registrados, como uma página html. Cada programa é designado pelo identificador principal do programa. Tanto os programas legados modernizados quanto os programas utilitários (IDCAMS, IEBGENER, etc...) estão sendo retornados na lista. Observe que os programas utilitários disponíveis dependerão dos aplicativos web de utilitários que foram implantados em seu servidor tomcat. Por exemplo, os programas de suporte do utilitário z/OS podem não estar disponíveis para ativos modernizados do iSeries, pois não são relevantes.

Listando serviços registrados

- Método suportado: GET
- Caminho: `/services`
- Retorna a lista de serviços de tempo de execução registrados, como uma página html. Os serviços fornecidos são fornecidos pelo tempo de execução do AWS Blu Age como utilitários, que podem ser usados, por exemplo, em scripts interessantes. Os serviços de carregamento Blusam (para criar conjuntos de dados Blusam a partir de conjuntos de dados legados) se enquadram nessa categoria.

Resposta de exemplo:

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

Status de integridade

- Método suportado: GET
- Caminho: `/`
- Retorna uma mensagem simples, indicando que o aplicativo gapwalk está funcionando (`Jics application is running.`)

Lista de transações JICS disponíveis

- Método suportado: GET
- Caminho: `/transactions`
- Retorna uma página html listando todas as transações JICS disponíveis. Isso só faz sentido para ambientes com elementos JICS (modernização de elementos antigos do CICS).

Resposta de exemplo:

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

Inicie uma transação JICS

- Métodos suportados: GET, POST
- Caminho: `/jicstransrunner/{jtrans:.+}`
- arguments:
  - Identificador de transação JICS (string, obrigatório): identificador da transação JICS a ser iniciada (8 caracteres no máximo)
  - obrigatório: dados de entrada adicionais para passar para a transação, como um mapa<String, Object>. O conteúdo desse mapa será usado para alimentar a COMMAREA que será consumida pela transação do JICS. O mapa pode ficar vazio se nenhum dado for necessário para executar a transação.
  - opcional: entradas de cabeçalhos HTTP, para personalizar o ambiente de execução da transação em questão. As seguintes chaves de cabeçalho estão sendo suportadas:
    - `jics-channel`: O nome do JICS CHANNEL a ser usado pelo programa que será lançado com o lançamento dessa transação.
    - `jics-container`: O nome do CONTÊINER JICS a ser usado para o lançamento dessa transação JICS.
    - `jics-startcode`: o STARTCODE (string, até 2 caracteres) a ser usado no início da transação JICS. Consulte STARTCODE para valores possíveis (navegue pela página).
    - `jicxa-xid`: O XID (estrutura XID do identificador de transação X/Open) de uma “transação global” (XA), iniciada pelo chamador, da qual participará o lançamento atual da transação JICS.

- Retorna: uma serialização `com.netfective.bluage.gapwalk.rt.shared.web.TransactionResultBean` JSON, representando o resultado do lançamento da transação JICS.

Para obter mais informações sobre os detalhes da estrutura, consulte Estrutura de resultados do lançamento da transação.

Iniciar uma transação JICS (alternativa)

- métodos suportados: GET, POST
- caminho: `/jicstransaction/{jtrans:.+}`
- arguments:

  Identificador de transação JICS (string, obrigatório)

  identificador da transação JICS a ser iniciada (8 caracteres no máximo)

  necessário: dados de entrada adicionais para passar para a transação, como um mapa<String, Object>

  O conteúdo desse mapa será usado para alimentar a COMMAREA que será consumida pela transação do JICS. O mapa pode ficar vazio se nenhum dado for necessário para executar a transação.

  opcional: entradas de cabeçalhos HTTP, para personalizar o ambiente de execução da transação em questão.

  As seguintes chaves de cabeçalho estão sendo suportadas:

  - `jics-channel`: O nome do JICS CHANNEL a ser usado pelo programa que será lançado com o lançamento dessa transação.
  - `jics-container`: O nome do CONTÊINER JICS a ser usado para o lançamento dessa transação JICS.
  - `jics-startcode`: o STARTCODE (string, até 2 caracteres) a ser usado no início da transação JICS. Para valores possíveis, consulte STARTCODE (navegue pela página).
  - `jicxa-xid`: O XID (estrutura XID do identificador de transação X/Open) de uma “transação global” (XA), iniciada pelo chamador, da qual participará o lançamento atual da transação JICS.
- retorna: uma serialização `com.netfective.bluage.gapwalk.rt.shared.web.RecordHolderBean` JSON,

representando o resultado do lançamento da transação JICS. Os detalhes da estrutura podem ser encontrados emEstrutura de resultados do registro de lançamento da transação.

## Endpoints relacionados ao Job Queues

As Filas de Tarefas são o suporte do AWS Blu Age para o mecanismo de envio de trabalhos do AS400. As filas de trabalhos são usadas no AS400 para executar trabalhos em grupos de threads específicos. Uma fila de trabalhos é definida por um nome e um número máximo de threads que corresponde ao número máximo de programas que podem ser executados simultaneamente nessa fila. Se mais trabalhos forem enviados na fila do que o número máximo de segmentos, os trabalhos aguardarão até que um tópico esteja disponível.

Para obter uma lista completa do status de um trabalho em uma fila, consulte. Possível status de trabalho em uma fila

As operações nas filas de trabalhos são realizadas por meio dos seguintes endpoints dedicados. Você pode invocar essas operações a partir do URL do aplicativo Gapwalk com o seguinte URL raiz:. `http://server:port/gapwalk-application/jobqueue`

Tópicos

- Listar filas disponíveis
- Iniciar ou reiniciar uma fila de trabalhos
- Envie um trabalho para lançamento
- Listar todos os trabalhos agendados
- Listar todos os trabalhos que estão “em espera”
- Listar todos os trabalhos ativos
- Listar todos os trabalhos que estão esperando para serem lançados
- Libere todos os trabalhos que estão “em espera”
- Libere todos os trabalhos que estão “em espera” para um determinado nome de trabalho
- Libere um determinado trabalho para um número de trabalho

### Listar filas disponíveis

- Método suportado: GET
- Caminho: `list-queues`

- Retorna a lista de filas disponíveis junto com seu status, como uma lista JSON de valores-chave.

Resposta de exemplo:

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

Os possíveis status de uma fila de trabalhos são:

EM ESPERA

a fila de trabalhos está esperando para ser iniciada.

STARTED

a fila de trabalhos está ativa e funcionando.

UNKNOWN

o status da fila de trabalhos não pode ser determinado.

Iniciar ou reiniciar uma fila de trabalhos

- Método suportado: POST
- Caminho: `/restart/{name}`
- Argumento: o nome da fila a ser iniciada/reiniciada, como uma string - obrigatório.
- O endpoint não retorna nada, mas depende do status http para indicar o resultado da operação de início/reinício:

  HTTP 200

  a operação de início/reinício correu bem: a fila de trabalhos fornecida agora está INICIADA.

  HTTP 404

  a fila de trabalhos não existe.

  HTTP 503

  ocorreu uma exceção durante a tentativa de iniciar/reiniciar (os registros do servidor devem ser inspecionados para descobrir o que deu errado).

Envie um trabalho para lançamento

- Método suportado: POST
- Argumento: obrigatório como corpo da solicitação, uma serialização JSON de um `com.netfective.bluage.gapwalk.rt.jobqueue.SubmitJobMessage` objeto. Para ter mais informações, consulte Enviar entrada de trabalho.
- Retornos: um JSON contendo o original `SubmitJobMessage` e um registro indicando se o trabalho foi enviado ou não.

Listar todos os trabalhos agendados

- Método suportado: GET
- Caminho: `list-jobs`
- Retorna: uma lista de todos os trabalhos agendados, como uma string JSON. Para obter um exemplo de resposta, consulte Lista de respostas de trabalhos agendados.

Listar todos os trabalhos que estão “em espera”

- Método suportado: GET
- Caminho: `list-jobs-hold`
- Retorna: uma lista de todos os trabalhos agendados, como uma string JSON. Para obter um exemplo de resposta, consulte Lista de respostas de vagas “em espera”.

Listar todos os trabalhos ativos

- Método suportado: GET
- Caminho: `list-jobs-active`
- Retorna: uma lista de todos os trabalhos com status ATIVO, como uma string JSON. A resposta é semelhante em estrutura à de Lista de respostas de trabalhos agendados.

Listar todos os trabalhos que estão esperando para serem lançados

- Método suportado: GET
- Caminho: `list-jobs-waiting`

- Retorna: uma lista de todos os trabalhos com o status EXECUTION_WAIT (trabalhos que estão aguardando a disponibilidade de um thread para lançamento), como uma string JSON. A resposta é semelhante em estrutura à de the section called “Lista de respostas de trabalhos agendados”.

Libere todos os trabalhos que estão “em espera”

- Método suportado: POST
- Caminho: `release-all`
- Retorna: uma mensagem indicando o resultado da operação de tentativa de liberação. Dois casos possíveis aqui:
  - HTTP 200 e uma mensagem “Todos os trabalhos foram lançados com sucesso!” se todos os trabalhos foram liberados com sucesso.
  - HTTP 503 e uma mensagem “Trabalhos não lançados. Ocorreu um erro desconhecido. Consulte o registro para obter mais detalhes” se algo deu errado com a tentativa de lançamento.

Libere todos os trabalhos que estão “em espera” para um determinado nome de trabalho

<job name, job number>Para um determinado nome de trabalho, vários trabalhos podem ser enviados, com números de trabalho diferentes (a unicidade de um trabalho executado é garantida por um casal). O endpoint tentará liberar todos os envios de trabalhos com o nome de trabalho fornecido, que estão “em espera”.

- Método suportado: POST
- Caminho: `/release/{name}`
- Argumentos: o nome do trabalho a ser procurado, como uma string. Obrigatório.
- Retorna: uma mensagem indicando o resultado da operação de tentativa de liberação. Dois casos possíveis aqui:
  - HTTP 200 e uma mensagem “Jobs in group <name>(<number of released jobs>) lançados com sucesso!” os trabalhos foram liberados com sucesso.
  - HTTP 503 e uma mensagem “Trabalhos no grupo <name>não lançados. Ocorreu um erro desconhecido. Consulte o registro para obter mais detalhes” se algo deu errado com a tentativa de lançamento.

Libere um determinado trabalho para um número de trabalho

<job name, job number>O endpoint tentará liberar o envio exclusivo do trabalho, que está "suspenso", para o casal em questão.

- Método suportado: POST
- Caminho: `/release/{name}/{number}`
- Argumentos:

  name

  o nome do trabalho a ser procurado, como uma string. Obrigatório.

  número

  o número do trabalho a ser procurado, como um número inteiro. Obrigatório.

  retorna

  uma mensagem indicando o resultado da operação de tentativa de liberação. Dois casos possíveis aqui:
  - HTTP 200 e uma mensagem "" Job <name/number> released with success!" se o trabalho foi lançado com sucesso.
  - HTTP 503 e uma mensagem "Job <name/number> not released. Ocorreu um erro desconhecido. Consulte o registro para obter mais detalhes" se algo deu errado com a tentativa de lançamento.

## Pontos de extremidade REST do Blusam Application Console

O Blusam Application Console é uma API projetada para simplificar o gerenciamento de conjuntos de dados VSAM modernizados. Os endpoints do aplicativo web Blusam usam o caminho raiz `/bac`

Tópicos

## Conjuntos de dados relacionados a endpoints

Use os seguintes endpoints para criar ou gerenciar um conjunto de dados específico.

Tópicos



### Criar um conjunto de dados

Criar um endpoint de conjunto de dados permite criar uma definição de conjunto de dados e requer autenticação.

- Métodos suportados: POST
- Caminho: `/api/services/rest/bluesamservice/createDataSet`
- Argumentos:

  name

  (obrigatório, string): o nome do conjunto de dados.

  type

  (obrigatório, string): o tipo de conjunto de dados. Os valores possíveis são: `ESDS`, `KSDS` `RRDS`.

  Tamanho do registro

  (opcional, string): tamanho máximo de cada registro do conjunto de dados.

  Comprimento fixo

  (opcional, booleano): indica se o tamanho dos registros é fixo.

compression

(opcional, booleano): indica se o conjunto de dados está compactado.

Habilitar cache

(opcional, booleano): indica se o armazenamento em cache está ativado para o conjunto de dados.

Chaves alternativas

(opcional, lista de chaves):

- deslocamento (obrigatório, número)
- comprimento (obrigatório, número)
- nome (obrigatório, número)

- Retorna um arquivo json representando o conjunto de dados recém-criado.

Exemplo de solicitação

```
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
}
```

Resposta de exemplo:

```
{
    "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
  }
```

### Carregar um arquivo

Esse endpoint permite fazer upload de arquivos para o servidor. O arquivo é armazenado em uma pasta temporária que corresponde a cada usuário específico. Esse endpoint deve ser usado sempre que for necessário carregar um arquivo. Isso requer autenticação.

- Métodos suportados: POST
- Caminho: `/api/services/rest/bluesamservice/upload`
- Argumentos:

  file

  (obrigatório, multipartes/dados de formulário): o arquivo a ser carregado.
- Retorna um booleano que reflete o status do carregamento

### Carregar um conjunto de dados

Depois que a definição do conjunto de dados for criada usando o endpoint de criação de conjunto de dados descrito anteriormente, você poderá carregar registros associados ao arquivo carregado em um conjunto de dados específico. Isso requer autenticação.

- Métodos suportados: POST
- Caminho: `/api/services/rest/bluesamservice/loadDataSet`
- Argumentos:

  name

  (obrigatório, string): o nome do conjunto de dados.
- Retorna o status da solicitação e do conjunto de dados carregado.

### Carregar um conjunto de dados de um bucket do Amazon S3

Carrega um conjunto de dados usando um arquivo listcat de um bucket do Amazon S3.

- Métodos suportados: GET
- Caminho: `` `/api/services/rest/bluesamservice/loadDataSetFromS3` ``
- Argumentos:

Localização de ListCAT Files3

(obrigatório, string): a localização do arquivo listcat no Amazon S3.

Localização dos arquivos do conjunto de dados 3

(obrigatório, string): a localização do arquivo do conjunto de dados do conjunto de dados no Amazon S3.

região

(obrigatório, string): o Amazon S3 Região da AWS onde os arquivos são armazenados.

- Retorna o conjunto de dados recém-criado

Exemplo de solicitação

```
/BAC/api/services/rest/bluesamservice/loadDataSetFromS3?region=us-
east-1&listcatFileS3Location=s3://bucket-name/listcat.json&datasetFileS3Location=s3://
bucket-name/dataset.DAT
```

Exportar um conjunto de dados para um bucket do Amazon S3

Exporta um conjunto de dados para o bucket do Amazon S3 especificado.

- Métodos suportados: GET
- Caminho: `/api/services/rest/bluesamservice/exportDataSetToS3`
- Argumentos:

  s3Location

  (obrigatório, string): o local do Amazon S3 para o qual exportar o conjunto de dados.

  datasetName

  (obrigatório, string): o nome do conjunto de dados a ser exportado.

  região

  (obrigatório, string): o Região da AWS do bucket do Amazon S3.
- Retorna o conjunto de dados exportado

Exemplo de solicitação

```
/BAC/api/services/rest/bluesamservice/exportDataSetToS3?region=eu-
west-1&s3Location=s3://bucket-name/dump&datasetName=dataset
```

### Limpar um conjunto de dados

Limpa todos os registros de um conjunto de dados. Isso requer autenticação.

- Métodos suportados: POST
- Caminho: `/api/services/rest/bluesamservice/clearDataSet`
- Argumentos:

  name

  (obrigatório, string): o nome do conjunto de dados a ser limpo.
- Retorna o status da solicitação.

### Excluir um conjunto de dados

Exclui a definição e os registros do conjunto de dados. Isso requer autenticação.

- Métodos suportados: POST
- Caminho: `/api/services/rest/bluesamservice/deleteDataSet`
- Argumentos:

  name

  (obrigatório, string): o nome do conjunto de dados a ser excluído.
- Retorna o status da solicitação e do conjunto de dados excluído.

### Contar registros do conjunto de dados

Esse endpoint retorna o número de registros associados a um conjunto de dados. Isso requer autenticação.

- Métodos suportados: POST
- Caminho: `/api/services/rest/bluesamservice/countRecords`
- Argumentos:

name

(obrigatório, string): o nome do conjunto de dados.

- Retornos: o número de registros

## Conjuntos de dados em massa relacionados a endpoints

Use os seguintes endpoints para criar ou gerenciar vários conjuntos de dados ao mesmo tempo.

Tópicos

- Exportar conjuntos de dados
- Criar vários conjuntos de dados
- Listar todos os conjuntos de dados
- Listar diretamente todos os conjuntos de dados
- Excluir todos os conjuntos de dados
- Obtenha as definições do conjunto de dados do arquivo listcat
- Obtenha as definições do conjunto de dados do arquivo cat da lista carregado
- Carregar listcat do arquivo json

Exportar conjuntos de dados

- Métodos suportados: GET
- Caminho: `/api/services/rest/bluesamservice/exportDataSet`
- Argumentos:

  name

  (obrigatório, string): o nome do conjunto de dados a ser excluído.

  datasetOutputFile

  (opcional, string): o caminho onde armazenar o conjunto de dados exportado no servidor

  vermelho

  (opcional, booleano): os campos RDW devem ser exportados.
- retorna o status da solicitação e do arquivo contendo o conjunto de dados exportado.

Criar vários conjuntos de dados

- Métodos suportados: POST
- Caminho: `/api/services/rest/bluesamservice/createAllDataSets`
- Argumentos:
  - Lista de conjuntos de dados

    name

    (obrigatório, string): o nome do conjunto de dados.

    type

    (obrigatório, string): o tipo de conjunto de dados. Os valores possíveis são: `ESDS`, `KSDS` `RRDS`.

    Tamanho do registro

    (opcional, string): tamanho máximo de cada registro do conjunto de dados.

    Comprimento fixo

    (opcional, booleano): indica se o tamanho dos registros é fixo.

    compression

    (opcional, booleano): indica se o conjunto de dados está compactado.

    Habilitar cache

    (opcional, booleano): indica se o armazenamento em cache está ativado para o conjunto de dados.
- Retornos: o status da solicitação e o conjunto de dados recém-criado.

Listar todos os conjuntos de dados

- Métodos suportados: GET
- Caminho: `/api/services/rest/bluesamservice/listDataSet`
- Argumentos: nenhum
- Retornos: o status da solicitação e a lista dos conjuntos de dados.

Listar diretamente todos os conjuntos de dados

- Métodos suportados: GET
- Caminho: `/api/services/rest/bluesamservice/directListDataSet`
- Argumentos: nenhum
- Retornos: o status da solicitação e a lista dos conjuntos de dados.

Excluir todos os conjuntos de dados

- Métodos suportados: POST
- Caminho: `/api/services/rest/bluesamservice/removeAll`
- Argumentos: nenhum
- Retorna: um booleano representando o status da solicitação.

Obtenha as definições do conjunto de dados do arquivo listcat

- Métodos suportados: POST
- Caminho: `/api/services/rest/bluesamservice/getDataSetsDefinitionFromListcat`
- Argumentos:

  paramFilePath

  (obrigatório, string): o caminho para o arquivo listcat.
- Retornos: uma lista de conjuntos de dados

Obtenha as definições do conjunto de dados do arquivo cat da lista carregado

- Métodos suportados: POST
- Caminho: `` `/api/services/rest/bluesamservice/getDataSetsDefinitionFromUploadedListcat` ``
- Argumentos: nenhum
- Retornos: uma lista de conjuntos de dados

Carregar listcat do arquivo json

- Métodos suportados: GET
- Caminho: `/api/services/rest/bluesamservice/loadListcatFromJsonFile`
- Argumentos:

  filePath

  (obrigatório, string): o caminho para o arquivo listcat.
- Retornos: uma lista de conjuntos de dados

## Registros

Use os seguintes endpoints para criar ou gerenciar registros em um conjunto de dados.

Tópicos



Criar um registro

Esse endpoint permite criar um novo registro. Isso requer autenticação.

- Métodos suportados: POST
- Caminho: `/api/services/rest/crud/createRecord`
- Argumentos:

  conjunto de dados

  (obrigatório, DataSet): o objeto do conjunto de dados

  máscara

  (obrigatório, máscara): o objeto da máscara.
- Retorna o status da solicitação e o registro criado.

### Leia um conjunto de dados

Esse endpoint permite ler um conjunto de dados.

- Métodos suportados: GET
- Caminho: `/api/services/rest/crud/readDataSet`
- Argumentos:

  conjunto de dados

  (obrigatório, DataSet): o objeto do conjunto de dados.
- Retorna o status da solicitação e o conjunto de dados com os registros.

### Excluir um registro

Esse endpoint permite excluir um registro de um conjunto de dados. Isso requer autenticação.

- Métodos suportados: DELETE
- Caminho: `/api/services/rest/crud/deleteRecord`
- Argumentos:

  conjunto de dados

  (obrigatório, DataSet): o objeto do conjunto de dados

  registro

  (obrigatório, Registro): o registro a ser excluído
- Retorna o status da exclusão.

### Atualizar um registro

Esse endpoint permite atualizar um registro associado a um conjunto de dados. Isso requer autenticação.

- Métodos suportados: POST
- Caminho: `/api/services/rest/crud/updateRecord`
- Argumentos:

conjunto de dados

(obrigatório, DataSet): o objeto do conjunto de dados

registro

(obrigatório, Registro): o registro a ser atualizado

- Retorna o status da solicitação e o conjunto de dados com os registros.

Salvar um registro

Esse endpoint permite salvar um registro em um conjunto de dados e usar uma máscara. Isso requer autenticação.

- Métodos suportados: POST
- Caminho: `/api/services/rest/crud/saveRecord`
- Argumentos:

  conjunto de dados

  (obrigatório, DataSet): o objeto do conjunto de dados

  registro

  (obrigatório, Registro): o registro a ser salvo
- Retorna o status da solicitação e o conjunto de dados com os registros.

## Máscaras

Use os seguintes endpoints para carregar ou aplicar máscaras a um conjunto de dados.

Tópicos

- Carregue máscaras
- Aplicar máscara
- Aplicar filtro de máscara

### Carregue máscaras

Esse endpoint permite recuperar todas as máscaras associadas a um conjunto de dados específico. Isso requer autenticação.

- Métodos suportados: POST
- Caminho: `/api/services/rest/crud/loadMasks`
- Argumentos:

  conjunto de dados

  (obrigatório, DataSet): o objeto do conjunto de dados
- Retorna o status da solicitação e a lista das máscaras.

### Aplicar máscara

Esse endpoint permite aplicar uma máscara a um conjunto de dados específico. Isso requer autenticação.

- Métodos suportados: POST
- Caminho: `/api/services/rest/crud/applyMask`
- Argumentos:

  conjunto de dados

  (obrigatório, DataSet): o objeto do conjunto de dados

  máscara

  (obrigatório, Máscara): o objeto do conjunto de dados
- Retorna o status da solicitação e do conjunto de dados com a máscara aplicada.

### Aplicar filtro de máscara

Esse endpoint permite aplicar uma máscara e um filtro a um conjunto de dados específico. Isso requer autenticação.

- Métodos suportados: POST
- Caminho: `/api/services/rest/crud/applyMaskFilter`

- Argumentos:

  conjunto de dados

  (obrigatório, DataSet): o objeto do conjunto de dados

  máscara

  (obrigatório, Máscara): o objeto do conjunto de dados
- Retorna o status da solicitação e do conjunto de dados com a máscara e o filtro aplicados.

## Outros

Use os seguintes endpoints para gerenciar o cache de um conjunto de dados ou verificar as características do conjunto de dados:

Tópicos



### Verificar o cache de aquecimento

Verifica se o cache de aquecimento está ativado para um conjunto de dados específico. Isso requer autenticação.

- Métodos suportados: POST
- Caminho: `` `/api/services/rest/bluesamservice/warmupCache` ``
- Argumentos:

  name

  (obrigatório, string): o nome do conjunto de dados.
- Retorna: verdadeiro se o cache de aquecimento estiver ativado e falso caso contrário.

### Verifique o cache ativado

Verifica se o cache está habilitado para um conjunto de dados específico. Isso requer autenticação.

- Métodos suportados: GET
- Caminho: `/api/services/rest/bluesamservice/isEnableCache`
- Argumentos: nenhum
- Retorna verdadeiro se o armazenamento em cache estiver ativado.

### Habilitar cache

- Métodos suportados: GET
- Caminho: `/api/services/rest/bluesamservice/enableDisableCache/{enable}`
- Argumentos:

  enable

  (obrigatório, booleano): se definido como verdadeiro, ele habilitará o armazenamento em cache.
- Nenhum retorno

### Verifique o cache de RAM alocado

Esse endpoint permite recuperar a memória cache RAM alocada. Isso requer autenticação.

- Métodos suportados: GET
- Caminho: `/api/services/rest/bluesamservice/allocatedRamCache`
- Argumentos: nenhum
- Retorna: o tamanho da memória como uma string

### Verificar a persistência

- Métodos suportados: GET
- Caminho: `/api/services/rest/bluesamservice/persistence`
- Argumentos: nenhum
- Retorna: a persistência usada como uma string

Verifique os tipos de conjuntos de dados compatíveis

- Caminho: `/api/services/rest/bluesamservice/getDataSetTypes`
- Argumentos: nenhum
- Retorna: a lista de tipos de conjuntos de dados compatíveis como uma lista de cadeias de caracteres.

Verificar a integridade do servidor

- Métodos suportados: GET
- Caminho: `/api/services/rest/bluesamservice/serverIsUp`
- Argumentos: nenhum
- Retornos: nenhum

## Usuários

Use os seguintes endpoints para gerenciar as interações do usuário.

Tópicos



Login

- Métodos suportados: POST
- Caminho: `/api/services/security/servicelogin/login`
- Argumentos:

  username

  (obrigatório, string)

password

(obrigatório, string)

- Retorna o nome de usuário e as funções do usuário conectado

Exemplo de resposta

```
{"login":"some-user","roles":[{"id":0,"roleName":"ROLE_ADMIN"}]}
```

Verificar conta de usuário

- Métodos suportados: POST
- Caminho: `/api/services/security/servicelogin/hasAccount`
- Argumentos: nenhum
- Retorna: verdadeiro se o usuário já estiver logado

Fazer logon

- Métodos suportados: POST
- Caminho: `/api/services/security/servicelogin/recorduser`
- Argumentos: nenhum
- Retorna: verdadeiro se o usuário já estiver logado

Listar todos os usuários

- Métodos suportados: GET
- Caminho: `/api/services/security/servicelogin/listusers`
- Argumentos: nenhum
- Retorna: a lista de todos os usuários

logout

- Métodos suportados: POST
- Caminho: `/api/services/security/servicelogout/logout`

- Argumentos: nenhum
- Retorna: verdadeiro se o usuário tiver sido desconectado com sucesso.

## Console de aplicativos JICS

O componente do JICS é o suporte do AWS Blu Age para a modernização dos recursos antigos do CICS. O aplicativo web do JICS Application Console é dedicado a administrar os recursos do JICS. Os seguintes endpoints permitem realizar as tarefas de administração sem precisar interagir com a interface de usuário do JAC. Sempre que um endpoint exigir autenticação, a solicitação deverá incluir detalhes de autenticação (nome de usuário/senha normalmente, conforme exigido pela Autenticação Básica). Os endpoints do aplicativo web do JICS Application Console usam o caminho raiz. `/jac/`

Tópicos

- Gerenciamento de recursos do JICS
- Endpoints de gerenciamento de usuários do JAC

### Gerenciamento de recursos do JICS

Todos os endpoints a seguir estão relacionados ao gerenciamento de recursos do JICS, permitindo que os administradores do JICS lidem com os recursos diariamente.

Tópicos

- Listar LISTAS E GRUPOS DO JICS
- Listar JICS GROUPS
- Listar GRUPOS JICS para uma determinada LISTA
- LISTAR recursos JICS para um determinado GRUPO
- LISTAR recursos JICS para um determinado GRUPO (alternativa usando um nome)
- Editando os GRUPOS próprios de várias LISTAS
- Excluir um link
- Excluir um grupo
- Excluir uma TRANSAÇÃO
- Como excluir um programa
- Excluir um arquivo
- Excluir um TDQUEUE

- Exclui um modelo.
- Criar uma LISTA
- Criar um grupo
- Considerações comuns sobre a criação de RECURSOS
- Crie um ID de transação.
- Como criar um programa
- Crie um arquivo .
- Crie um TDQUEUE
- Cria um modelo.
- Atualizar um link
- Atualiza um grupo.
- Considerações sobre a atualização de RECURSOS COMUNS
- Atualizar uma TRANSAÇÃO
- Atualizar um PROGRAMA
- Atualizar um ARQUIVO
- Atualizar um TDQUEUE
- Atualiza um modelo.

Listar LISTAS E GRUPOS DO JICS

A LISTA e os GRUPOS são os principais recursos de contêiner proprietários dentro do componente JICS. Todos os recursos do JICS devem pertencer a um GRUPO. Os grupos podem pertencer às LISTAS, mas isso não é obrigatório. As LISTAS podem até não existir em um determinado ambiente JICS, mas, na maioria das vezes, as LISTAS existem para fornecer uma camada extra de organização dos recursos. Para obter mais informações sobre a organização de recursos do CICS, consulte os recursos do CICS.

- Método suportado: GET
- Requer autenticação
- Caminho: `/api/services/rest/jicsservice/listJicsListsAndGroups`
- Retorna: uma lista de JicsContainer objetos serializados, tanto LISTS quanto GROUPS, como JSON.

Resposta de exemplo:

```
[
    {
      "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
    }
  ]
```

Listar JICS GROUPS

- Método suportado: GET
- Requer autenticação
- Caminho: `/api/services/rest/jicsservice/listJicsGroups`
- Retorna: uma lista de JicsContainer objetos serializados (GRUPOS) como JSON. Os GRUPOS estão sendo retornados sem suas próprias informações de LISTA.

Resposta de exemplo:

```
[
```

```
  {
    "jacType": "JACGroup",
    "name": "MURACHS",
    "isActive": true,
    "children": []
  },
  {
    "jacType": "JACGroup",
    "name": "TEST",
    "isActive": true,
    "children": []
  }
]
```

Listar GRUPOS JICS para uma determinada LISTA

- Método suportado: POST
- Requer autenticação
- Caminho: `/api/services/rest/jicsservice/listGroupsForList`
- Argumentos: uma carga JSON, representando a LISTA JICS cujos GRUPOS estamos procurando. Essa é a serialização JSON de um objeto `com.netfective.bluage.jac.entities.JACList`.

  Exemplo de solicitação

```
{
    "jacType":"JACList",
    "name":"MURACHS",
    "isActive":true
  }
```

- Retorna: uma lista de JicsContainer objetos serializados (GROUPS) como JSON, anexados à LIST fornecida. Os GRUPOS estão sendo retornados sem suas próprias informações de LISTA.

  Resposta de exemplo:

```
[
    {
      "jacType": "JACGroup",
      "name": "MURACHS",
      "isActive": true,
```

```
      "children": []
    }
  ]
```

LISTAR recursos JICS para um determinado GRUPO

- Método suportado: POST
- Requer autenticação
- Caminho: `/api/services/rest/jicsservice/listResourcesForGroup`
- Argumentos: uma carga JSON, representando o JICS GROUP cujos recursos estamos procurando. Essa é a serialização JSON de um objeto `com.netfective.bluage.jac.entities.JACGroup`. Você não precisa especificar todos os campos para o GRUPO, mas o nome é obrigatório.

  Exemplo de solicitação

```
{
    "jacType":"JACGroup",
    "name":"MURACHS",
    "isActive":true
  }
```

- Retorna: uma lista de JicsResource objetos serializados, de propriedade do DETERMINADO GRUPO. Os objetos não estão sendo retornados em nenhuma ordem específica e são de tipos diferentes (PROGRAMA, TRANSAÇÃO, ARQUIVO, etc...).

LISTAR recursos JICS para um determinado GRUPO (alternativa usando um nome)

- Método suportado: POST
- Requer autenticação
- Caminho: `/api/services/rest/jicsservice/listResourcesForGroupName`
- Argumentos: o nome do GRUPO que possui os recursos que estamos procurando.
- Retorna: uma lista de JicsResource objetos serializados, de propriedade do DETERMINADO GRUPO. Os objetos estão sendo retornados em nenhuma ordem específica e são de tipos diferentes (PROGRAMA, TRANSAÇÃO, ARQUIVO, etc...)

Editando os GRUPOS próprios de várias LISTAS

- Método suportado: POST
- Requer autenticação
- Caminho: `/api/services/rest/jicsservice/editGroupsList`
- Argumentos: uma representação JSON de uma coleção de LISTAS com grupos infantis;

  Exemplo de solicitação

```
[
  {
        "jacType": "JACList",
        "name": "MURACHS",
        "isActive": true,
        "children": [
          {
            "jacType": "JACGroup",
            "name": "MURACHS",
            "isActive": true,
            "children": []
          },
          {
            "jacType": "JACGroup",
            "name": "TEST",
            "isActive": true,
            "children": []
          }
        ]
  }
]
```

  Antes dessa edição, somente o grupo chamado “MURACHS” pertencia à LISTA chamada “MURACHS”. Com essa edição, “adicionamos” o grupo chamado “TEST” à LISTA chamada “MURACHS”.
- Retorna um valor Booliano. Se o valor for 'true', as modificações do LISTS persistiram adequadamente no armazenamento JICS subjacente.

Excluir um link

- Método suportado: POST

- Requer autenticação
- Caminho: `/api/services/rest/jicsservice/deleteList`
- Argumentos: uma carga JSON, representando a LISTA JICS a ser excluída. Essa é a serialização JSON de um objeto `com.netfective.bluage.jac.entities.JACList`.
- Retorna um valor Booliano. Se o valor for 'true', a exclusão de LIST foi operada adequadamente no armazenamento JICS subjacente.

Excluir um grupo

- Método suportado: POST
- Requer autenticação
- Caminho: `/api/services/rest/jicsservice/deleteGroup`
- Argumentos: uma carga JSON, representando o GRUPO JICS a ser excluído. Essa é a serialização JSON de um objeto `com.netfective.bluage.jac.entities.JACGroup`.
- Retorna um valor Booliano. Se o valor for 'true', a exclusão do GRUPO foi operada adequadamente no armazenamento JICS subjacente.

Excluir uma TRANSAÇÃO

- Método suportado: POST
- Requer autenticação
- Caminho: `/api/services/rest/jicsservice/deleteTransaction`
- Argumentos: uma carga JSON, representando a transação JICS a ser excluída. Essa é a serialização JSON de um objeto `com.netfective.bluage.jac.entities.JACTransaction`.
- Retorna um valor Booliano. Se o valor for 'true', a exclusão da TRANSAÇÃO foi operada adequadamente no armazenamento JICS subjacente.

Como excluir um programa

- Método suportado: POST
- Requer autenticação
- Caminho: `/api/services/rest/jicsservice/deleteProgram`

- Argumentos: uma carga JSON, representando o programa JICS a ser excluído. Essa é a serialização JSON de um objeto `com.netfective.bluage.jac.entities.JACProgram`.
- Retorna um valor Booliano. Se o valor for 'true', a exclusão do PROGRAMA foi operada adequadamente no armazenamento JICS subjacente.

Excluir um arquivo

- Método suportado: POST
- Requer autenticação
- Caminho: `/api/services/rest/jicsservice/deleteFile`
- Argumentos: uma carga JSON, representando o arquivo JICS a ser excluído. Essa é a serialização JSON de um objeto `com.netfective.bluage.jac.entities.JACFile`.
- Retorna um valor Booliano. Se o valor for 'true', a exclusão do ARQUIVO foi operada adequadamente no armazenamento JICS subjacente.

Excluir um TDQUEUE

- Método suportado: POST
- Requer autenticação
- Caminho: `/api/services/rest/jicsservice/deleteTDQueue`
- Argumentos: uma carga JSON, representando o JICS TDQUEUE a ser excluído. Essa é a serialização JSON de um objeto `com.netfective.bluage.jac.entities.jactDQueue`.
- Retorna um valor Booliano. Se o valor for 'true', a exclusão do TDQUEUE foi operada adequadamente no armazenamento JICS subjacente.

Exclui um modelo.

- Método suportado: POST
- Requer autenticação
- Caminho: `/api/services/rest/jicsservice/deleteTSModel`
- Argumentos: uma carga JSON, representando o JICS TSMODEL a ser excluído. Essa é a serialização JSON de um objeto `com.netfective.bluage.jac.entities.JactsModel`.
- Retorna um valor Booliano. Se o valor for 'true', a exclusão do TSMODEL foi operada adequadamente no armazenamento JICS subjacente.

Criar uma LISTA

- Método suportado: POST
- Requer autenticação
- Caminho: `/api/services/rest/jicsservice/createList`
- Argumentos: uma carga JSON, representando a LISTA JICS a ser criada. Essa é a serialização JSON de um objeto `com.netfective.bluage.jac.entities.jacList`.
- Retorna um valor Booliano. Se o valor for 'true', a LIST foi criada corretamente no armazenamento JICS subjacente.

> **Note**
>
> A LISTA sempre será criada vazia. Anexar GRUPOS à LISTA exigirá outra operação.

Criar um grupo

- Método suportado: POST
- Requer autenticação
- Caminho: `/api/services/rest/jicsservice/createGroup`
- Argumentos: uma carga JSON, representando o GRUPO JICS a ser criado. Essa é a serialização JSON de um objeto `com.netfective.bluage.jac.entities.JACGroup`.
- Retorna um valor Booliano. Se o valor for 'true', o GRUPO foi criado corretamente no armazenamento JICS subjacente.

> **Note**
>
> O GRUPO sempre será criado vazio. Anexar RECURSOS ao GRUPO exigirá operações adicionais (a criação de recursos os anexará automaticamente a um determinado GRUPO).

Considerações comuns sobre a criação de RECURSOS

Todos os endpoints a seguir estão relacionados à criação de JICS RESOURCES e compartilham algumas restrições comuns: na carga útil da solicitação a ser enviada ao endpoint, o campo `groupName` precisa ser valorizado.

Restrição de propriedade do GRUPO:

Nenhum recurso pode ser criado sem ser anexado a um grupo existente, e o endpoint usa o GroupName para recuperar o grupo ao qual esse recurso será anexado. O `groupName` deve ser igual ao nome de um GRUPO existente. Uma mensagem de erro com HTTP STATUS 400 será enviada se não `groupName` estiver apontando para um grupo existente no armazenamento subjacente do JICS.

Restrição de unicidade dentro de um GRUPO:

Um recurso especificado com um nome especificado deve ser exclusivo dentro de um grupo especificado. A verificação da unicidade será realizada por cada endpoint de criação de recursos. Se a carga útil fornecida não respeitar a restrição de unicidade, o endpoint enviará uma resposta com HTTP STATUS 400 (BAD REQUEST) -- veja o exemplo de resposta abaixo.

Exemplo de carga útil: tentamos criar a transação 'ARIT' no grupo 'TEST', mas uma transação com esse nome já existe nesse GRUPO.

```
{
    "jacType":"JACTransaction",
    "name":"ARIT",
    "groupName":"TEST",
    "isActive":true
  }
```

Recebemos a seguinte resposta de erro:

```
{
    "timestamp": 1686759054510,
    "status": 400,
    "error": "Bad Request",
    "path": "/jac/api/services/rest/jicsservice/createTransaction"
  }
```

A inspeção dos registros dos servidores confirmará a origem do problema:

```
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
```

Crie um ID de transação.

- Método suportado: POST
- Requer autenticação
- Caminho: `/api/services/rest/jicsservice/createTransaction`
- Argumentos: uma carga JSON, representando a TRANSAÇÃO JICS a ser criada. Essa é a serialização JSON de um objeto `com.netfective.bluage.jac.entities.JACTransaction`.
- Retorna um valor Booliano. Se o valor for 'true', a TRANSAÇÃO foi criada corretamente no armazenamento JICS subjacente.

Como criar um programa

- Método suportado: POST
- Requer autenticação
- Caminho: `/api/services/rest/jicsservice/createProgram`
- Argumentos: uma carga JSON, representando o PROGRAMA JICS a ser criado. Essa é a serialização JSON de um objeto `com.netfective.bluage.jac.entities.JACProgram`.
- Retorna um valor Booliano. Se o valor for 'true', o PROGRAMA foi criado corretamente no armazenamento JICS subjacente.

Crie um arquivo .

- Método suportado: POST
- Requer autenticação
- Caminho: `/api/services/rest/jicsservice/createFile`
- Argumentos: uma carga JSON, representando o ARQUIVO JICS a ser criado. Essa é a serialização JSON de um objeto `com.netfective.bluage.jac.entities.jacFile`.

- Retorna um valor Booliano. Se o valor for 'true', o ARQUIVO foi criado corretamente no armazenamento JICS subjacente.

Crie um TDQUEUE

- Método suportado: POST
- Requer autenticação
- Caminho: `/api/services/rest/jicsservice/createTDQueue`
- Argumentos: uma carga JSON, representando o JICS TDQUEUE a ser criado. Essa é a serialização JSON de um objeto `com.netfective.bluage.jac.entities.jactDQueue`.
- Retorna um valor Booliano. Se o valor for 'true', o TDQUEUE foi criado corretamente no armazenamento JICS subjacente.

Cria um modelo.

- Método suportado: POST
- Requer autenticação
- Caminho: `/api/services/rest/jicsservice/createTSModel`
- Argumentos: uma carga JSON, representando o JICS TSMODEL a ser criado. Essa é a serialização JSON de um objeto `com.netfective.bluage.jac.entities.JACTSModel`.
- Retorna um valor Booliano. Se o valor for 'true', o TSMODEL foi criado corretamente no armazenamento JICS subjacente.

Atualizar um link

- Método suportado: POST
- Requer autenticação
- Caminho: `/api/services/rest/jicsservice/updateList`
- Argumentos: uma carga JSON, representando a LISTA JICS a ser atualizada. Essa é a serialização JSON de um objeto `com.netfective.bluage.jac.entities.JACList`. Não há necessidade de fornecer os filhos do LIST, o mecanismo de atualização do LIST não levará isso em consideração.
- Retorna um valor Booliano. Se o valor for 'true', a LIST foi atualizada corretamente no armazenamento JICS subjacente.

A atualização do sinalizador LIST 'isActive' será propagada para todos os elementos de propriedade da LIST, ou seja, todos os GRUPOS pertencentes à LIST e todos os RECURSOS pertencentes a esses GRUPOS. Essa é uma maneira conveniente de desativar muitos recursos com uma única operação, em vários GRUPOS.

Atualiza um grupo.

- Método suportado: POST
- Requer autenticação
- Caminho: `/api/services/rest/jicsservice/updateGroup`
- Argumentos: uma carga JSON, representando o JICS GROUP a ser atualizado. Essa é a serialização JSON de um objeto `com.netfective.bluage.jac.entities.JACGroup`. Não há necessidade de fornecer os filhos do GRUPO, o mecanismo de atualização do GRUPO não levará isso em consideração.
- Retorna um valor Booliano. Se o valor for 'true', o GRUPO foi atualizado corretamente no armazenamento JICS subjacente.

> **Note**
>
> A atualização do sinalizador GROUP 'isActive' se propagará para todos os elementos de propriedade do GRUPO, ou seja, todos os RECURSOS de propriedade do GRUPO. Essa é uma maneira conveniente de desativar muitos recursos com uma única operação em um determinado GRUPO.

Considerações sobre a atualização de RECURSOS COMUNS

Todos os endpoints a seguir tratam da atualização do JICS RESOURCES. Usando o campo `groupName`, você pode alterar o GRUPO proprietário de qualquer RECURSO JICS, desde que o valor do campo aponte para um GRUPO existente no armazenamento JICS subjacente (caso contrário, você receberá uma resposta BAD REQUEST (HTTP STATUS 400) do endpoint).

Atualizar uma TRANSAÇÃO

- Método suportado: POST
- Requer autenticação
- Caminho: `/api/services/rest/jicsservice/updateTransaction`

- Argumentos: uma carga JSON, representando a TRANSAÇÃO JICS a ser atualizada. Essa é a serialização JSON de um objeto `com.netfective.bluage.jac.entities.JACTransaction`.
- Retorna um valor Booliano. Se o valor for 'true', a TRANSAÇÃO foi atualizada corretamente no armazenamento JICS subjacente.

Atualizar um PROGRAMA

- Método suportado: POST
- Requer autenticação
- Caminho: `/api/services/rest/jicsservice/updateProgram`
- Argumentos: uma carga JSON, representando o PROGRAMA JICS a ser atualizado. Essa é a serialização JSON de um objeto `com.netfective.bluage.jac.entities.JACProgram`.
- Retorna um valor Booliano. Se o valor for 'true', o PROGRAMA foi atualizado corretamente no armazenamento JICS subjacente.

Atualizar um ARQUIVO

- Método suportado: POST
- Requer autenticação
- Caminho: `/api/services/rest/jicsservice/updateFile`
- Argumentos: uma carga JSON, representando o ARQUIVO JICS a ser atualizado. Essa é a serialização JSON de um objeto `com.netfective.bluage.jac.entities.JACFile`.
- Retorna um valor Booliano. Se o valor for 'true', o ARQUIVO foi atualizado corretamente no armazenamento JICS subjacente.

Atualizar um TDQUEUE

- Método suportado: POST
- Requer autenticação
- Caminho: `/api/services/rest/jicsservice/updateTDQueue`
- Argumentos: uma carga JSON, representando o JICS TDQUEUE a ser atualizado. Essa é a serialização JSON de um objeto `com.netfective.bluage.jac.entities.JACTDQueue`.

- Retorna um valor Booliano. Se o valor for 'true', o TDQueue foi atualizado corretamente no armazenamento JICS subjacente.

Atualiza um modelo.

- Método suportado: POST
- Requer autenticação
- Caminho: `/api/services/rest/jicsservice/updateTSModel`
- Argumentos: uma carga JSON, representando o JICS TSMODEL a ser atualizado. Essa é a serialização JSON de um objeto `com.netfective.bluage.jac.entities.JACTSModel`.
- Retorna um valor Booliano. Se o valor for 'true', o TSMODEL foi atualizado corretamente no armazenamento JICS subjacente.

## Endpoints de gerenciamento de usuários do JAC

Tópicos

- Exclusão de um usuário do
- Faça login como usuário .
- Testando se existe pelo menos um usuário no sistema
- Gravação de um novo usuário
- Listar usuários
- Listar usuários
- Fazer logout do usuário atual
- Estado de integridade do servidor Jics

Exclusão de um usuário do

- Método suportado: POST
- Requer autenticação e direitos de administrador
- Caminho: `/api/services/security/servicelogin/deleteuser`
- Argumentos: a serialização JSON de um `com.netfective.bluage.jac.entities.SignOn` objeto, representando o usuário a ser removido do armazenamento.

- Retorna um valor Booliano. Se o valor for 'true', o usuário foi removido adequadamente do sistema JICS.

> **Note**
>
> Essa ação não pode ser desfeita, o usuário excluído não poderá se conectar ao aplicativo JAC novamente. Use com cautela.

Faça login como usuário .

- Método suportado: POST
- Requer autenticação e direitos de administrador
- Caminho: `/api/services/security/servicelogin/login`
- Retorna a serialização JSON de um `com.netfective.bluage.jac.entities.Sign0n` objeto, representando o usuário cujas credenciais são fornecidas na solicitação atual. A senha está oculta da exibição no objeto retornado. As funções atribuídas ao usuário estão sendo listadas.

Resposta de exemplo:

```
{
    "login": "sadmin",
    "password": null,
    "roles": [
      {
        "id": 0,
        "roleName": "ROLE_SUPER_ADMIN"
      }
    ]
  }
```

Testando se existe pelo menos um usuário no sistema

- Método suportado: GET
- Caminho: `/api/services/security/servicelogin/hasAccount`
- Retorna um valor booleano, cujo valor é verdadeiro se pelo menos um usuário -- diferente do usuário superadministrador padrão -- tiver sido criado no sistema JICS.

Gravação de um novo usuário

- Método suportado: POST
- Requer autenticação e direitos de administrador
- Caminho: `/api/services/security/servicelogin/recorduser`
- Argumentos: a serialização JSON de um `com.netfective.bluage.jac.entities.SignOn` objeto, representando o usuário a ser adicionado ao armazenamento. As funções do usuário devem ser definidas, caso contrário, o usuário talvez não consiga usar o recurso e os endpoints do JAC.

Exemplo de solicitação

```
{
    "login": "simpleuser",
    "password": "simpleuser",
    "roles": [
      {
        "id": 2,
        "roleName": "ROLE_USER"
      }
    ]
  }
```

Somente as seguintes funções podem ser usadas ao gravar um novo usuário:

- ROLE_ADMIN: pode gerenciar recursos e usuários do JICS.
- ROLE_USER: pode gerenciar recursos do JICS, mas não usuários.

Listar usuários

- Método suportado: GET
- Requer autenticação e direitos de administrador
- Caminho: `/api/services/security/servicelogin/listusers`
- Retorna uma lista de`com.netfective.bluage.jac.entities.SignOn`, serializada como JSON.

Listar usuários

- Método suportado: GET
- Requer autenticação e direitos de administrador
- Caminho: `/api/services/security/servicelogin/listusers`
- Retorna uma lista de`com.netfective.bluage.jac.entities.SignOn`, serializada como JSON.

Fazer logout do usuário atual

- Método suportado: GET
- Caminho: `/api/services/security/servicelogout/logout`
- Retorna a seguinte mensagem JSON: “{" success” :true}” se o logout do usuário atual for bem-sucedido (a sessão HTTP relacionada será invalidada).

Estado de integridade do servidor Jics

- Método suportado: GET
- Caminho: `/api/services/rest/jicsserver/serverIsUp`
- Indica que o servidor JICS está ativo e funcionando, se você receber uma resposta com HTTP STATUS 200.

## Estruturas de dados

Esta seção descreve os detalhes das várias estruturas de dados.

Tópicos

## Estrutura de mensagens do Job Execution Details

Cada detalhe da execução do trabalho terá os seguintes campos:

scriptId

o identificador do script chamado.

chamador

Endereço IP do chamador.

Identifier

identificador exclusivo de execução do trabalho.

startTime

data e hora em que a execução de trabalho foi iniciada.

endTime

data e hora em que a execução de trabalho foi encerrada.

status

um status para a execução do trabalho. Um valor possível entre:

- `DONE`: a execução do trabalho terminou normalmente.
- `TRIGGERED`: a execução do trabalho foi acionada, mas ainda não foi lançada.
- `RUNNING`: a execução do trabalho está em execução.
- `KILLED`: a execução do trabalho foi interrompida.
- `FAILED`: a execução do trabalho falhou.

Resultado da execução

uma mensagem para resumir o resultado da execução do trabalho. Essa mensagem pode ser uma mensagem simples se a execução do trabalho ainda não tiver sido concluída ou uma estrutura JSON com os seguintes campos:

- ExitCode: código de saída numérico; valores negativos indicam situações de falha.
- programa: último programa lançado pelo cargo.
- status: um valor possível entre:

  - `Error`: quando exitCode = -1; isso corresponde a um erro (técnico) que ocorre durante a execução do trabalho.
  - `Failed`: quando exitcode = -2; Isso corresponde a uma falha que ocorre durante a execução de um programa de serviço (como uma situação ABEND).
  - `Succeeded`: quando ExitCode >= 0;
- stepName: nome da última etapa executada no trabalho.

Modo de execução

síncrona ou assíncrona, dependendo da forma como a tarefa foi iniciada.

Exemplo de resultado:

```
{
    "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"
  }
```

## Estrutura de resultados do lançamento da transação

A estrutura pode conter os seguintes campos:

outCome

uma string representando o resultado da execução da transação. Os valores possíveis são:

- `Success`: a execução da transação foi até o final corretamente.
- `Failure`: a execução da transação falhou ao terminar corretamente, alguns problemas foram encontrados.

vírgula

uma string representando o valor final COMMAREA, como uma matriz de bytes codificada em byte64. Pode ser uma string vazia.

Registro de contêiner

(opcional) uma string representando o conteúdo do registro do CONTAINER como uma matriz de bytes codificada em byte64.

Descrição do servidor

Pode conter informações sobre o servidor que atendeu à solicitação (para fins de depuração). Pode ser uma string vazia.

Um código Bend

(opcional) se o programa referenciado pela transação lançada for alterado, o valor do código de abend será retornado como uma string nesse campo.

Exemplo de respostas:

Bem-sucedida

```
{
    "outCome": "Success",
    "commarea": "",
    "serverDescription": ""
  }
```

Falha

```
{
    "outCome": "Failure",
    "commarea": "",
    "serverDescription": "",
    "abendCode": "AEIA"
  }
```

## Estrutura de resultados do registro de lançamento da transação

A estrutura pode conter os seguintes campos:

Conteúdo do registro

uma string representando o conteúdo do registro do COMMAREA como uma matriz de bytes codificada em byte64.

Registro de contêiner

uma string representando o conteúdo do registro do CONTAINER como uma matriz de bytes codificada em byte64.

Descrição do servidor

Pode conter informações sobre o servidor que atendeu à solicitação (para fins de depuração). Pode ser uma string vazia.

Exemplo de respostas:

Bem-sucedida

```
{
    "recordContent": "",
    "serverDescription": ""
}
```

## Possível status de trabalho em uma fila

Em uma fila, os trabalhos podem ter o seguinte status:

ACTIVE

O trabalho está sendo executado atualmente na fila.

EXECUÇÃO_ESPERA

O trabalho está aguardando a disponibilidade de um tópico.

PROGRAMADO

Os trabalhos são programados para execução em uma data e hora específicas.

HOLD

Job está esperando para ser lançado antes de ser executado.

CONCLUÍDO

Job foi executado com sucesso.

COM FALHA

Houve falha na execução de trabalho.

UNKNOWN

O status é desconhecido.

## Enviar entrada de trabalho

A entrada do trabalho de envio é a serialização JSON de um objeto `com.netfective.bluage.gapwalk.rt.jobqueue.SubmitJobMessage`. O exemplo de entrada abaixo exibe todos os campos desse tipo de feijão.

Exemplo de entrada

```
{
    "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
}
```

Número do trabalho

se o número do trabalho for 0, o número do trabalho será gerado automaticamente usando o próximo número na sequência numérica do trabalho. Esse valor deve ser definido como 0 (exceto para fins de teste).

Prioridade do trabalho

A prioridade de trabalho padrão no AS400 é 5. O intervalo válido é de 0 a 9, sendo 0 a prioridade mais alta.

jobOnHold

Se um trabalho for enviado em espera, ele não será executado imediatamente, mas somente quando alguém o “liberar”. Um trabalho pode ser lançado usando a API REST (/release ou / release-all).

Data e hora do agendamento

Se esses valores não forem nulos, o trabalho será executado na data e hora especificadas.

Data

pode ser fornecido com o formato MMDdyy ou ddmMyYYY (o tamanho da entrada determinará qual formato será usado)

Time

pode ser fornecido com o formato HHmm ou HHMMss (o tamanho da entrada determinará qual formato será usado)

Parâmetros do programa

isso será passado para o programa como um mapa.

## Lista de respostas de trabalhos agendados

Essa é a estrutura do endpoint da fila de trabalhos list-jobs. Observe que a mensagem de envio de trabalho que foi usada para enviar esse trabalho faz parte da resposta. Isso pode ser usado para fins de rastreamento, teste/reenvio. Quando um trabalho for concluído, a data de início e a data de término também serão preenchidas.

```
[
    {
      "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
      }
    }
  ]
```

## Lista de respostas de vagas “em espera”

A estrutura é semelhante à anterior, exceto que todos os trabalhos devolvidos ficarão “em espera”.

```
[
    {
      "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 Configuração do Blu Age Runtime (não gerenciada)

Esta seção explica as etapas para configurar o AWS Blu Age Runtime (não gerenciado) em sua AWS infraestrutura.

Tópicos



## AWS Pré-requisitos do Blu Age Runtime

AWS O Blu Age Runtime (não gerenciado) está disponível em várias versões de lançamento. Se você tiver projetos de modernização em andamento, talvez precise de versões incrementais do tempo de execução para fins de implementação e teste. Para definir suas necessidades, entre em contato com seu gerente de entrega da AWS Blu Age.

Antes de iniciar o processo de integração do AWS Blu Age Runtime (não gerenciado), faça o seguinte:

- Verifique se você tem uma AWS conta.
- Certifique-se de ter um aplicativo modernizado refatorado com o Blu Age. AWS
- Escolha uma AWS região e uma computação (Amazon EC2 ou Amazon AWS Fargate ECS ativado).
- Escolha a versão do AWS Blu Age Runtime que você deseja usar.
- Analise the section called “Requisitos de configuração de infraestrutura” e valide os componentes adicionais necessários para executar o AWS Blu Age Runtime (não gerenciado).

> **Note**
>
> Se quiser testar os recursos do AWS Blu Age Runtime (não gerenciado), você pode usar o aplicativo de demonstração`Planets Demo`, que pode ser baixado do PlanetsDemo -v1.zip.

# AWS Integração do Blu Age Runtime

Para começar, crie um AWS Support caso para solicitar a integração para acessar o AWS Blu Age Runtime. Inclua em sua solicitação seu Conta da AWS ID, a AWS região que você deseja usar e uma opção de computação e uma versão de tempo de execução. Se você não tiver certeza de qual versão precisa, entre em contato com seu gerente de entrega da AWS Blu Age.

## AWS Blu Age Runtime (não gerenciado) no Amazon EC2

Armazenamos os artefatos do AWS Blu Age Runtime (não gerenciados) em diferentes buckets do Amazon S3 por região e por opção de computação. Para acessar o bucket do seu Região da AWS for AWS Blu Age Runtime (não gerenciado) no Amazon EC2, use o nome listado na tabela a seguir.

| Região da AWS | Liberação do balde | Construção de balde |
|---|---|---|
| Leste dos EUA (Ohio) | aws-bluage-runtime-artifacts-055777665268-us-east-2 | aws-bluage-runtime-artifacts-dev-055777665268-us-east-2 |
| Leste dos EUA (Norte da Virgínia) | aws-bluage-runtime-artifacts-139023371234-us-east-1 | aws-bluage-runtime-artifacts-dev-139023371234-us-east-1 |
| Oeste dos EUA (N. da Califórnia) | aws-bluage-runtime-artifacts-788454048782-us-west-1 | aws-bluage-runtime-artifacts-dev-788454048782-us-west-1 |
| Oeste dos EUA (Oregon) | aws-bluage-runtime-artifacts-836771190483-us-west-2 | aws-bluage-runtime-artifacts-dev-836771190483-us-west-2 |
| Europa (Irlanda) | aws-bluage-runtime-artifacts-925278190477-eu-west-1 | aws-bluage-runtime-artifacts-dev-925278190477-eu-west-1 |
| Europa (Paris) | aws-bluage-runtime-artifacts-673009995881-eu-west-3 | aws-bluage-runtime-artifacts-dev-673009995881-eu-west-3 |
| Europa (Frankfurt) | aws-bluage-runtime-artifacts-485196800481-eu-central-1 | aws-bluage-runtime-artifacts-dev-485196800481-eu-central-1 |
| América do Sul (São Paulo) | aws-bluage-runtime-artifacts-737536804457-sa-lest-1 | aws-bluage-runtime-artifacts-dev-737536804457-sa-east-1 |

| Região da AWS | Liberação do balde | Construção de balde |
|---|---|---|
| Ásia-Pacífico (Tóquio) | aws-bluage-runtime-artifacts-445578176276-ap-nordeste-1 | aws-bluage-runtime-artifacts-dev-445578176276-ap-nordeste-1 |
| Ásia-Pacífico (Sydney) | aws-bluage-runtime-artifacts-726160321909-ap-sudeste-2 | aws-bluage-runtime-artifacts-dev-726160321909-ap-southeast-2 |

## AWS Blu Age Runtime (não gerenciado) no Amazon ECS gerenciado pela Fargate

Armazenamos os artefatos do AWS Blu Age Runtime (não gerenciados) em diferentes buckets do Amazon S3 por região e por opção de computação. Para acessar o bucket do seu Região da AWS for AWS Blu Age Runtime (não gerenciado) no Amazon ECS gerenciado pela Fargate, use o nome listado na tabela a seguir.

| Região da AWS | Liberação do balde | Construção de balde |
|---|---|---|
| Leste dos EUA (Ohio) | aws-bluage-runtime-fargate-rel-483416914331-us-east-2 | aws-bluage-runtime-fargate-dev-483416914331-us-east-2 |
| Leste dos EUA (Norte da Virgínia) | aws-bluage-runtime-fargate-rel-308472162679-us-east-1 | aws-bluage-runtime-fargate-dev-308472162679-us-east-1 |
| Oeste dos EUA (N. da Califórnia) | aws-bluage-runtime-fargate-rel-343763094578-us-west-1 | aws-bluage-runtime-fargate-dev-343763094578-us-west-1 |
| Oeste dos EUA (Oregon) | aws-bluage-runtime-fargate-rel-688933007849-us-west-2 | aws-bluage-runtime-fargate-dev-688933007849-us-west-2 |
| Europa (Irlanda) | aws-bluage-runtime-fargate-rel-140138033705-eu-west-1 | aws-bluage-runtime-fargate-dev-140138033705-eu-west-1 |
| Europa (Paris) | aws-bluage-runtime-fargate-rel-339712948211-eu-west-3 | aws-bluage-runtime-fargate-dev-339712948211-eu-west-3 |

| Região da AWS | Liberação do balde | Construção de balde |
|---|---|---|
| Europa (Frankfurt) | aws-bluage-runtime-fargate-rel-339712918892-eu-central-1 | aws-bluage-runtime-fargate-dev-339712918892-eu-central-1 |
| América do Sul (São Paulo) | aws-bluage-runtime-fargate-rel-767397998881-sa-lest-1 | aws-bluage-runtime-fargate-dev-767397998881-sa-east-1 |
| Ásia-Pacífico (Tóquio) | aws-bluage-runtime-fargate-rel-891377400849-ap-nordeste-1 | aws-bluage-runtime-fargate-dev-891377400849-ap-nordeste-1 |
| Ásia-Pacífico (Sydney) | aws-bluage-runtime-fargate-rel-533267435478-ap-sudeste-2 | aws-bluage-runtime-fargate-dev-533267435478-ap-southeast-2 |

### Uso da linha de comando para listar o conteúdo do bucket

Após a integração, você pode listar o conteúdo do bucket executando o comando a seguir em um terminal.

`aws s3 ls bucket-name`

`bucket-name`Substitua pelo nome do seu bucket Região da AWS da tabela anterior.

Esse comando retorna uma lista de pastas que correspondem a diferentes versões do tempo de execução do AWS Blu Age Runtime (não gerenciado), como

```
PRE 3.3.0.1/
PRE 3.3.0.2/
```

É recomendável usar a versão mais recente disponível. Se isso não for possível, use a versão de tempo de execução que foi validada durante a fase de refatoração do aplicativo. Para listar as frameworks disponíveis para uma versão específica, execute o seguinte comando:

`aws s3 ls s3://bucket-name/version/Framework/`

`bucket-name`Substitua pelo nome do bucket para você Região da AWS e `version` pela versão desejada. Veja um exemplo a seguir.

`aws s3 ls s3://aws-bluage-runtime-artifacts-139023371234-us-east-1/3.4.0/Framework/`

O comando retornará uma lista de estruturas, como:

```
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
```

## Baixe o framework

Você pode baixar a estrutura, por exemplo, para atualizar a versão do Velocity Runtime em uma instância existente do Amazon EC2.

`aws s3 cp s3://`*`bucket-name`*`/`*`version`*`/Framework/ `*`folder-of-your-choice`*` --recursive`

Em que:

folder-of-your-choice

caminho da pasta onde você gostaria de baixar a estrutura.

Por exemplo: `aws s3 cp s3://aws-bluage-runtime-artifacts-139023371234-us-east-1/3.4.0/Framework/ . --recursive`

Esse comando produzirá a saída a seguir:

```
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
```

Você pode listar os arquivos da estrutura da seguinte forma:

```
ls -l
```

Esse comando produzirá a saída a seguir:

```
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
```

## Requisitos de configuração de infraestrutura para o AWS Blu Age Runtime (não gerenciado)

Este tópico descreve a configuração mínima de infraestrutura necessária para executar o AWS Blu Age Runtime (não gerenciado). Os procedimentos a seguir descrevem como configurar o AWS Blu Age Runtime (não gerenciado) na computação de sua escolha para implantar um aplicativo modernizado no Blu Age Runtime. AWS Os recursos que você cria devem estar em uma Amazon VPC que tenha uma sub-rede dedicada ao domínio do seu aplicativo.

Criar um grupo de segurança

1. Abra o console da Amazon VPC em https://console.aws.amazon.com/vpc/.
2. No painel de navegação esquerdo, em Segurança, escolha Grupos de segurança.
3. No painel central, escolha Criar grupo de segurança.
4. No campo Nome do grupo de segurança, insira **`M2BluagePrivateLink-SG`**.
5. Na seção Regras de entrada, escolha Adicionar regra.
6. Para Tipo, escolha HTTPS.
7. Em Origem, insira seu CIDR da VPC.
8. Na seção Regras de saída, escolha Adicionar regra.
9. Para Tipo, escolha HTTPS.
10. Em Destination, insira **`0.0.0.0/0`**.
11. Escolha Create security group (Criar grupo de segurança).

### Criar um endpoint da Amazon VPC

1. Abra o console da Amazon VPC em https://console.aws.amazon.com/vpc/.
2. No menu à de navegação esquerdo, em Nuvem privada virtual, escolha Endpoints.
3. No painel central, escolha Criar endpoint.
4. Na seção Serviços, insira **SQS** o campo de pesquisa e selecione o serviço Amazon SQS que corresponde à sua região.
5. Em VPC, selecione a Amazon VPC criada na etapa anterior.
6. Na seção Sub-redes, selecione a sub-rede que você criou para o domínio da aplicação.
7. Na seção Grupos de segurança, selecione M2 BluagePrivateLink -SG.
8. Escolha Criar endpoint.

### Criar uma política do IAM

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação esquerdo, em Gerenciamento de acesso, escolha Políticas.
3. No painel central, escolha Criar política.
4. Na seção Editor de políticas, escolha JSON.
5. Substitua todo o JSON que você vê no editor pelo seguinte JSON:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "sqs:GetQueueUrl",
                "sqs:ReceiveMessage",
                "sqs:SendMessage"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> Se precisar de mais detalhes para personalizar sua política, entre em contato com seu gerente de entrega ou gerente de contas da AWS Blu Age.

6. Escolha Próximo.
7. Insira um nome para a política e escolha Criar política.

Criar um perfil do IAM

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação esquerdo, em Gerenciamento de acesso, escolha Perfis.
3. No painel central, escolha Criar perfil.
4. Na seção Caso de uso, escolha EC2 ou Elastic Container Service (e depois Elastic Container Service Task), dependendo da sua escolha de computação.
5. Escolha Próximo.
6. No campo de pesquisa, insira o nome da política que você criou anteriormente.
7. Marque a caixa de seleção à esquerda da política.
8. Escolha Próximo.
9. Insira um nome para o perfil e escolha Criar perfil.

## Executando o AWS Blu Age Runtime no Amazon EC2

Criar uma instância do Amazon EC2

1. Abra o console do Amazon EC2 em https://console.aws.amazon.com/ec2/.
2. Escolha Iniciar instância.
3. Em Tipo de instância, escolha um dos tipos listados emthe section called “Tipos de instância do Amazon EC2 para AWS Blu Age Runtime (no Amazon EC2)”.
4. Na seção Par de chaves, selecione um par de chaves existente ou crie um.
5. Na seção Configurações de rede, escolha Selecionar grupo de segurança existente.
6. Para Grupos de segurança comuns, escolha M2 BluagePrivateLink -SG.
7. Expanda a seção Detalhes avançados.

8. Em Perfil de instância do IAM, selecione o perfil do IAM que você criou anteriormente.
9. Escolha Iniciar instância.

Quando o estado da instância do Amazon EC2 mudar para Em execução, conecte-se à instância e instale os seguintes componentes de software:

- Ambiente de Execução Java (JRE) 11.
- Apache Tomcat 9.
- AWS Blu Age Runtime (no Amazon EC2). Instale o runtime do AWS Blu Age na raiz da pasta de instalação do Apache Tomcat (alguns arquivos serão adicionados e outros serão sobrescritos).

Se você quiser instalar aplicativos da web adicionais, instale-os em uma pasta separada. Nesse caso, repita o processo de instalação do Apache Tomcat e, depois, o de arquivamento logo em seguida.

## Executando o AWS Blu Age Runtime no Amazon ECS gerenciado por AWS Fargate

Crie um cluster do Amazon ECS e uma função de tarefa para ser usada ao iniciar uma AWS Fargate tarefa posteriormente. Para a função da tarefa, inclua a política que você criou anteriormente. Dependendo do seu cenário, talvez seja necessário anexar políticas adicionais do IAM à função da tarefa.

## Tipos de instância do Amazon EC2 para AWS Blu Age Runtime (no Amazon EC2)

A seguir está uma lista dos tipos de instância do Amazon EC2 que você pode usar para o AWS Blu Age Runtime (no 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 Implantação do Blu Age Runtime no Amazon ECS gerenciado por AWS Fargate

Os tópicos desta seção descrevem como configurar o AWS Blu Age Runtime no Amazon ECS gerenciado por AWS Fargate, como atualizar a versão do tempo de execução, como monitorar sua implantação usando CloudWatch alarmes da Amazon e como adicionar dependências licenciadas.

Tópicos

- Configurando o AWS Blu Age Runtime no Amazon ECS gerenciado por AWS Fargate

- Atualizando o AWS Blu Age Runtime no Amazon ECS gerenciado por AWS Fargate
- Amazon CloudWatch Alarms for AWS Blu Age Runtime no Amazon ECS gerenciado por AWS Fargate
- Configurando dependências licenciadas no AWS Blu Age Runtime no Amazon ECS gerenciado por AWS Fargate

## Configurando o AWS Blu Age Runtime no Amazon ECS gerenciado por AWS Fargate

Este tópico explica como configurar e implantar o aplicativo de PlanetsDemo amostra usando o AWS Blu Age Runtime no Amazon ECS gerenciado por. AWS Fargate

AWS O Blu Age Runtime no Amazon ECS gerenciado por AWS Fargate está disponível para Linux/X86.

Tópicos

- Pré-requisitos
- Configuração
- Teste o PlanetsDemo aplicativo

### Pré-requisitos

Antes de começar, certifique-se de que você concluiu os seguintes pré-requisitos.

- Configure o AWS CLI seguindo as etapas em Configuração da AWS CLI.
- Preencha the section called “AWS Pré-requisitos do Blu Age Runtime”, e the section called “AWS Integração do Blu Age Runtime”.
- Baixe o AWS Blu Age Runtime no Amazon ECS gerenciado por AWS Fargate binários. Para obter instruções, consulte the section called “AWS Integração do Blu Age Runtime”.
- Baixe os binários do Apache Tomcat 9.
- Baixe o arquivo do PlanetsDemo aplicativo.
- Crie um banco de dados Amazon Aurora PostgreSQL para JICS e execute a consulta nele. `PlanetsDemo-v1/jics/sql/initJics.sql` Para obter informações sobre como criar um banco de dados Amazon Aurora PostgreSQL, consulte Criação e conexão com um cluster de banco de dados Aurora PostgreSQL.

### Configuração

Para configurar o aplicativo de PlanetsDemo amostra, conclua as etapas a seguir.

1. Depois de baixar os binários do Apache Tomcat, extraia o conteúdo e vá para a pasta. `conf` Abra o `catalina.properties` arquivo para edição e substitua a linha que começa `common.loader` com a seguinte linha.

```
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. Comprima a pasta Apache Tomcat usando o comando tar para criar um arquivo `tar.gz`.
3. Prepare um Dockerfile para criar sua imagem personalizada com base nos binários de tempo de execução fornecidos e nos binários do servidor Apache Tomcat. Veja o exemplo a seguir do Dockerfile. O objetivo é instalar o Apache Tomcat 9, seguido pelo AWS Blu Age Runtime (para Amazon ECS gerenciado por AWS Fargate) extraído na raiz do diretório de instalação do Apache Tomcat 9 e, em seguida, instalar o exemplo de aplicativo modernizado chamado. PlanetsDemo

> **Note**
>
> O conteúdo dos scripts install-gapwalk.sh e install-app.sh, usados neste exemplo do Dockerfile, está listado após o 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"]
```

A seguir está o conteúdo de 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}
```

A seguir está o conteúdo de 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. Forneça as informações de conexão do banco de dados que você criou como parte dos pré-requisitos no trecho a seguir no `application-main.yml` arquivo, localizado na pasta. `{TOMCAT_GAPWALK_DIR}/config` Para obter mais informações, consulte Criação e conexão a um cluster Aurora PostgreSQL DB.

```
datasource:
   jicsDs:
     driver-class-name :
     url:
     username:
     password:
     type :
```

5. Crie e envie a imagem para o seu repositório Amazon ECR. Para obter instruções, consulte Como enviar uma imagem do Docker no Guia do usuário do Amazon Elastic Container Registry.
6. Abra o console em https://console.aws.amazon.com/ecs/v2.
7. No painel de navegação, escolha Task definitions (Definições de tarefa).

8. Em Tipo de lançamento, escolha AWS Fargate.
9. Selecione a função da tarefa que você criou como parte dathe section called “Requisitos de configuração de infraestrutura”.
10. Anexe sua imagem ao contêiner.
11. Conclua o preenchimento do formulário e escolha Criar.
12. No painel de navegação esquerdo, escolha Clusters e, em seguida, escolha seu cluster na lista.
13. Na página de detalhes do seu cluster, na guia Serviços, escolha Criar.
14. Selecione a definição da tarefa.
15. Expanda a seção Rede e configure a VPC, as sub-redes e o grupo de segurança que você criou como parte. the section called “Requisitos de configuração de infraestrutura”
16. Implante seu serviço Amazon ECS.

Se a implantação falhar, verifique os registros. Para encontrá-los, acesse a página de tarefas no Amazon ECS gerenciada por e AWS Fargate, em seguida, escolha a guia Logs. Se você encontrar códigos de erro que começam com C seguido por um número, como CXXXX, anote as mensagens de erro. Por exemplo, o código de erro C5102 é um erro comum que indica uma configuração incorreta da infraestrutura. Você também pode navegar dentro da sua tarefa em execução e executar alguns comandos, semelhantes ao AWS Blu Age Runtime (no Amazon EC2). Para obter mais informações, consulte Usando o Amazon ECS Exec para depuração no Amazon Elastic Container Service Developer Guide.

Para abrir um shell interativo, execute o comando a seguir em sua máquina local.

```
aws ecs execute-command --cluster your_cluster_name --container your_container_name --
task task_id --interactive --command /bin/sh
```

Teste o PlanetsDemo aplicativo

Para verificar o status do PlanetsDemo aplicativo implantado, execute os comandos a seguir depois de substituir `load-balancer-DNS-name`, `listener-port`, e `web-binary-name` com os valores corretos para sua configuração.

```
curl http://load-balancer-DNS-name:listener-port/gapwalk-application/
```

Se o aplicativo estiver em execução, você verá a seguinte mensagem de saída:`Jics application is running`.

Em seguida, execute o comando a seguir.

```
curl http://load-balancer-DNS-name:listener-port/jac/api/services/rest/jicsservice/
```

Se o aplicativo estiver em execução, você verá a seguinte mensagem de saída:`Jics application is running`.

```
Jics application is running
```

Se você configurou o Blusam, pode esperar uma resposta vazia ao executar o comando a seguir.

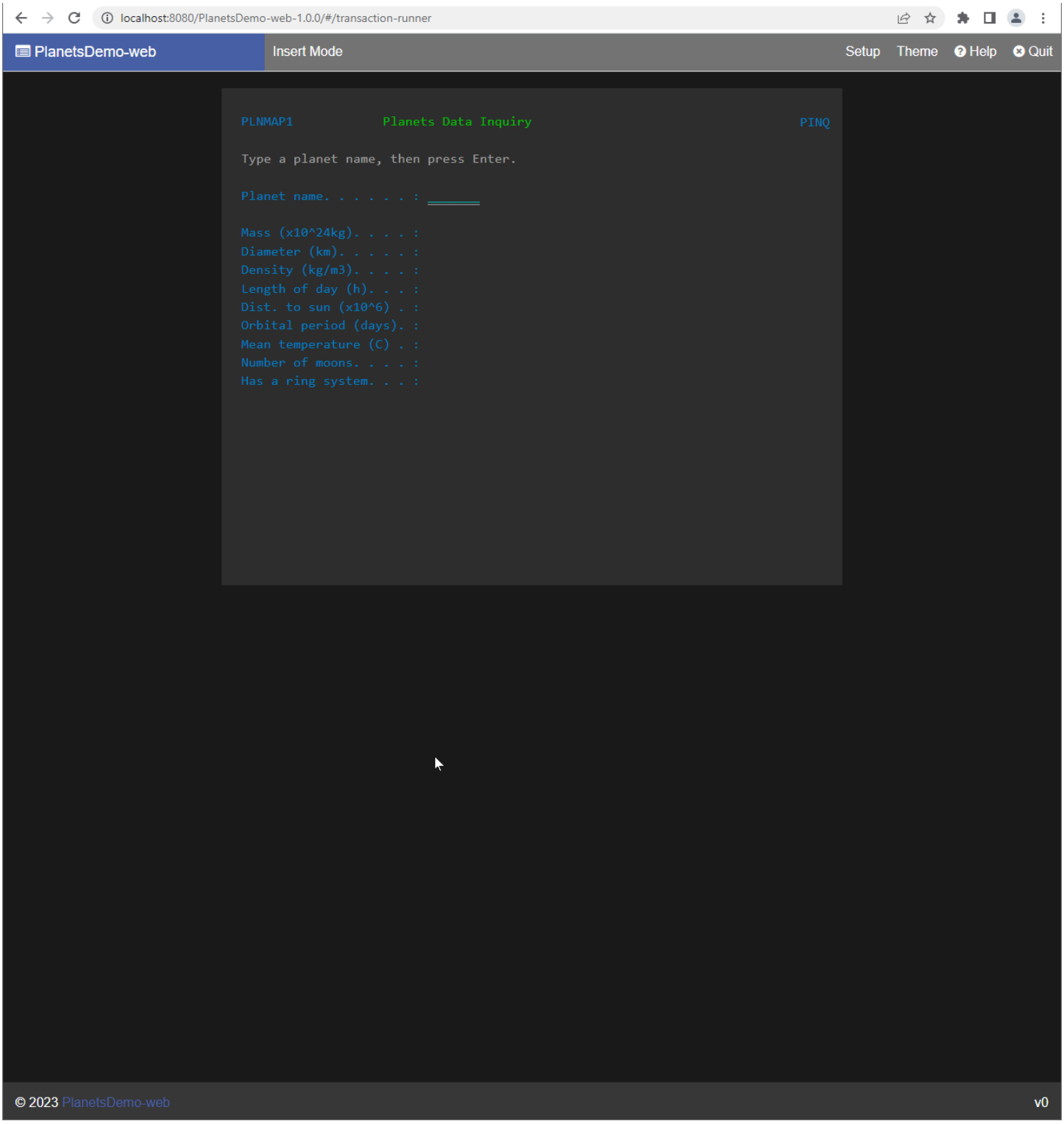
```
curl http://load-balancer-DNS-name:listener-port/bac/api/services/rest/bluesamserver/
serverIsUp
```

Observe o nome do binário da web (PlanetsDemo-web-1.0.0, se inalterado). Para acessar o PlanetsDemo aplicativo, use um URL com o seguinte formato.

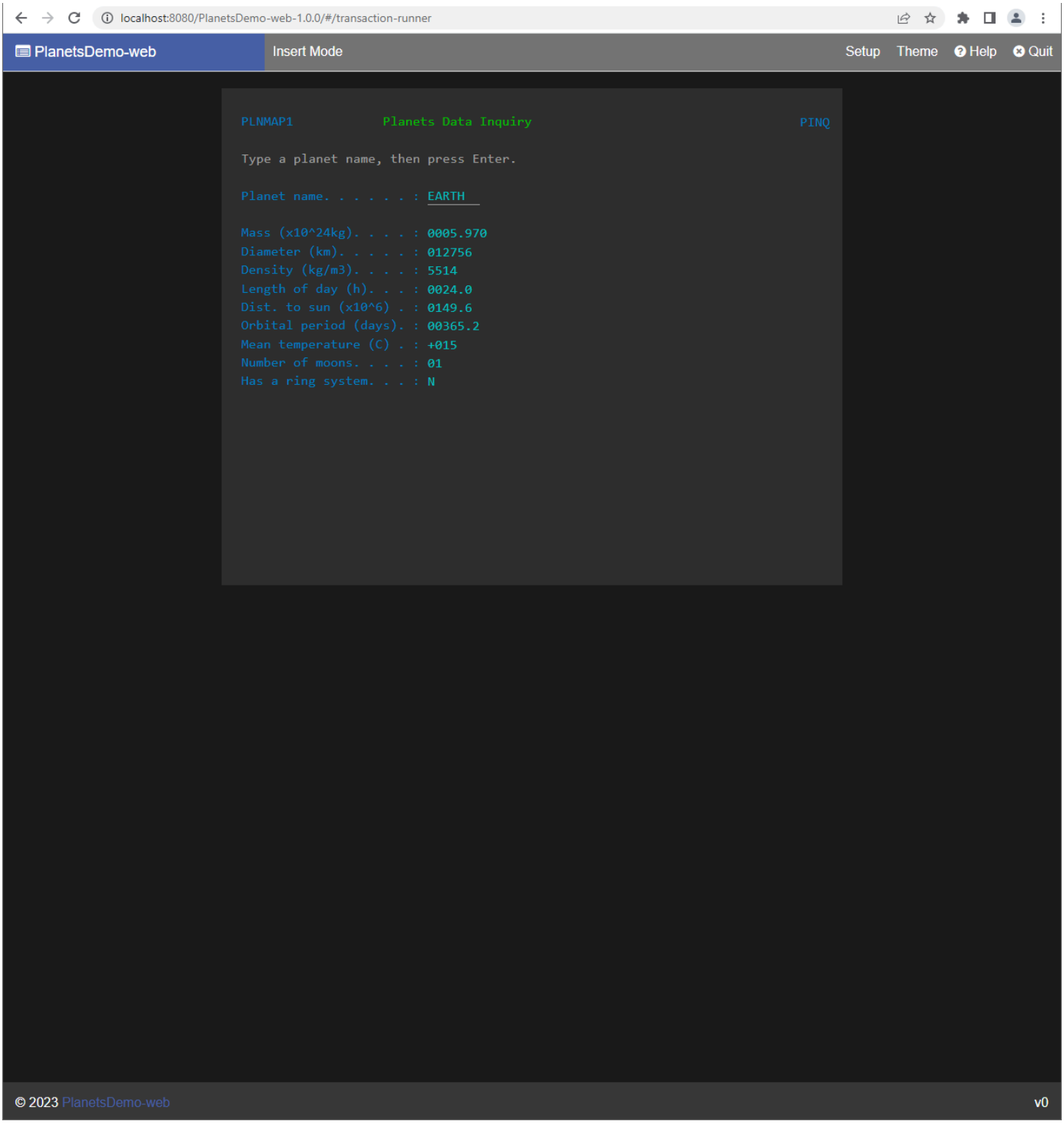
```
https://load-balancer-DNS-name:listener-port/web-binary-name
```

Depois que o PlanetsDemo aplicativo for iniciado, a página inicial será exibida.


Digite PINQ na caixa de texto e pressione Enter. A página de consulta de dados é exibida.




Por exemplo, insira EARTH no campo do PlanetsDemo nome e pressione Enter. A página do planeta que você inseriu é exibida.




## Atualizando o AWS Blu Age Runtime no Amazon ECS gerenciado por AWS Fargate

Este guia descreve como atualizar o AWS Blu Age Runtime no Amazon ECS gerenciado por. AWS Fargate

Tópicos



### Pré-requisitos

Antes de começar, verifique se você atende aos seguintes pré-requisitos.

- Preencha the section called “AWS Pré-requisitos do Blu Age Runtime”, e the section called “AWS Integração do Blu Age Runtime”.
- Baixe a versão do AWS Blu Age Runtime para a qual você deseja atualizar. Para ter mais informações, consulte the section called “AWS Integração do Blu Age Runtime”. A estrutura consiste em dois arquivos binários: `aws-bluage-runtime-x.x.x.x.tar.gz` `aws-bluage-webapps-x.x.x.x.tar.gz` e.

### Atualize o tempo de execução da velocidade

Conclua as etapas a seguir para atualizar o Velocity Runtime.

1. Reconstrua sua imagem do Docker com a versão desejada do AWS Blu Age Runtime. Para obter instruções, consulte the section called “Configurando o AWS Blu Age Runtime no Amazon ECS gerenciado por AWS Fargate”.
2. Envie sua imagem do Docker para o seu repositório Amazon ECR.
3. Pare e reinicie seu serviço Amazon ECS.
4. Verificar os logs.

O AWS Blu Age Runtime foi atualizado com sucesso.

## Amazon CloudWatch Alarms for AWS Blu Age Runtime no Amazon ECS gerenciado por AWS Fargate

Para ter notificações mais visíveis sempre que seus aplicativos implantados encontrarem exceções, configure CloudWatch para receber o registro do aplicativo e adicione um alarme para avisá-lo sobre possíveis erros.

### Configuração de alarme

Com CloudWatch os registros, você pode configurar qualquer número de métricas e alarmes, dependendo do seu aplicativo e das suas necessidades.

Especificamente, você pode configurar alarmes proativos para alertas de uso diretamente durante a criação do cluster do Amazon ECS, para que você seja notificado quando ocorrerem erros. Para destacar erros na conexão com o sistema de controle AWS Blu Age, adicione uma métrica referente à string “Erro C” nos registros. Depois, você poderá definir um alarme que reaja a essa métrica.

## Configurando dependências licenciadas no AWS Blu Age Runtime no Amazon ECS gerenciado por AWS Fargate

Este tópico descreve como configurar dependências licenciadas adicionais que você pode usar com o AWS Blu Age Runtime no Amazon ECS gerenciado pelo. AWS Fargate

Tópicos



### Pré-requisitos

Antes de começar, certifique-se de que você concluiu os seguintes pré-requisitos.

- Preencha the section called “AWS Pré-requisitos do Blu Age Runtime”, e the section called “AWS Integração do Blu Age Runtime”.
- Obtenha as seguintes dependências de sua fonte.

### Banco de dados Oracle

Forneça um driver de banco de dados Oracle. Por exemplo, ojdbc8-19.8.0.0.jar.

### Conexão IBM MQ

Forneça um cliente IBM MQ. Por exemplo, com.ibm.mq.allclient-9.3.0.15.jar.

Com essa versão de dependência, forneça também as seguintes dependências transitivas:

- javax.jms-api-2.0.1.jar
- json-20080701.jar

Arquivos de impressora DDS

Forneça a biblioteca de relatórios do Jasper. Por exemplo, jasperreports-6.16.0.jar, mas uma versão mais recente pode ser compatível.

Com essa versão de dependência, forneça também as seguintes dependências transitivas:

- castor-core-1.4.1.jar
- castor-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

Visão geral

Para instalar as dependências, conclua as etapas a seguir.

1. Copie qualquer uma das dependências acima, conforme necessário, para sua pasta de criação de imagens do Docker.
2. Se seu banco de dados JICS ou Blusam estiver hospedado no Oracle, forneça o driver do banco de dados Oracle. *your-tomcat-path*`/extra`
3. Em seu Dockerfile, copie essas dependências para. *your-tomcat-path*`/extra`
4. Crie sua imagem do Docker e, em seguida, envie-a para o Amazon ECR.
5. Pare e reinicie seu serviço Amazon ECS.
6. Verificar os logs.

## AWS Implantação do Blu Age Runtime no Amazon EC2

Os tópicos desta seção descrevem como configurar o AWS Blu Age Runtime (não gerenciado) no Amazon EC2, como atualizar a versão do tempo de execução, como monitorar sua implantação usando alarmes da CloudWatch Amazon e como adicionar dependências licenciadas.

Tópicos



## Configurando o AWS Blu Age Runtime (não gerenciado) no Amazon EC2

Este tópico explica como configurar e implantar o aplicativo de PlanetsDemo amostra usando o AWS Blu Age Runtime (não gerenciado) no Amazon EC2.

Tópicos



### Pré-requisitos

Antes de começar, certifique-se de que você concluiu os seguintes pré-requisitos.

- Configure o AWS CLI seguindo as etapas em Configuração da AWS CLI.
- Preencha the section called “AWS Pré-requisitos do Blu Age Runtime”, e the section called “AWS Integração do Blu Age Runtime”.
- Crie uma instância do Amazon EC2 que contenha o AWS Blu Age Runtime mais recente (no Amazon EC2). Para obter mais informações, consulte Introdução às instâncias do Amazon EC2 Linux.
- Certifique-se de que você possa se conectar à instância do Amazon EC2 com sucesso, por exemplo, usando SSM.
- Baixe e extraia o AWS Blu Age Runtime (no Amazon EC2) em. *your-tomcat-path*/* Para obter instruções, consulte the section called “AWS Integração do Blu Age Runtime”.
- Baixe o arquivo do PlanetsDemo aplicativo.
- Descompacte o arquivo e faça o upload do aplicativo para um bucket do Amazon S3 de sua escolha.

- Crie um banco de dados Amazon Aurora PostgreSQL para JICS e execute a consulta nele. `PlanetsDemo-v1/jics/sql/initJics.sql` Para obter informações sobre como criar um banco de dados Amazon Aurora PostgreSQL, consulte Criação e conexão com um cluster de banco de dados Aurora PostgreSQL.

### Configuração

Para configurar o aplicativo de PlanetsDemo amostra, conclua as etapas a seguir.

1. Conecte-se à sua instância do Amazon EC2 e vá até a `conf` pasta abaixo da pasta de instalação do Apache Tomcat 9. Abra o `catalina.properties` arquivo para edição e substitua a linha que começa `common.loader` com a seguinte linha.

```
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. Navegue para a pasta *<your-tomcat-path>*`/webapps`.
3. Copie os PlanetsDemo binários disponíveis na PlanetsDemo pasta -v1/webapps/ do bucket do Amazon S3 usando o comando a seguir.

```
aws s3 cp s3://path-to-demo-app-webapps/ . --recursive
```

> **Note**
>
> `path-to-demo-app-webapps`Substitua pelo URI correto do Amazon S3 para o bucket em que você descompactou o arquivo anteriormente. PlanetsDemo

4. Copie o conteúdo da pasta `PlanetsDemo-v1/config/` para *<your-tomcat-path>*`/config/`.
5. Forneça as informações de conexão do banco de dados que você criou como parte dos pré-requisitos no trecho a seguir no arquivo. `application-main.yml` Para obter mais informações, consulte Criação e conexão a um cluster Aurora PostgreSQL DB.

```
datasource:
   jicsDs:
     driver-class-name :
```

```
    url:
    username:
    password:
    type :
```

6. Inicie seu servidor Apache Tomcat e verifique os registros.

```
your-tomcat-path/startup.sh

tail -f your-tomcat-path/logs/catalina.log
```

   Se você encontrar códigos de erro que começam com C seguido por um número, como CXXXX, anote as mensagens de erro. Por exemplo, o código de erro C5102 é um erro comum que indica uma configuração incorreta da infraestrutura.

### Teste o PlanetsDemo aplicativo

Para verificar o status do PlanetsDemo aplicativo implantado, execute os comandos a seguir depois de substituir `load-balancer-DNS-namelistener-port`, e `web-binary-name` com os valores corretos para sua configuração.

```
curl http://load-balancer-DNS-name:listener-port/gapwalk-application/
```

Se o aplicativo estiver em execução, você verá a seguinte mensagem de saída:`Jics application is running`.

Em seguida, execute o comando a seguir.

```
curl http://load-balancer-DNS-name:listener-port/jac/api/services/rest/jicsservice/
```

Se o aplicativo estiver em execução, você verá a seguinte mensagem de saída:`Jics application is running`.

```
Jics application is running
```

Se você configurou o Blusam, pode esperar uma resposta vazia ao executar o comando a seguir.

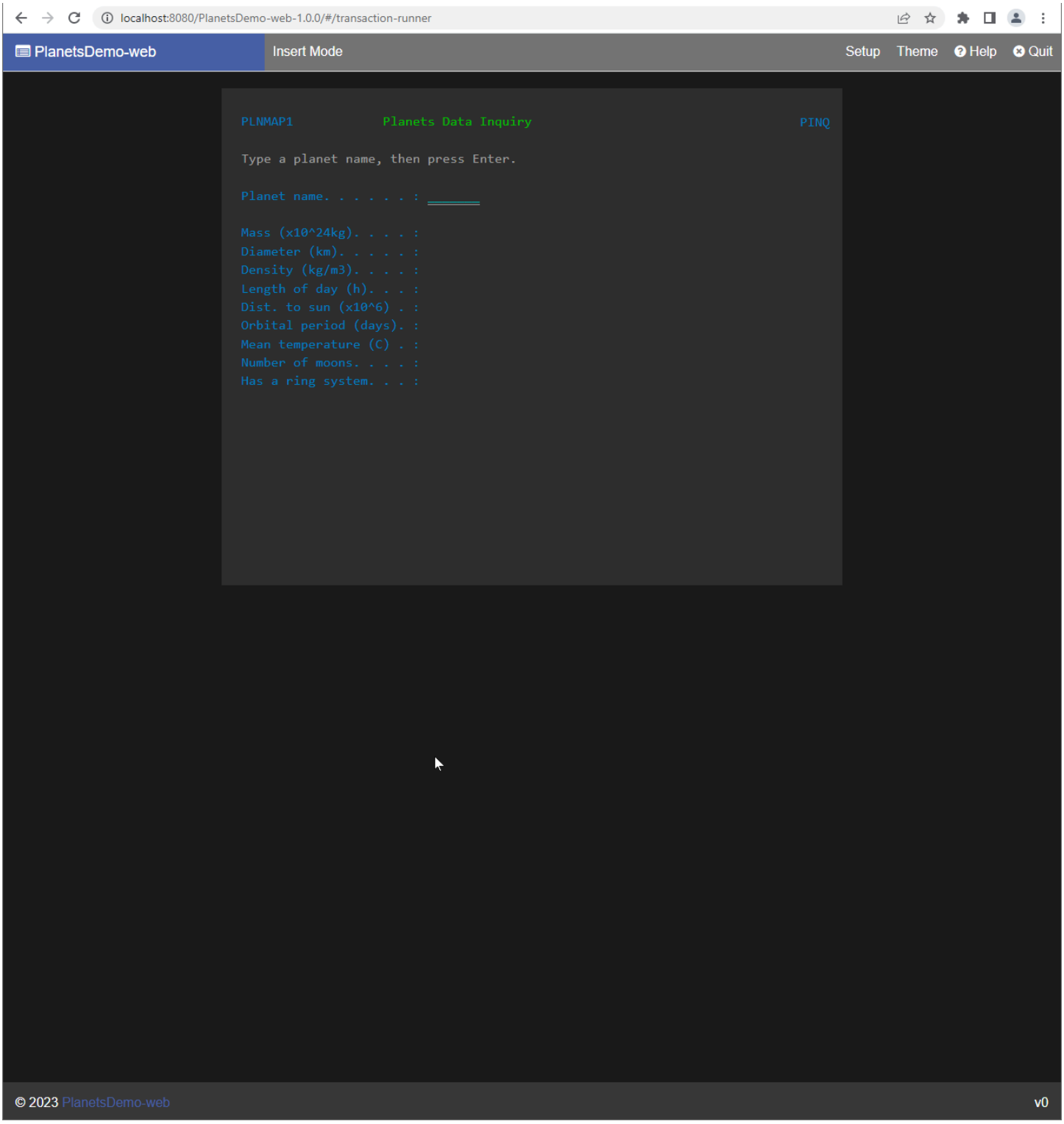
```
curl http://load-balancer-DNS-name:listener-port/bac/api/services/rest/bluesamserver/
serverIsUp
```

Observe o nome do binário da web (PlanetsDemo-web-1.0.0, se inalterado). Para acessar o PlanetsDemo aplicativo, use um URL com o seguinte formato.

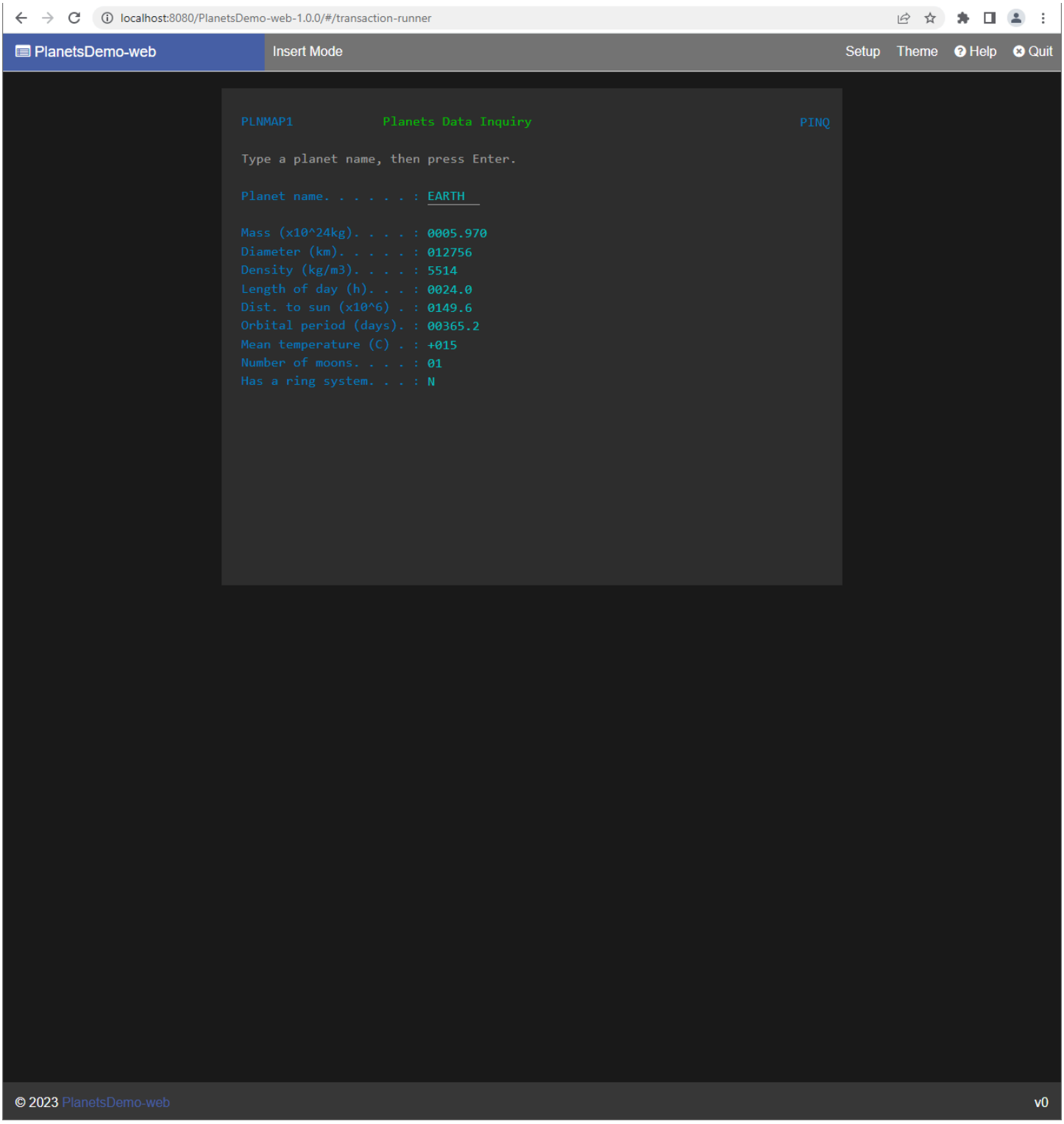
```
https://load-balancer-DNS-name:listener-port/web-binary-name
```

Depois que o PlanetsDemo aplicativo for iniciado, a página inicial será exibida.


Digite PINQ na caixa de texto e pressione Enter. A página de consulta de dados é exibida.




Por exemplo, insira EARTH no campo do PlanetsDemo nome e pressione Enter. A página do planeta que você inseriu é exibida.




## Atualizando o AWS Blu Age Runtime no Amazon EC2

Este guia descreve como atualizar o AWS Blu Age Runtime no Amazon EC2.

Tópicos



## Pré-requisitos

Antes de começar, verifique se você atende aos seguintes pré-requisitos.

- Preencha the section called “AWS Pré-requisitos do Blu Age Runtime”, e the section called “AWS Integração do Blu Age Runtime”.
- Certifique-se de ter uma instância do Amazon EC2 que contenha o tempo de execução mais recente do AWS Blu Age. Para obter mais informações, consulte Introdução às instâncias do Amazon EC2 Linux.
- Certifique-se de que você possa se conectar à instância do Amazon EC2 com sucesso, por exemplo, usando SSM.
- Baixe a versão do AWS Blu Age Runtime (no Amazon EC2) para a qual você deseja fazer o upgrade. Para obter mais informações, consulte the section called “AWS Configuração do Blu Age Runtime (não gerenciada)” A estrutura consiste em dois arquivos binários: `aws-bluage-runtime-x.x.x.x.tar.gz` `aws-bluage-webapps-x.x.x.x.tar.gz` e.

## Visão geral

Conclua as etapas a seguir para atualizar o Velocity Runtime.

1. Conecte-se à sua instância do Amazon EC2 e altere o usuário para su executando o comando a seguir.

   ```
   sudo su
   ```

   Você precisa do privilégio de superusuário para executar comandos neste tutorial.
2. Crie duas pastas, uma para cada arquivo binário.
3. Nomeie cada pasta usando o mesmo nome que o arquivo binário.
4. Copie cada arquivo binário para a pasta correspondente.

> **Warning**
>
> A extração de cada binário produz uma pasta com o mesmo nome. Portanto, se você extrair os dois arquivos binários no mesmo local, um após o outro, substituirá o conteúdo.

5. Para extrair os binários, use os seguintes comandos. Execute os comandos em cada pasta.

```
tar xvf aws-bluage-runtime-x.x.x.x.tar.gz
tar xvf aws-bluage-webapps-x.x.x.x.tar.gz
```

6. Pare os serviços do Apache Tomcat usando os seguintes comandos.

```
systemctl stop tomcat.service
systemctl stop tomcat-webapps.service
```

7. Substitua o conteúdo `<your-tomcat-path>/shared/` de pelo conteúdo de`aws-bluage-runtime-x.x.x.x/velocity/shared/`.
8. Substitua `<your-tomcat-path>/webapps/gapwalk-application.war` pelo `aws-bluage-runtime-x.x.x.x/velocity/webapps/gapwalk-application.war`.
9. Substitua os arquivos war em`<your-tomcat-path>/webapps/`, ou seja`jac.war`, `bac.war` e, pelos mesmos arquivos de`aws-bluage-webapps-x.x.x.x/velocity/webapps/`.
10. Inicie os serviços do Apache Tomcat executando os seguintes comandos.

```
systemctl start tomcat.service
systemctl start tomcat-webapps.service
```

11. Verificar os logs.

Para verificar o status do aplicativo implantado, execute os seguintes comandos.

```
curl http://localhost:8080/gapwalk-application/
```

A seguinte mensagem deve ser exibida.

```
Jics application is running
```

```
curl http://localhost:8181/jac/api/services/rest/jicsservice/
```

A seguinte mensagem deve ser exibida.

```
Jics application is running
```

```
curl http://localhost:8181/bac/api/services/rest/bluesamserver/serverIsUp
```

A resposta deve estar vazia.

O tempo de execução do AWS Blu Age foi atualizado com sucesso.

## AWS Blu Age Runtime (no Amazon EC2) Amazon Alarms CloudWatch

Para que você tenha notificações mais visíveis quando seus aplicativos implantados encontrarem exceções que colocarão seu aplicativo em um período de carência, você pode configurar CloudWatch o recebimento do registro do aplicativo e adicionar um alarme para avisá-lo sobre possíveis erros.

Implantação do CloudWatch registro

Por padrão, o arquivo `application-main.yml` inclui uma referência a outro arquivo de configuração de registro chamado `logback-cloudwatch.yml`.

```
logging:
  config: classpath:logback-cloudwatch.xml
```

Ambos os arquivos estão na pasta config e é assim que o CloudWatch registro é configurado, conforme explicado nas seções a seguir.

Configuração do CloudWatch registro

O arquivo `logback-cloudwatch.xml` tem o seguinte conteúdo.

```
<?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>
```

Tudo fora do <appender name="cloudwatch"/> elemento é uma configuração padrão de logback. Há dois anexadores neste arquivo: um anexador de console para enviar registros para o console e um CloudWatch anexador para o qual enviar registros. CloudWatch

O atributo `level` no elemento `root` especifica o nível de registro de todo o aplicativo.

Os valores necessários dentro da tag <appender name="cloudwatch"/> são:

- <logGroup/>:Define o nome do grupo de registros em CloudWatch. Se o valor não for especificado, o padrão será `BluAgeRuntimeOnEC2-Logs` Se o grupo de registros não existir, ele será criado automaticamente. Esse comportamento pode ser alterado por meio da configuração, que será abordada a seguir.
- <logStream/>: define o nome do LogStream (dentro do grupo de registros) em. CloudWatch

Valores opcionais:

- <region/>: Substitui a região na qual o fluxo de registro será gravado. Por padrão, os registros vão para a mesma região que a instância do EC2.
- <layout/>: o padrão que as mensagens de registro usarão.

- <maxbatchsize/>: o número máximo de mensagens de registro a serem enviadas CloudWatch por operação.
- <maxbatchtimemillis/>: o tempo em milissegundos para permitir que CloudWatch os registros sejam gravados.
- <maxqueuewaittimemillis/>: o tempo em milissegundos para tentar inserir solicitações na fila de registros interna.
- <internalqueuesize/>: o tamanho máximo da fila interna.
- <createlogdests/>: crie um grupo de registros e um fluxo de registros, se eles não existirem.
- <initialwaittimemillis/>: a quantidade de tempo em que você deseja que o tópico permaneça suspenso na inicialização. Essa espera inicial permite um acúmulo inicial de registros.
- <maxeventmessagesize/>: o tamanho máximo de um evento de registro. Os registros que excederem esse tamanho não serão enviados.
- <truncateeventmessages/>: trunque as mensagens que são muito longas.
- <printrejectedevents/>: Ative o anexador de emergência.

### CloudWatch configuração

Para que a configuração acima envie corretamente os registros para CloudWatch, atualize sua função de perfil de instância IAM do Amazon EC2 para conceder permissões adicionais para o grupo de log `BluAgeRuntimeOnEC2-Logs` e seus fluxos de log:

- `logs:CreateLogStream`
- `logs:DescribeLogStreams`
- `logs:CreateLogGroup`
- `logs:PutLogEvents`
- `logs:DescribeLogGroups`

### Configuração de alarme

Graças aos CloudWatch registros, você pode configurar diferentes métricas e alarmes, dependendo do seu aplicativo e de suas necessidades. Especificamente, você pode configurar alarmes proativos para alertas de uso, para ser avisado no caso de erros que possam colocar a aplicação em um período de carência (e, no final, impedir que ele funcione). Para fazer isso, você pode adicionar uma métrica relacionada à string “Erro C5001" nos registros, que destaca os erros na conexão com o sistema de controle AWS Blu Age. Depois, você poderá definir um alarme que reaja a essa métrica.

# Configurando dependências licenciadas no AWS Blu Age Runtime no Amazon EC2

Este guia descreve como configurar dependências licenciadas adicionais que você pode usar com o AWS Blu Age Runtime no Amazon EC2.

Tópicos



## Pré-requisitos

Antes de começar, certifique-se de que você concluiu os seguintes pré-requisitos.

- Preencha the section called “AWS Pré-requisitos do Blu Age Runtime”, e the section called “AWS Integração do Blu Age Runtime”.
- Certifique-se de ter uma instância do Amazon EC2 contendo a versão mais recente do AWS Blu Age Runtime (no Amazon EC2). Para obter mais informações, consulte Introdução às instâncias do Amazon EC2 Linux.
- Certifique-se de que você possa se conectar à instância do Amazon EC2 com sucesso, por exemplo, usando SSM.
- Obtenha as seguintes dependências de suas fontes.

### Banco de dados Oracle

Forneça um driver de banco de dados Oracle. Testamos a funcionalidade do AWS Blu Age Runtime (no Amazon EC2) com a versão ojdbc8-19.8.0.0.jar, mas uma versão mais recente pode ser compatível.

### Conexão IBM MQ

Forneça um cliente IBM MQ. Testamos a funcionalidade do AWS Blu Age Runtime (no Amazon EC2) com a versão com.ibm.mq.allclient-9.3.0.15.jar, mas uma versão mais recente pode ser compatível.

Com essa versão de dependência, forneça também as seguintes dependências transitivas:

- javax.jms-api-2.0.1.jar
- json-20080701.jar

Arquivos de impressora DDS

Forneça a biblioteca de relatórios do Jasper. Testamos a funcionalidade do AWS Blu Age Runtime (no Amazon EC2) com jasperreports-6.16.0.jar, mas uma versão mais recente pode ser compatível.

Com essa versão de dependência, forneça também as seguintes dependências transitivas:

- castor-core-1.4.1.jar
- castor-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

Visão geral

Para instalar as dependências, conclua as etapas a seguir.

1. Conecte-se à sua instância do Amazon EC2 e altere o usuário para su executando o comando a seguir.

   ```
   sudo su
   ```

   Você precisa do privilégio de superusuário para executar comandos neste tutorial.
2. Navegue para a pasta `<your-tomcat-path>/extra/`.

   ```
   cd <your-tomcat-path>/extra/
   ```
3. Copie qualquer uma das dependências acima conforme necessário nesta pasta.
4. Pare e inicie o tomcat.service executando os seguintes comandos.

   ```
   systemctl stop tomcat.service
   ```

   ```
   systemctl start tomcat.service
   ```

5. Verifique o status do serviço para se certificar de que ele está sendo executado.

```
systemctl status tomcat.service
```

6. Verificar os logs.

Configurar as dependências para aplicativos web JAC e BAC

1. Se o seu banco de dados JICS ou Blusam estiver hospedado no Oracle, será necessário fornecer o driver do banco de dados Oracle em `<your-tomcat-path>/extra`.
2. Crie a pasta se ela ainda não estiver presente.
3. Pare e reinicie seu servidor Apache Tomcat.
4. Verificar os logs.

# Modifique o código-fonte com o Blu Age Developer IDE

Se você estiver usando o mecanismo de tempo AWS de execução AWS Blu Age gerenciado, poderá usar o Blu Age Developer para modificar o código-fonte gerado. Talvez você queira fazer isso se precisar atualizar o código modernizado por algum motivo ou se uma parte do código-fonte antigo não puder ser modernizada. Você acessa o Blu Age Developer por meio da Amazon AppStream 2.0. Esta seção descreve como configurar o Blu Age Developer na AppStream versão 2.0. Também explica como usar o Blu Age Developer para atualizar o código-fonte usando o aplicativo PlanetsDemo de amostra.

Tópicos

- Tutorial: Configurar AppStream 2.0 para AWS Blu Age Developer IDE
- Tutorial: Use o AWS Blu Age Developer na versão 2.0 AppStream

## Tutorial: Configurar AppStream 2.0 para AWS Blu Age Developer IDE

AWS A modernização do mainframe fornece várias ferramentas por meio do Amazon AppStream 2.0. AppStream 2.0 é um serviço de streaming de aplicativos totalmente gerenciado e seguro que permite transmitir aplicativos de desktop para usuários sem reescrever aplicativos. AppStream O 2.0 fornece aos usuários acesso instantâneo aos aplicativos de que precisam, com uma experiência de usuário responsiva e fluida no dispositivo de sua escolha. O uso da AppStream versão 2.0 para hospedar ferramentas específicas do Runtime Engine oferece às equipes de aplicativos do cliente

a capacidade de usar as ferramentas diretamente de seus navegadores da web, interagindo com arquivos de aplicativos armazenados em buckets ou repositórios do Amazon S3. CodeCommit

Para obter informações sobre o suporte a navegadores na AppStream versão 2.0, consulte System Requirements and Feature Support (Web Browser) no Amazon AppStream 2.0 Administration Guide. Se você tiver problemas ao usar a AppStream versão 2.0, consulte Solução de problemas do usuário AppStream 2.0 no Guia de administração da Amazon AppStream 2.0.

Este documento descreve como configurar o AWS Blu Age Developer IDE em uma frota AppStream 2.0.

Tópicos



## Pré-requisito

Baixe o arquivo que contém os artefatos necessários para configurar o AWS Blu Age Developer IDE em AppStream 2.0.

> **Note**
>
> Esse é um arquivo grande. Se você tiver problemas com o tempo limite da operação, recomendamos usar uma instância do Amazon EC2 para melhorar o desempenho de upload e download.

## Etapa 1: Crie um bucket do Amazon S3

Crie um bucket do Amazon S3 da Região da AWS mesma forma que a frota AppStream 2.0 que você criará. Esse bucket conterá os artefatos necessários para você concluir este tutorial.

## Etapa 2: anexar uma política ao bucket S3

Anexe a política a seguir ao bucket que você criou para este tutorial. Certifique-se de `MYBUCKET` substituir pelo nome real do bucket que você criou.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "AllowAppStream2.0ToRetrieveObjects",
        "Effect": "Allow",
        "Principal": {
            "Service": "appstream.amazonaws.com"
        },
        "Action": "s3:GetObject",
        "Resource": "arn:aws:s3:::MYBUCKET/*"
    }]
}
```

## Etapa 3: Faça upload de arquivos para o bucket do Amazon S3

Descompacte os arquivos que você baixou no Pré-requisito e faça o upload da pasta `appstream` no seu bucket. O upload dessa pasta cria a estrutura correta no seu bucket. Para obter mais informações, consulte Upload de objetos no Guia do usuário do Amazon S3.

## Etapa 4: baixar AWS CloudFormation modelos

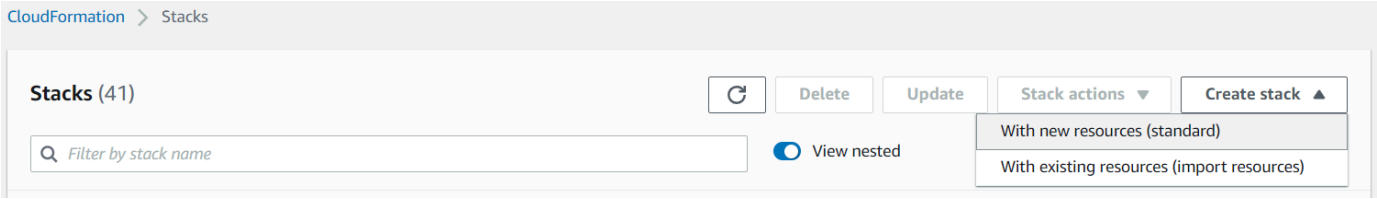
Baixe os AWS CloudFormation modelos a seguir. Você precisa desses modelos para criar e preencher a frota AppStream 2.0.

- 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

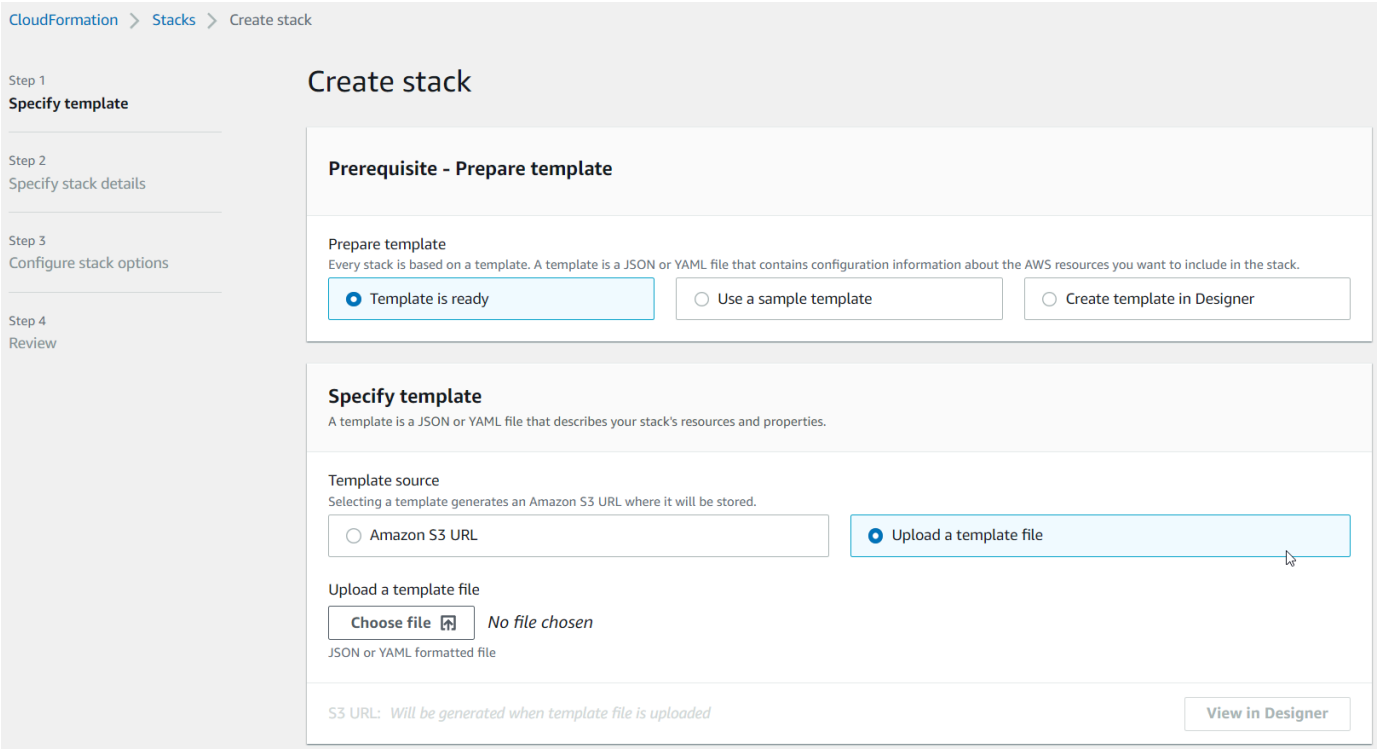
## Etapa 5: Crie a frota com AWS CloudFormation

Nesta etapa, você usa o `cfn-m2-appstream-elastic-fleet-linux.yaml` AWS CloudFormation modelo para criar uma frota AppStream 2.0 e uma pilha para hospedar o AWS Blu Age Developer IDE. Depois de criar a frota e a pilha, você executará os outros AWS CloudFormation modelos baixados na etapa anterior para instalar o Developer IDE e outras ferramentas necessárias.

1. Navegue até AWS CloudFormation o console AWS de gerenciamento e escolha Pilhas.
2. Em Pilhas, selecione Criar pilha e Com novos recursos (padrão):




3. Em Criar pilha, escolha O modelo está pronto e fazer o upload de um arquivo de modelo:




4. Escolha Escolher arquivo,e navegue até o arquivo `cfn-m2-appstream-elastic-fleet-linux.yaml`. Escolha Próximo.
5. Na página Especificar detalhes do StackSet, forneça as seguintes informações.
   - Um nome para a pilha.
   - Seu grupo de segurança padrão e duas sub-redes desse grupo de segurança.

> **Note**
>
> As duas sub-redes do grupo de segurança devem estar em zonas de disponibilidade diferentes.

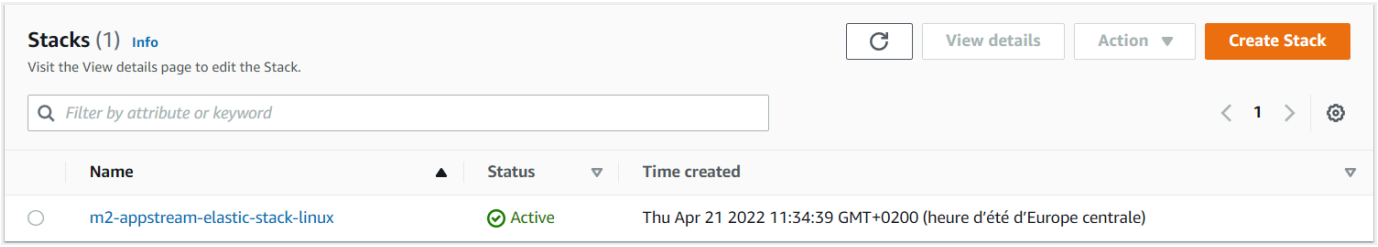
6. Selecione Próximo e Próximo novamente.
7. Escolha Eu reconheço que AWS CloudFormation pode criar recursos do IAM com nomes personalizados. e, em seguida, escolha Enviar.
8. Depois de criar a frota, crie CloudFormation pilhas com os outros modelos baixados para concluir a configuração dos aplicativos. Certifique-se de atualizar BucketNamesempre para apontar para o bucket correto do S3. Você pode editar o BucketNameno CloudFormation console. Como alternativa, você pode editar os arquivos do modelo diretamente e atualizar a `S3Bucket` propriedade.

> **Note**
>
> Os modelos baixados esperam encontrar ativos em um bucket do S3 com uma estrutura de pastas chamada`appstream/bluage/developer-ide/`. O bucket deve estar na Região da AWS mesma frota que você criou.

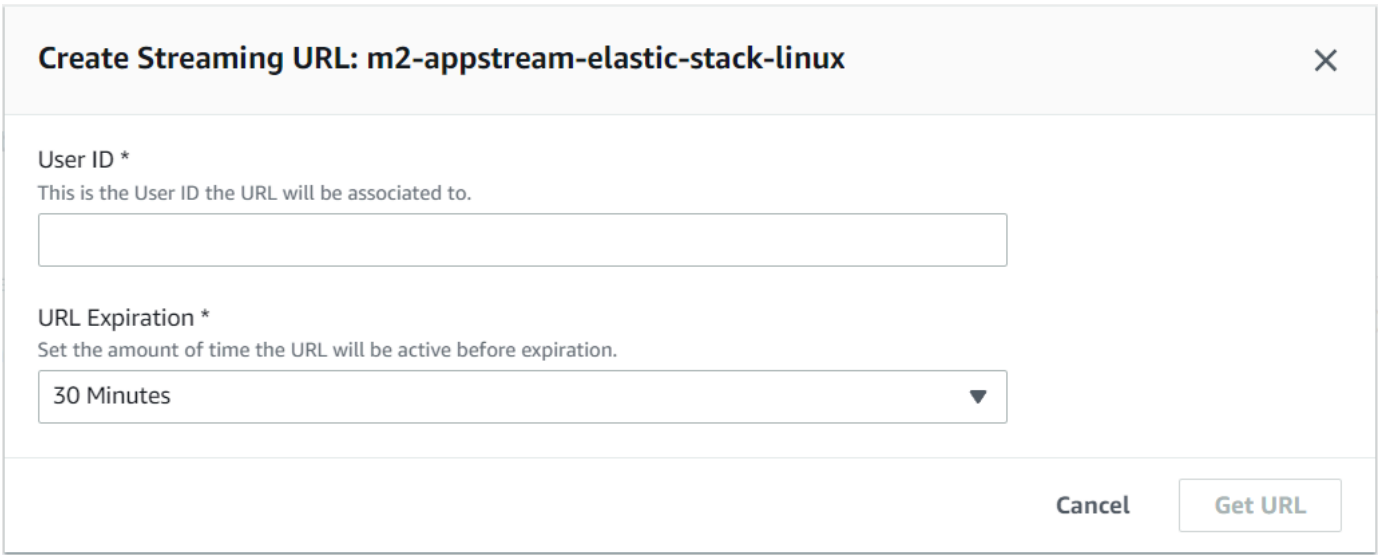
## Etapa 6: acessar uma instância

Depois de criar e iniciar a frota, você pode criar um link temporário para acessar a frota por meio do cliente nativo.

1. Navegue até AppStream 2.0 no AWS Management Console e escolha a pilha criada anteriormente:




2. Na página de detalhes da pilha, escolha Ação e, em seguida, escolha Criar URL de streaming:




3. Em Criar URL de streaming, insira um ID de usuário arbitrário e um prazo de validade do URL e escolha Obter URL. Você obtém uma URL que pode ser usada para transmitir para um navegador ou para o cliente nativo. Recomendamos que você transmita para o cliente nativo.

## Limpeza de recursos

Para o procedimento de limpeza da pilha e das frotas criadas, consulte Criar uma frota e uma pilha AppStream 2.0.

Depois de excluir os objetos AppStream 2.0, você ou o administrador da conta também podem limpar os buckets do S3 para configurações do aplicativo e pastas pessoais.

> **Note**
>
> A pasta inicial de um determinado usuário é exclusiva em todas as frotas, portanto, talvez seja necessário mantê-la se outras pilhas AppStream 2.0 estiverem ativas na mesma conta.

Você não pode usar o console AppStream 2.0 para excluir usuários. Em vez disso, você deve usar a API do serviço com AWS CLI o. Para obter mais informações, consulte Administração de grupos de usuários no Guia de administração da Amazon AppStream 2.0.

# Tutorial: Use o AWS Blu Age Developer na versão 2.0 AppStream

Este tutorial mostra como acessar o AWS Blu Age Developer na AppStream versão 2.0 e usá-lo com um aplicativo de amostra para que você possa experimentar os recursos. Ao concluir este tutorial, você poderá usar as mesmas etapas em seus próprios aplicativos.

Tópicos



## Etapa 1: criar um banco de dados

Nesta etapa, você usa o Amazon RDS para criar um banco de dados PostgreSQL gerenciado que o aplicativo de demonstração usa para armazenar informações de configuração.

1. Abra o console do Amazon RDS.
2. Escolha Bancos de dados > Criar banco de dados.
3. Escolha Standard create > PostgreSQL, deixe a versão padrão e escolha Free tier.
4. Escolha um identificador de instância de BD.
5. Em Configurações de credenciais, escolha Gerenciar credenciais mestre em AWS Secrets Manager. Para obter mais informações, consulte Gerenciamento de senhas com Amazon RDS e o AWS Secrets Manager no Guia do usuário do Amazon RDS.
6. Certifique-se de que a VPC seja a mesma que você usa para a instância AppStream 2.0. Você pode solicitar esse valor ao administrador.
7. Para o grupo de segurança VPC, escolha Criar novo.

8. Defina Acesso público como Sim.
9. Deixe todos os outros valores padrão. Analise esses valores.
10. Selecione Criar banco de dados.

Para tornar o servidor de banco de dados acessível a partir da sua instância, selecione o servidor de banco de dados no Amazon RDS. Em Conectividade e segurança, escolha o grupo de segurança VPC para o servidor de banco de dados. Esse grupo de segurança foi criado anteriormente para você e deve ter uma descrição semelhante à do console de gerenciamento Criado pelo RDS. Escolha Ação > Editar regras de entrada, escolha Adicionar regra e crie uma regra do tipo PostgreSQL. Para fonte de regra, use o grupo de segurança padrão. Você pode começar a digitar o nome da fonte no campo Fonte e, em seguida, aceitar a ID sugerida. Por fim, escolha Salvar regras.

## Etapa 2: acessar o ambiente

Nesta etapa, você acessa o ambiente de desenvolvimento AWS Blu Age na AppStream versão 2.0.

1. Entre em contato com seu administrador para saber a forma correta de acessar sua instância AppStream 2.0. Para obter informações gerais sobre possíveis clientes e configurações, consulte AppStream 2.0 Access Methods and Clients no Amazon AppStream 2.0 Administration Guide. Considere usar o cliente nativo para obter a melhor experiência.
2. Na AppStream versão 2.0, escolha Desktop.

## Etapa 3: Configurar o tempo de execução

Nesta etapa, você configura o tempo de execução do AWS Blu Age. Você deve configurar o tempo de execução na primeira inicialização e novamente se for notificado sobre uma atualização do tempo de execução. Essa etapa preenche sua pasta `.m2`.

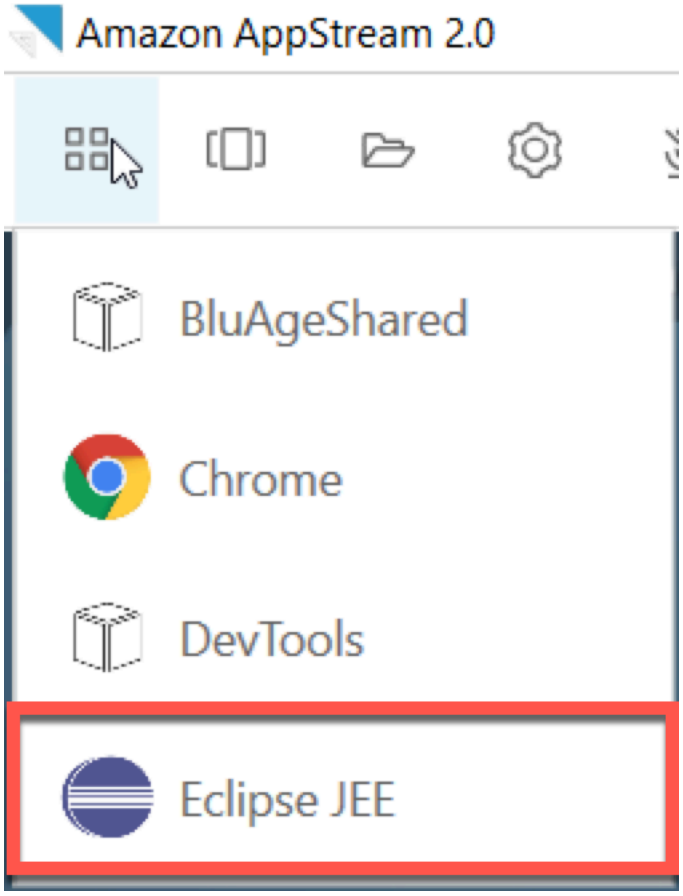
1. Escolha Aplicativos, na barra de menu, e escolha Terminal.
2. Digite o comando :

```
~/_install-velocity-runtime.sh
```

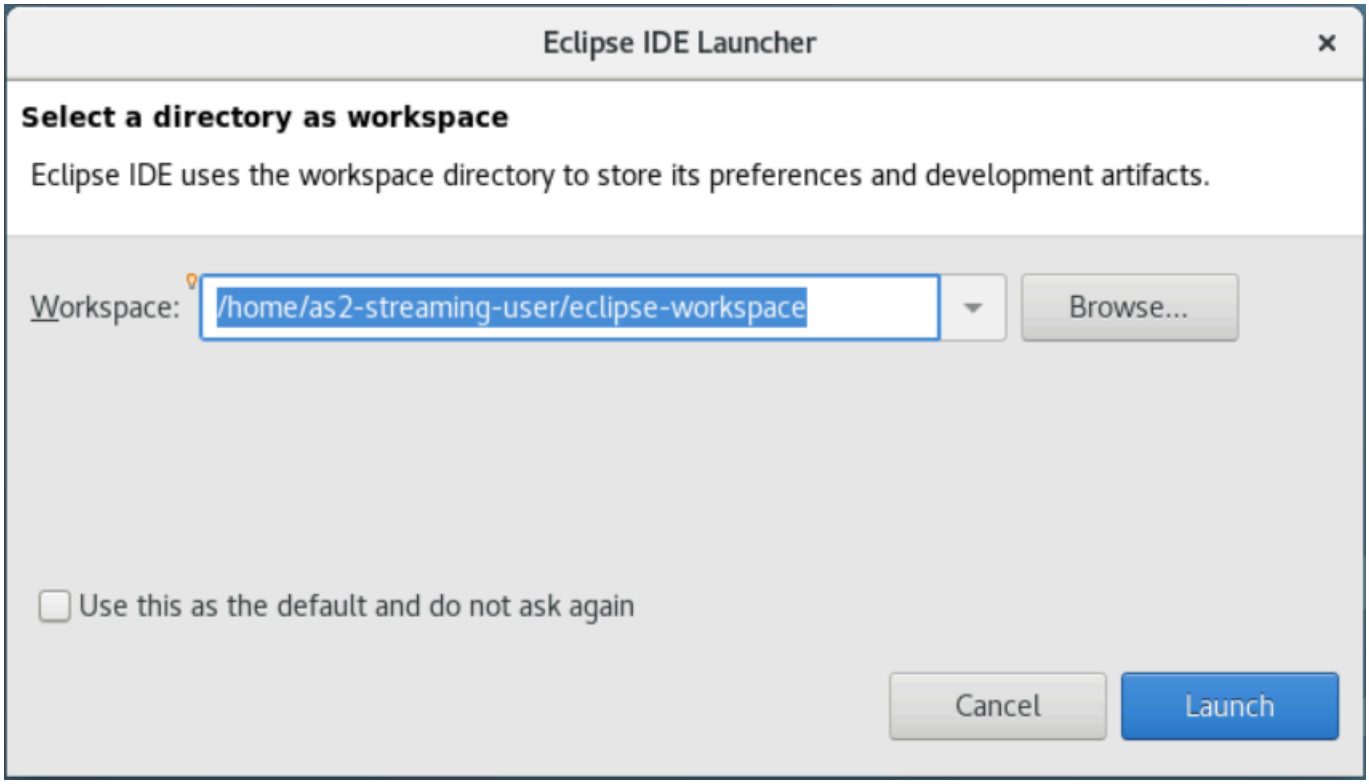
## Etapa 4: Iniciar o Eclipse IDE

Nesta etapa, você inicia o Eclipse IDE e escolhe um local onde deseja criar um espaço de trabalho.

1. Na AppStream versão 2.0, escolha o ícone Launch Application na barra de ferramentas e, em seguida, escolha Eclipse JEE.




2. Quando o inicializador abrir, insira o local em que você deseja criar seu espaço de trabalho e escolha Iniciar.




Opcionalmente, você pode iniciar o Eclipse a partir da linha de comando, da seguinte forma:

```
~/eclipse &
```

## Etapa 5: Configurar um projeto Maven

Nesta etapa, você importará um projeto em Maven para o aplicativo de demonstração Planets.

1. Faça PlanetsDemoo upload de -pom.zip para sua pasta inicial. Você pode usar o recurso “Meus arquivos” do cliente nativo para fazer isso.
2. Use a ferramenta de linha de `unzip` comando para extrair os arquivos.
3. Navegue dentro da pasta descompactada e abra a raiz `pom.xml` do seu projeto em um editor de texto.
4. Edite a `gapwalk.version` propriedade para que ela corresponda ao tempo de execução do AWS Blu Age instalado.

   Se não tiver certeza da versão instalada, emita o seguinte comando em um terminal:

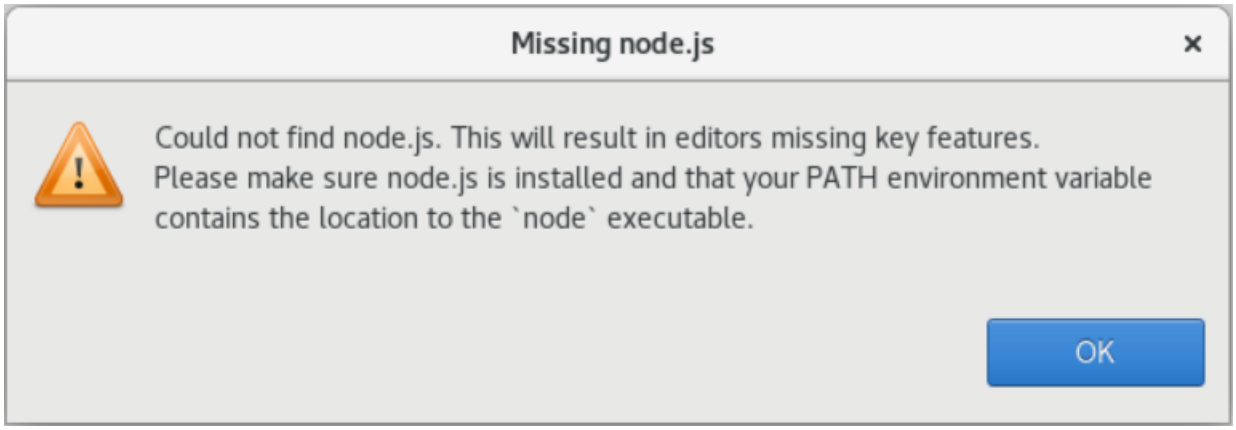
   ```
   cat ~/runtime-version.txt
   ```

Esse comando imprime a versão de tempo de execução atualmente disponível, por exemplo,`3.1.0-b3257-dev`.

> **Note**
>
> Não inclua o sufixo `-dev` em `gapwalk.version`. Por exemplo, um valor válido seria `<gapwalk.version>3.1.0-b3257</gapwalk.version>`.

5. Em Eclipse, escolha Arquivo e Importar. Na janela de diálogo Importar, expanda Maven e escolha Projetos existentes do Maven. Escolha Próximo.
6. Em Importar projetos do Maven, forneça a localização dos arquivos extraídos e escolha Concluir.

   Você pode ignorar isso com segurança. O Maven baixa uma cópia local do `node.js` para criar a parte Angular (*-web) do projeto:

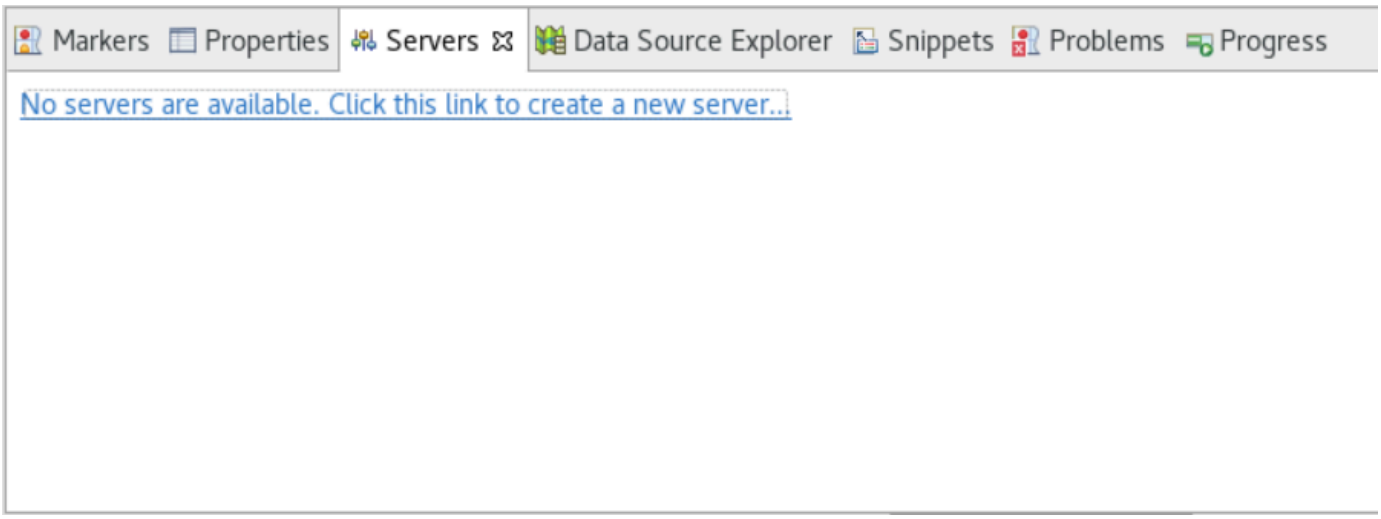
   
   

   Espere até o final da compilação. É possível acompanhar a compilação na visualização Progresso.
7. No Eclipse, selecione o projeto e escolha Executar como. Em seguida, escolha a instalação do Maven. Depois que a instalação do Maven for bem-sucedida, ele criará o `war` arquivo abaixo. `PlanetsDemoPom/PlanetsDemo-web/target/PlanetsDemo-web-1.0.0.war`

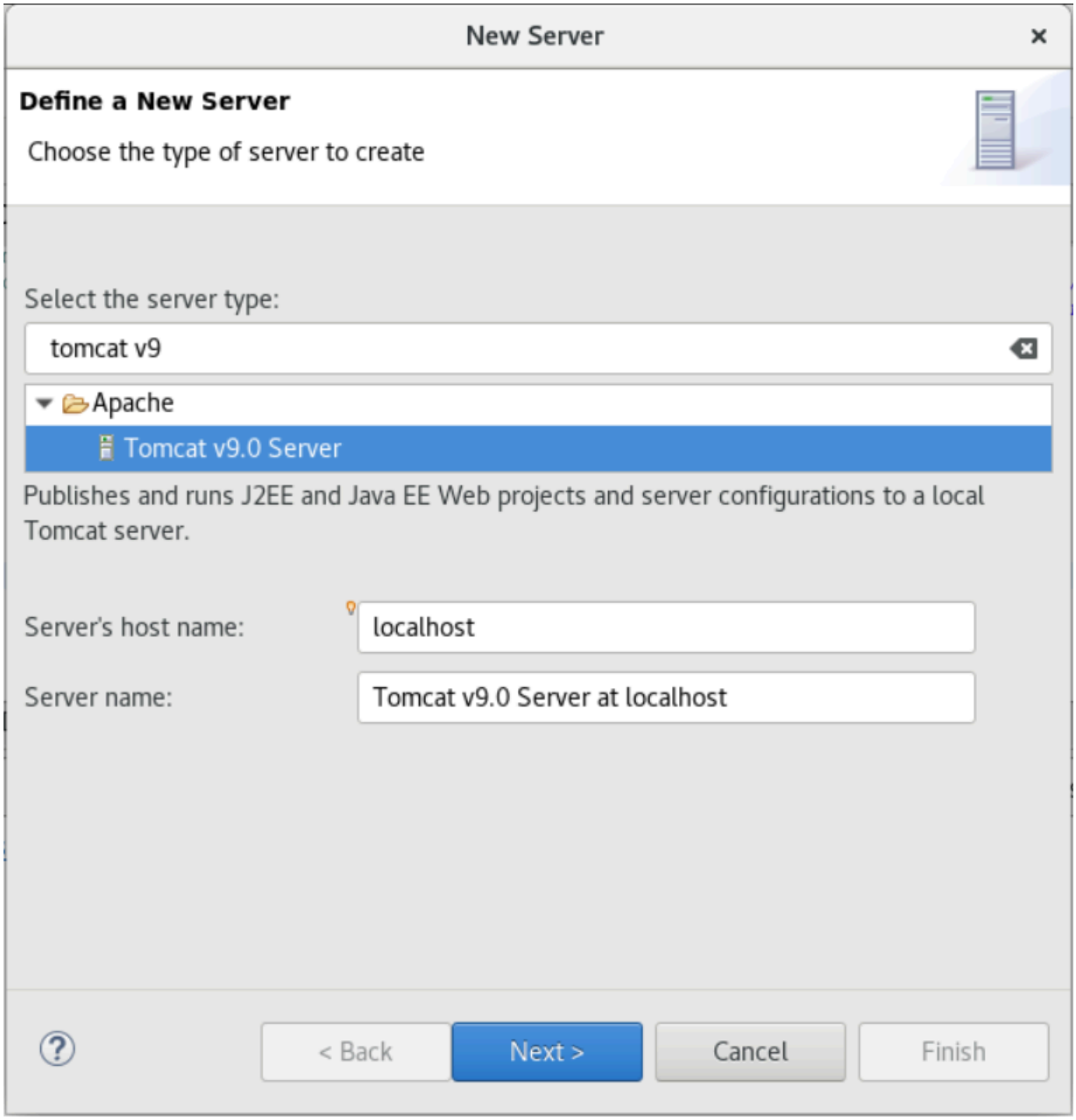
## Etapa 6: configurar um servidor Tomcat

Nesta etapa, você configura um servidor Tomcat onde você implanta e inicia seu aplicativo compilado.

1. No Eclipse, escolha Janela > Mostrar visualização > Servidores para mostrar a visualização Servidores:

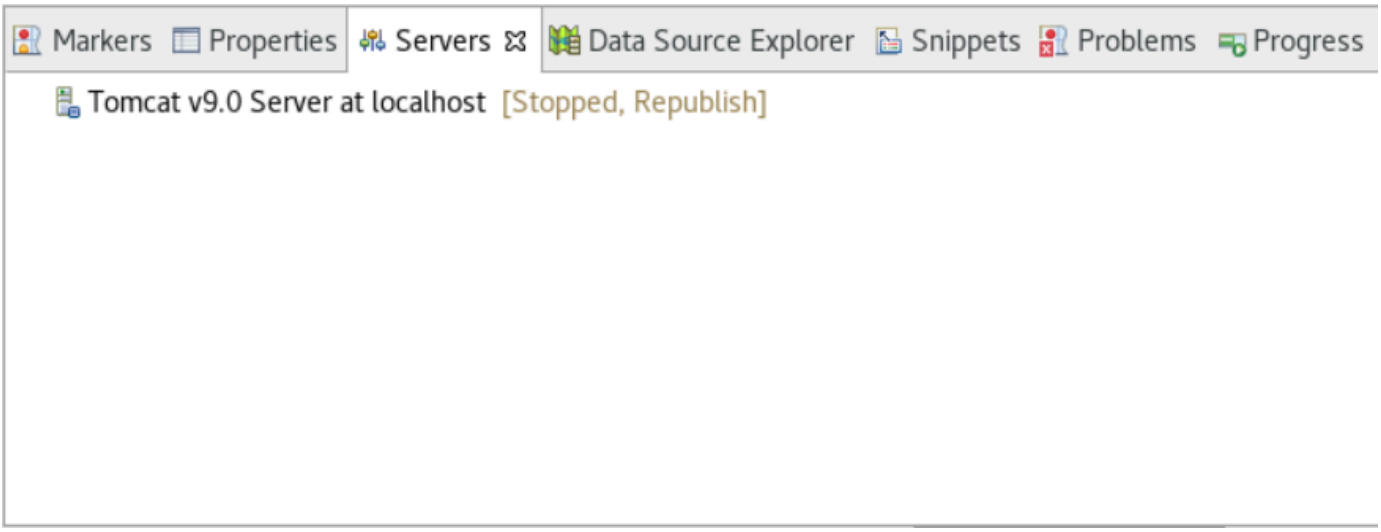



2. Escolha Nenhum servidor está disponível. Clique neste link para criar um novo servidor... . O assistente de Novo Servidor é exibido. No campo Selecionar o tipo de servidor do assistente, digite tomcat v9 e escolha Servidor Tomcat v9.0. Em seguida, escolha Próximo.




3. Escolha Procurar e escolha a pasta tomcat na raiz da pasta inicial. Deixe o JRE em seu valor padrão e escolha Concluir.

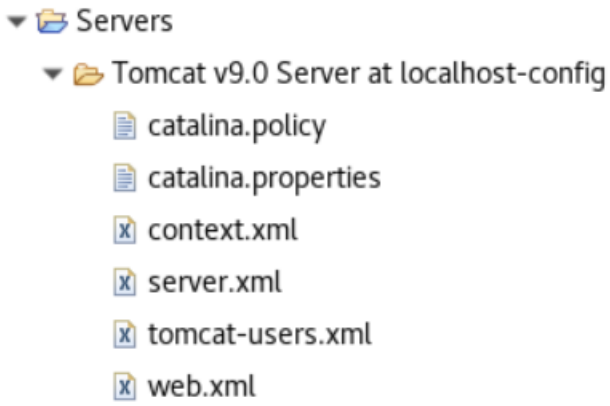
   Um projeto de servidores é criado no espaço de trabalho, e um servidor Tomcat v9.0 agora está disponível na visualização de servidores. É aqui que o aplicativo compilado será implantado e iniciado:



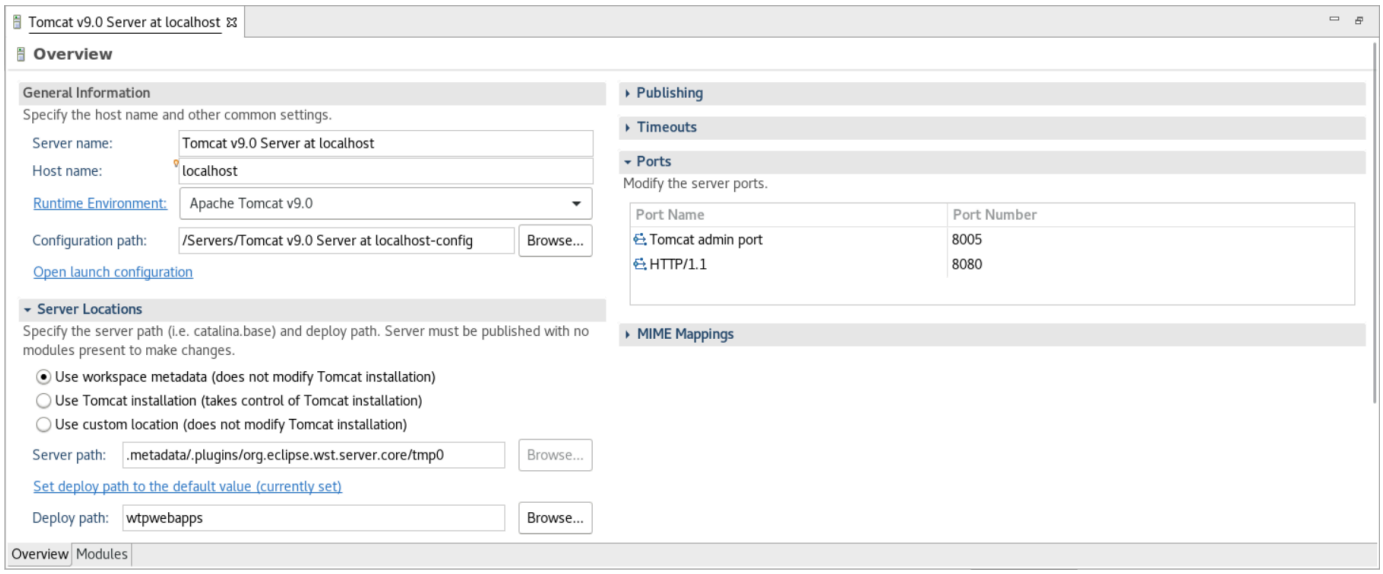

## Etapa 7: Implantar no Tomcat

Nesta etapa, você implantará o aplicativo de demonstração Planets no servidor Tomcat para poder executar o aplicativo.

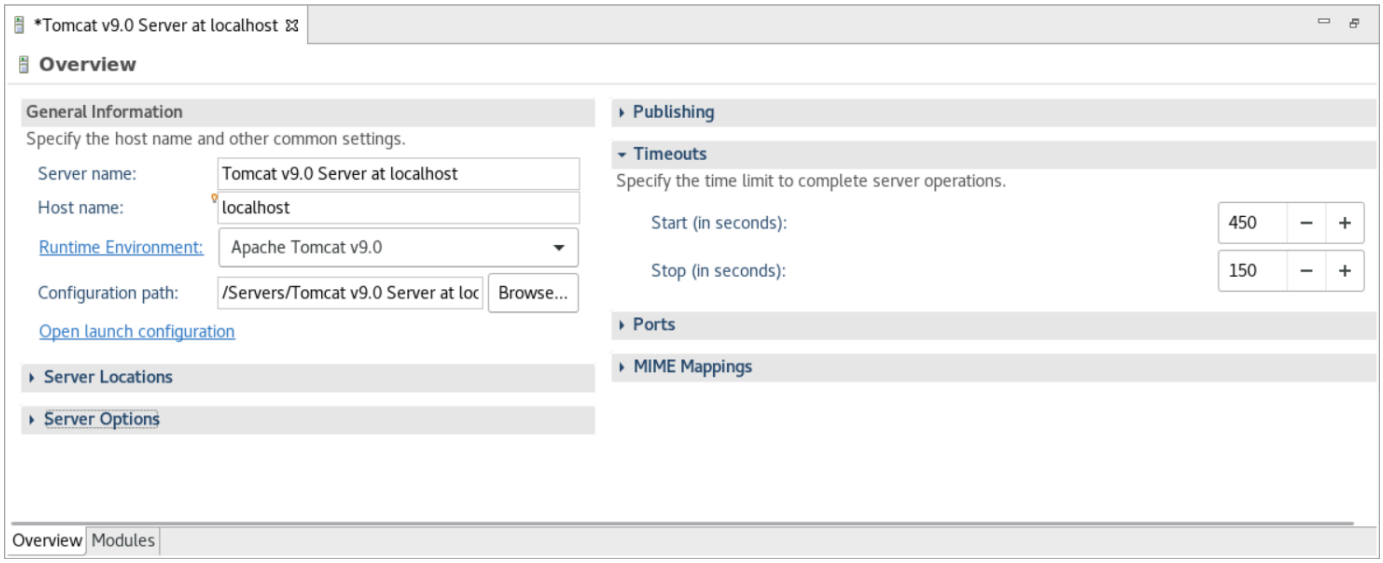
1. Selecione o `PlanetsDemo-web` arquivo e escolha Executar como > Instalação do Maven. Selecione `PlanetsDemo-web` novamente e escolha Atualizar para garantir que o frontend compilado com npm seja compilado corretamente em um .war e seja notado pelo Eclipse.
2. Faça upload do PlanetsDemo-runtime.zip na instância e descompacte o arquivo em um local acessível. Isso garante que o aplicativo de demonstração possa acessar as pastas e arquivos de configuração necessários.
3. Copie o conteúdo de `PlanetsDemo-runtime/tomcat-config` na `Servers/Tomcat v9.0...` subpasta que você criou para o seu servidor Tomcat:



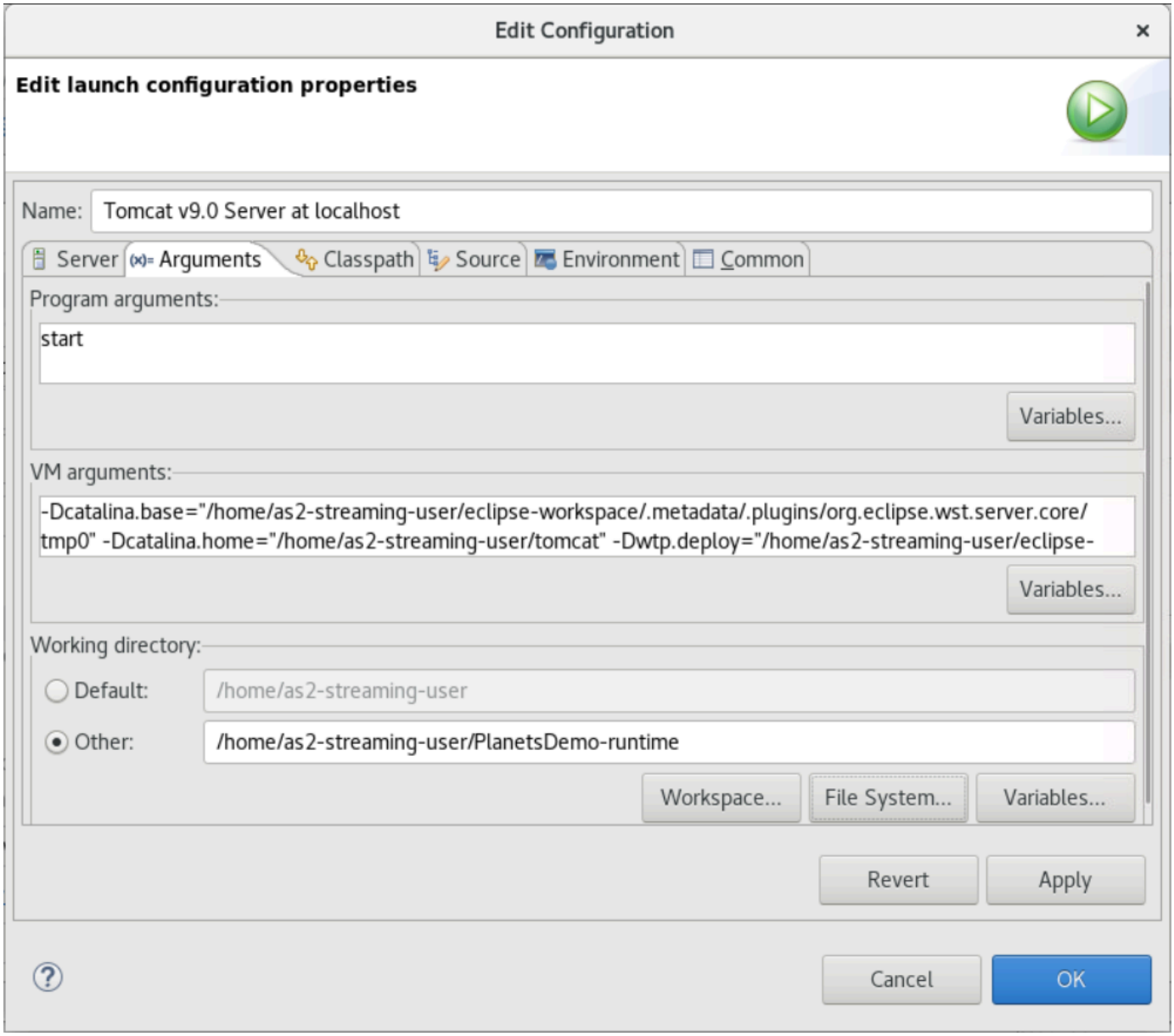
4. Abra a entrada `tomcat v9.0` do servidor na visualização Servidores. O editor de propriedades do servidor aparece:




5. Na guia Visão geral, aumente os valores de tempo limite para 450 segundos para Iniciar e 150 segundos para Parar, conforme mostrado aqui:




6. Escolha Abrir configuração de lançamento. É exibido um assistente. No assistente, navegue até a pasta Argumentos e, em Diretório de trabalho, escolha Outro. Escolha Sistema de arquivos e navegue até a `PlanetsDemo-runtime` pasta descompactada anteriormente. Essa pasta deve conter uma subpasta direta chamada config.

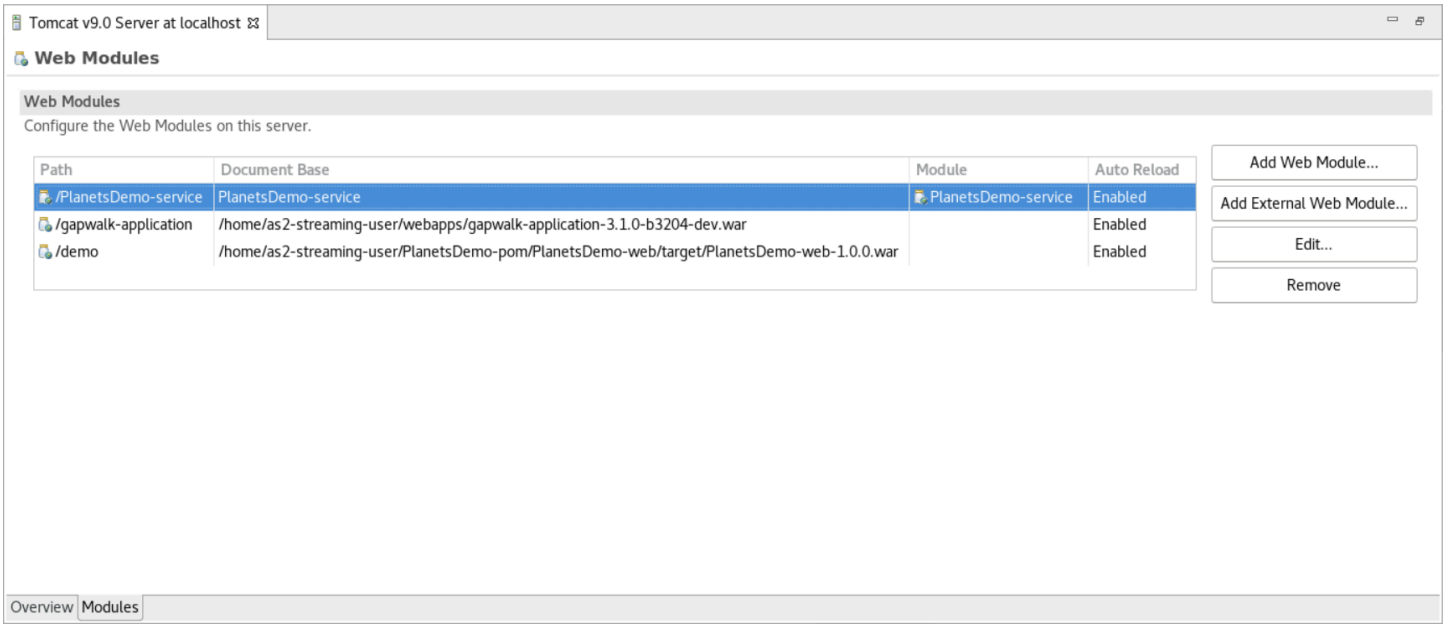



7. Escolha a guia Módulos do editor de propriedades do servidor e faça as seguintes alterações:

   - Escolha Adicionar módulo Web e adicione `PlanetsDemo-service`.
   - Escolha Adicionar módulo Web externo. A janela de diálogo Adicionar módulo Web é exibida. Faça as seguintes alterações em:
     - Na base de documentos, escolha Procurar e navegue até `~/webapps/gapwalk-application...war`
     - Em Caminho, insira`/gapwalk-application`.
   - Escolha OK.
   - Escolha Adicionar módulo Web externo novamente e faça as seguintes alterações:

  - Para Document base, insira o caminho para o frontend .war (in) `PlanetsDemo-web/target`
  - Em Path, insira `/demo`
- Escolha OK
- Salve as modificações do editor (Ctrl + S).

O conteúdo do editor agora deve ser semelhante ao seguinte.



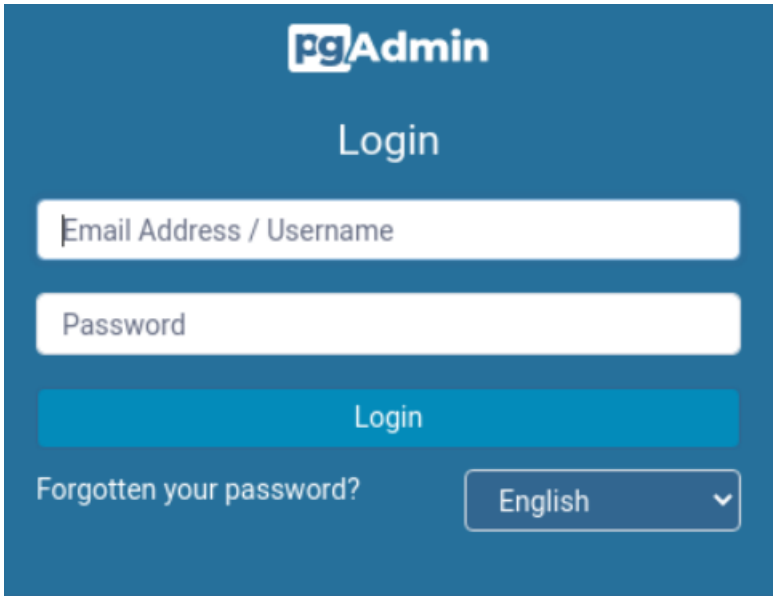

## Etapa 8: criar o banco de dados do JICS

Nesta etapa, você se conecta ao banco de dados do criado em Etapa 1: criar um banco de dados.

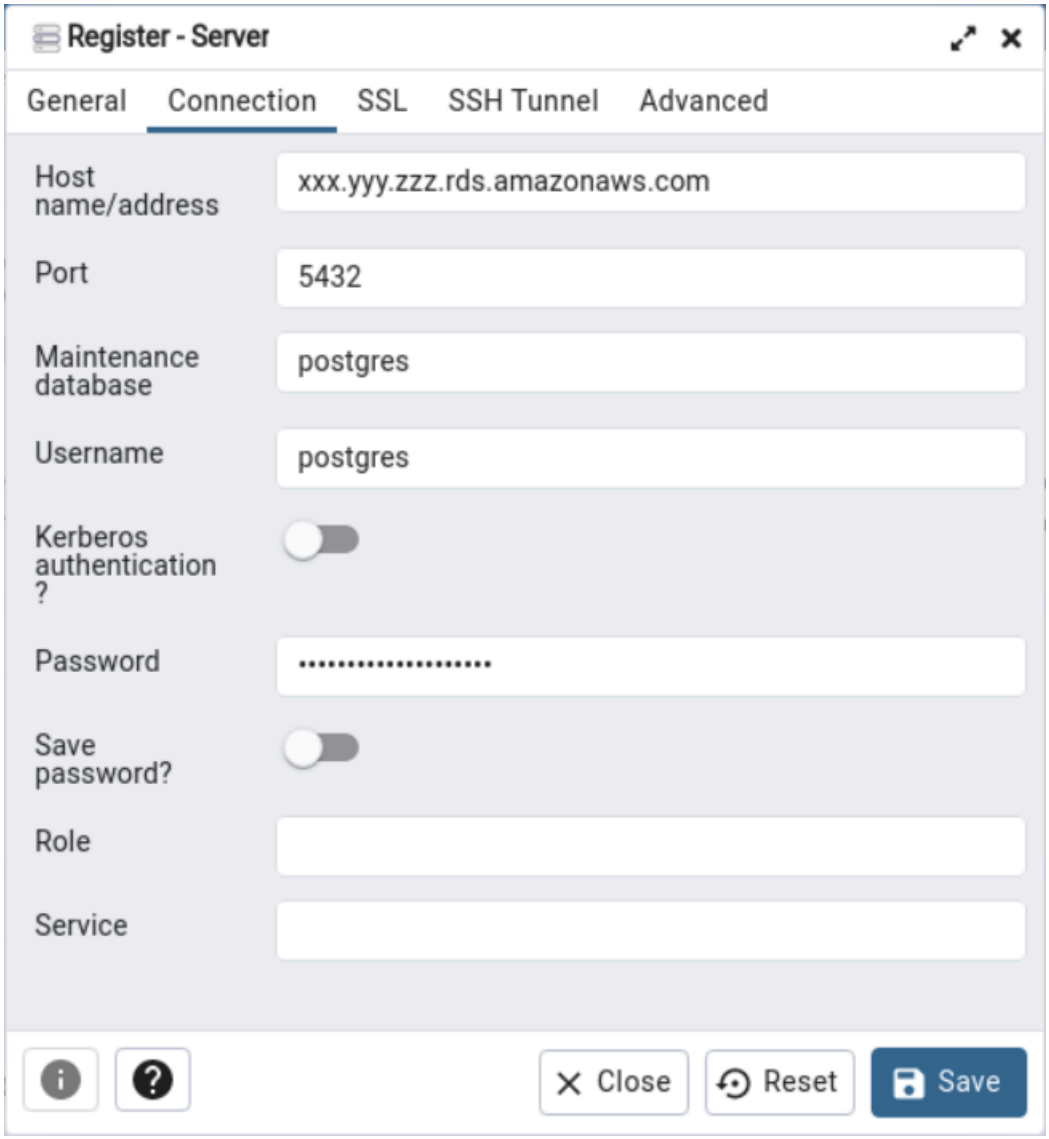
1. Na instância AppStream 2.0, execute o seguinte comando em um terminal para iniciar`pgAdmin`:

   ```
   ./pgadmin-start.sh
   ```

2. Escolha um endereço de e-mail e uma senha como identificadores de login. Anote o URL fornecido (normalmente http://127.0.0.1:5050). Inicie o Google Chrome na instância, copie e cole o URL no navegador e faça login com seus identificadores.




3. Depois de fazer login, escolha Adicionar novo servidor e insira as informações de conexão no banco de dados criado anteriormente da seguinte maneira.

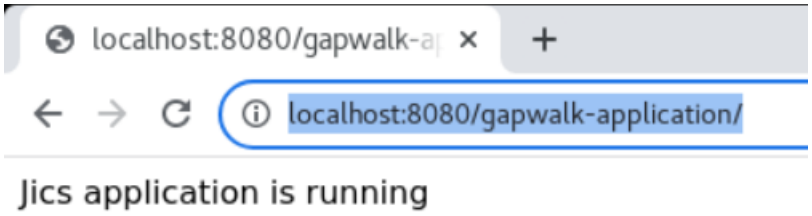



4. Ao se conectar ao servidor do banco de dados, use Objeto > Criar > Banco de dados e crie um novo banco de dados chamado jics.
5. Edite as informações de conexão do banco de dados que o aplicativo de demonstração usou. Essas informações são definidas em `PlanetsDemo-runtime/config/application-main.yml`. Pesquise a `jicsDs` entrada. Para recuperar os valores `username` e `password`, no console do Amazon RDS, navegue até o banco de dados. Na guia Configuration (Configuração), em Master Credentials ARN (ARN das credenciais principais), escolha Manage in Secrets Manager (Gerenciar no Secrets Manager). Em seguida, no console do Secrets Manager, no segredo, escolha Recuperar valor secreto.

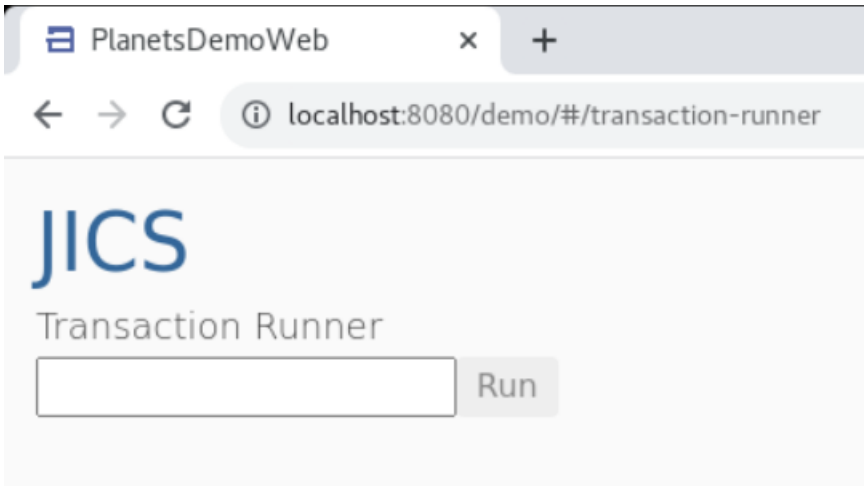
## Etapa 9: Inicie e teste o aplicativo

Nesta etapa, você inicia o servidor Tomcat e o aplicativo de demonstração para poder testá-lo.

1. Para iniciar o servidor Tomcat e os aplicativos implantados anteriormente, selecione a entrada do servidor na visualização Servidores e escolha Iniciar. É exibido um console que exibe os registros de inicialização.
2. Verifique o status do servidor na visualização Servidores ou aguarde a inicialização do servidor em [xxx] milissegundos na mensagem no console. Depois que o servidor for iniciado, verifique se o aplicativo gapwalk-application está implantado corretamente. Para fazer isso, acesse a URL http://localhost:8080/gapwalk-application em um navegador Google Chrome. Você deverá ver o seguinte.



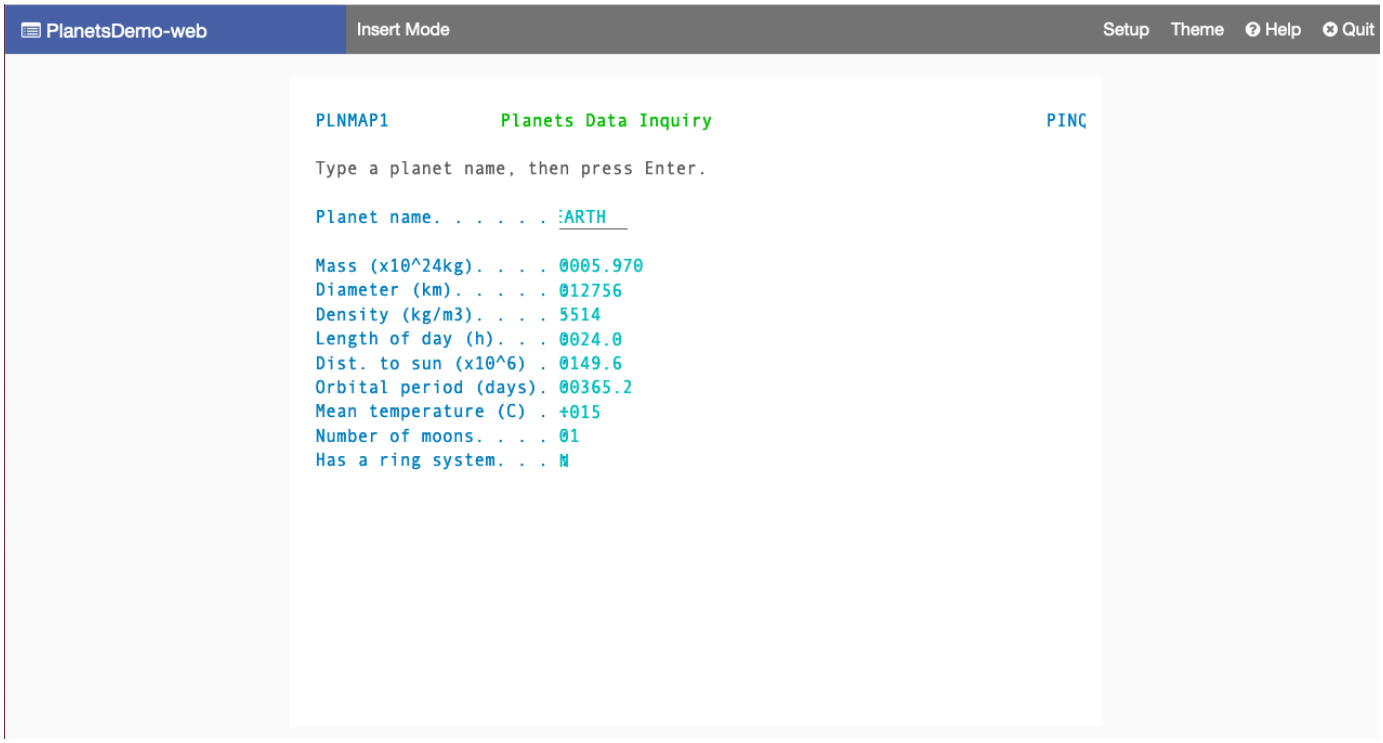

3. Acesse o front-end do aplicativo implantado no Google Chrome em http://localhost:8080/demo. A seguinte página do Transaction Launcher deve aparecer.




4. Para iniciar a transação do aplicativo, insira `PINQ` no campo de entrada e escolha Executar (ou pressione Enter).

   A tela do aplicativo de demonstração deve aparecer.

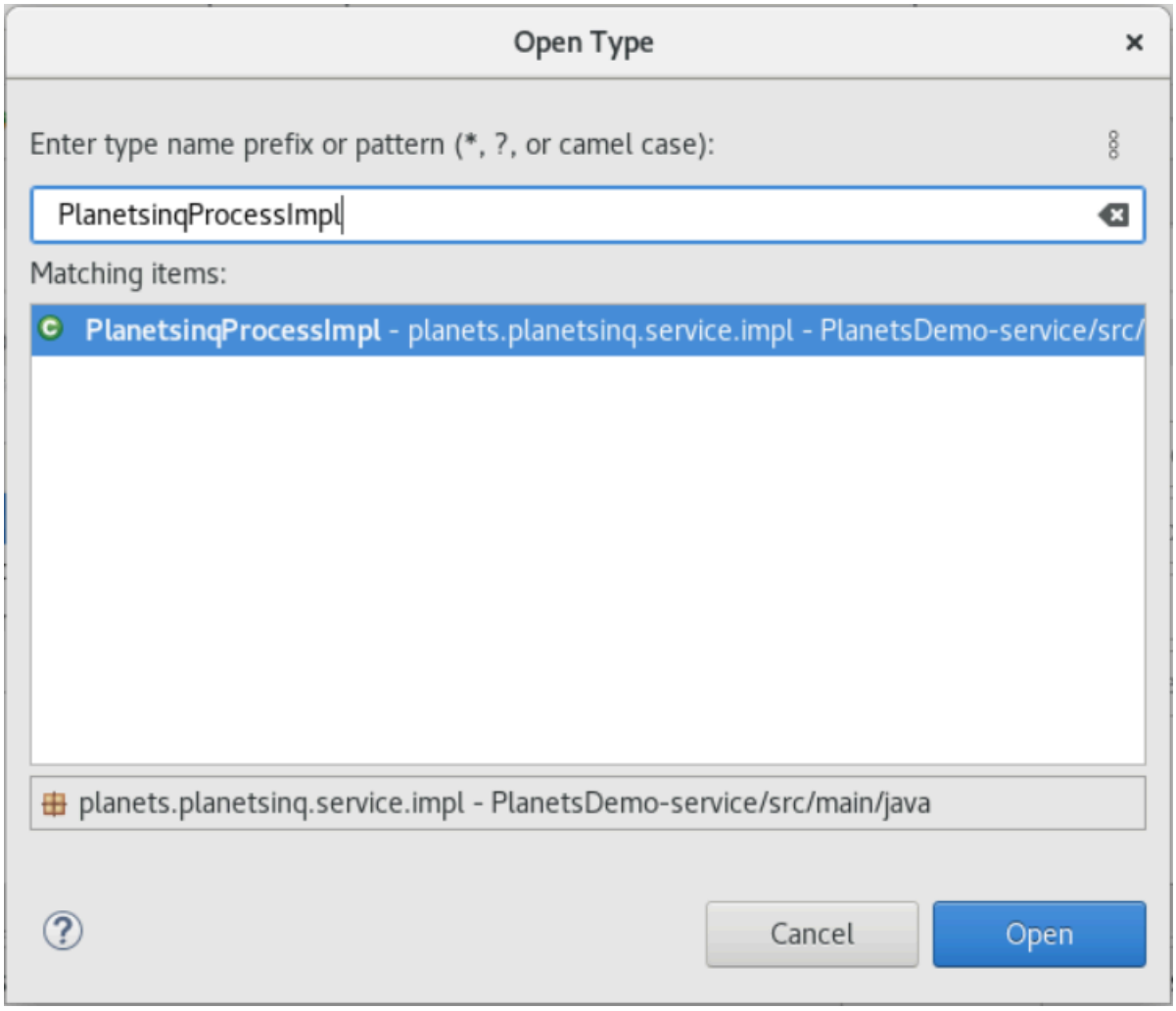

5. Digite o nome do planeta no campo correspondente e pressione Enter.



## Etapa 10: Depurar o aplicativo

Nesta etapa, você testará usando os recursos de depuração padrão do Eclipse. Esses recursos estão disponíveis quando você trabalha em um aplicativo modernizado.

1. Para abrir a classe de serviço principal, pressione Ctrl + Shift + T. Em seguida, insira. `PlanetsinqProcessImpl`




2. Navegue até o `searchPlanet` método e coloque um ponto de interrupção nele.
3. Selecione o nome do servidor e selecione Reiniciar na depuração.
4. Repita as etapas anteriores. Ou seja, acesse o aplicativo, insira o nome do planeta e pressione Enter.

   O Eclipse interromperá o aplicativo no `searchPlanet` método. Agora você pode examiná-lo.

## Limpeza de recursos

Se os recursos criados durante este tutorial não forem mais necessários, exclua-os para não incorrer em cobranças adicionais. Execute as etapas a seguir:

- Se o aplicativo Planets ainda estiver em execução, pare-o.
- Exclua o banco de dados que você criou em Etapa 1: criar um banco de dados. Para ter mais informações, consulte Excluir uma instância de banco de dados.

# Reestruturando aplicativos com a Micro Focus

Esta seção descreve cada etapa do processo de reformulação de plataforma. Ele descreve todas as tarefas e inclui informações sobre como configurar e operar o tempo de execução da modernização de AWS mainframe no Amazon EC2.

Tópicos



# Configuração do Micro Focus Runtime (no Amazon EC2)

AWS A modernização do mainframe fornece várias imagens de máquina da Amazon (AMIs) que incluem produtos licenciados pela Micro Focus. Essas AMIs permitem que você provisione rapidamente instâncias do Amazon Elastic Compute Cloud (Amazon EC2) para oferecer suporte a ambientes da Micro Focus que você controla e gerencia. Este tópico fornece as etapas necessárias para acessar e iniciar essas AMIs. O uso dessas AMIs é totalmente opcional e elas não são obrigatórias para concluir os tutoriais deste guia do usuário.

Tópicos

- Criando a AWS Identity and Access Management função
- Conceda ao License Manager as permissões necessárias
- Inscreva-se para receber as imagens de máquina da Amazon
- Inicie uma instância Micro Focus de modernização de AWS mainframe
- Sub-rede ou VPC sem acesso à Internet
- Solução de problemas de licença

## Pré-requisitos

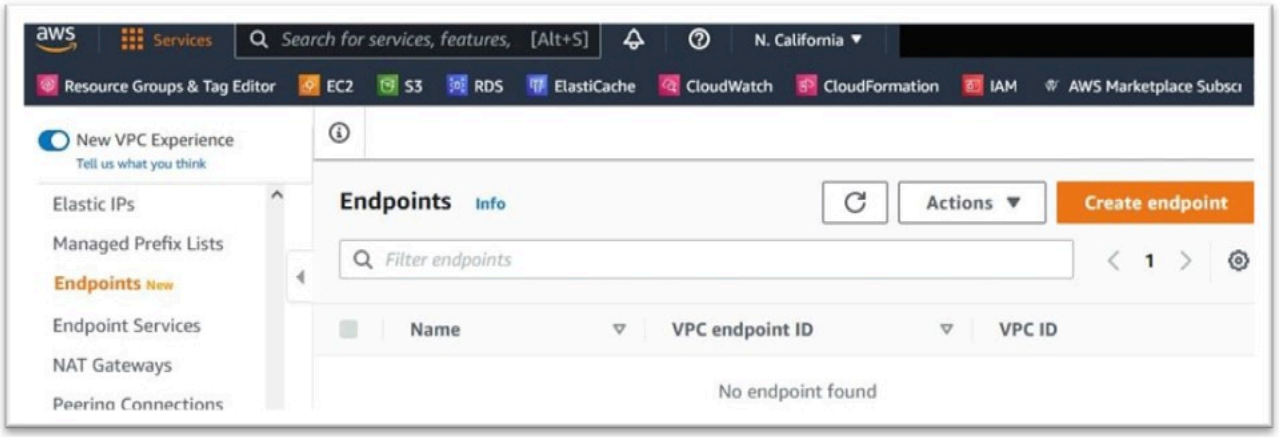
Certifique-se de que você atende aos seguintes pré-requisitos.

- Acesso do administrador à conta na qual as instâncias do Amazon EC2 serão criadas.
- Identifique Região da AWS onde as instâncias do Amazon EC2 serão criadas e verifique se o serviço de modernização do AWS mainframe está disponível. Consulte Serviços por região AWS. Selecione uma região na qual o serviço esteja disponível.
- Identifique a Amazon Virtual Private Cloud (Amazon VPC) na qual as instâncias do Amazon EC2 serão criadas.

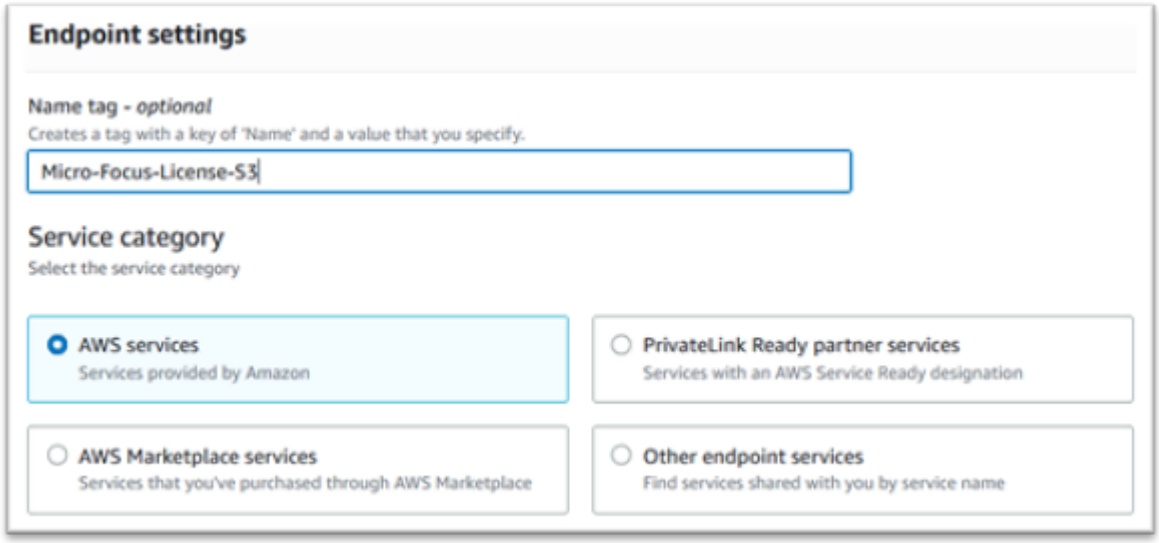
## Criar o ponto de extremidade do Amazon VPC para o Amazon S3

Nesta seção, você cria um endpoint da Amazon VPC para o Amazon S3 usar.

1. Navegue até o Amazon VPC na seção AWS Management Console
2. No painel de navegação, escolha Endpoints.
3. Escolha Criar endpoint.



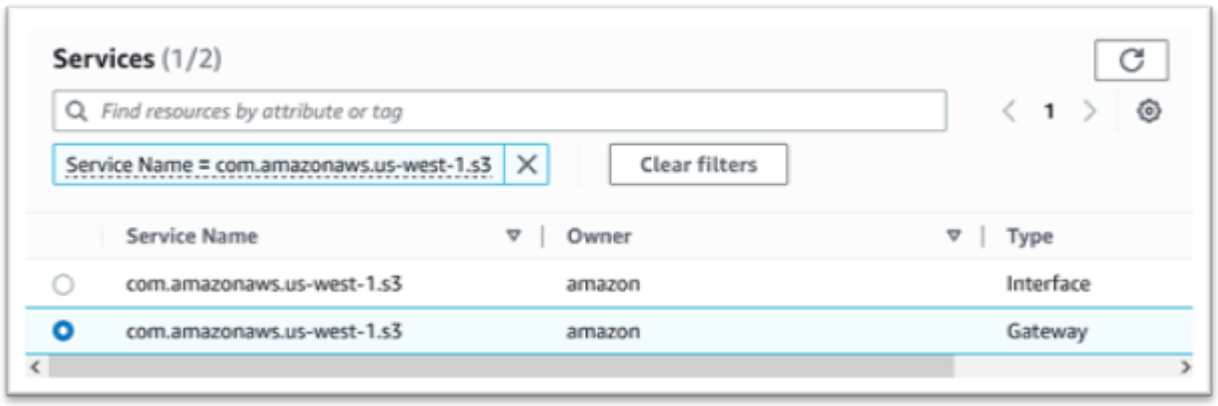
4. Insira uma etiqueta de nome significativa, por exemplo: “Micro-Focus-License-S3”.
5. Escolha AWS Services como a categoria de serviço.



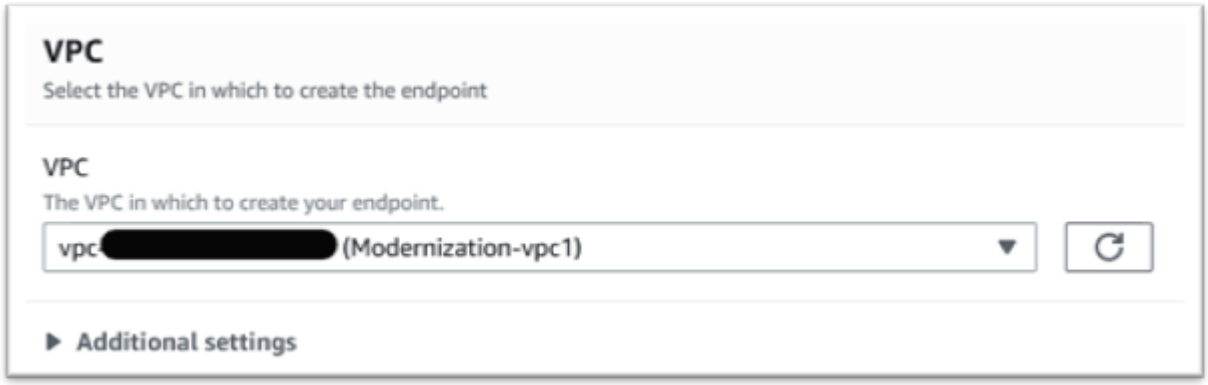

6. Em Serviços, pesquise o serviço Amazon S3 Gateway: com.amazonaws. [região] .s3.

   Para `us-west-1` isso seria:`com.amazonaws.us-west-1.s3`.

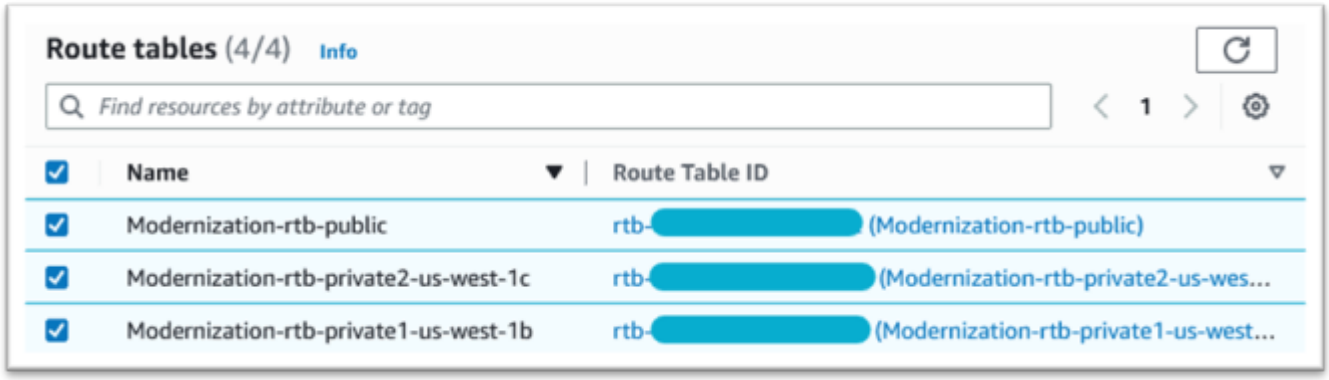
7. Escolha o serviço Gateway.



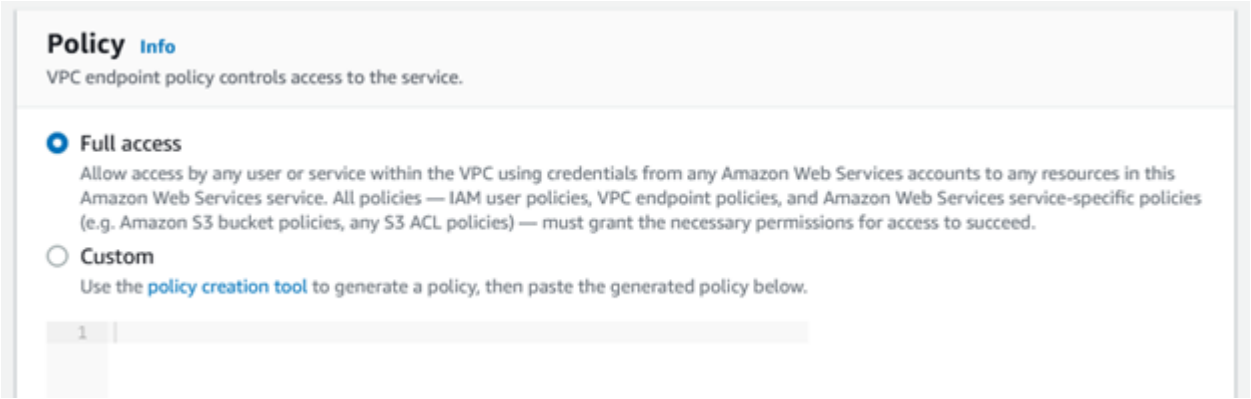

8. Para VPC, escolha a VPC que você usará.




9. Escolha todas as tabelas de rotas para a VPC.




10. Em Política, escolha Acesso total.




11. Escolha Criar endpoint.

## Solicite a atualização da lista de permissões para a conta

Trabalhe com seu AWS representante para que sua conta seja incluída na lista de permissões para as AMIs de modernização de AWS mainframe. Forneça as seguintes informações:

- O Conta da AWS ID.

- O Região da AWS local onde o endpoint da Amazon VPC foi criado.
- O ID do ponto de extremidade do Amazon VPC Amazon S3 criado em Criar o ponto de extremidade do Amazon VPC para o Amazon S3. Esse é o `vpce-xxxxxxxxxxxxxxxxx` id do com.amazonaws. Endpoint do gateway [region] .s3.
- O número de licenças necessárias em todas as instâncias do Micro Focus Enterprise Suite AMI Amazon EC2.

  É necessária uma licença por núcleo de CPU (por 2 vCPUs para a maioria das instâncias do Amazon EC2).

  Para obter mais informações, consulte Otimizar as opções da CPU.

  O número solicitado pode ser ajustado futuramente por AWS.

> **Note**
>
> O AWS representante deve abrir o ticket de suporte para a solicitação da Lista de Permissões. Ela não pode ser solicitada diretamente e a solicitação pode levar vários dias para ser concluída.

## Criando a AWS Identity and Access Management função

Crie uma AWS Identity and Access Management política e uma função para serem usadas pelas instâncias de modernização de AWS mainframe do Amazon EC2. Criar a função por meio do console do IAM criará um perfil da instância associado com o mesmo nome. A atribuição desse perfil da instância às instâncias do Amazon EC2 permite que as licenças da Micro Focus sejam atribuídas. Para obter mais informações sobre perfis de instância, consulte Uso de uma função de IAM para conceder permissões a aplicativos executados em instâncias do Amazon EC2.

### Criar uma política do IAM

Uma política de IAM é criada primeiro e depois anexada à função.

1. Navegue até AWS Identity and Access Management no AWS Management Console.
2. Escolha Políticas e depois Criar política.

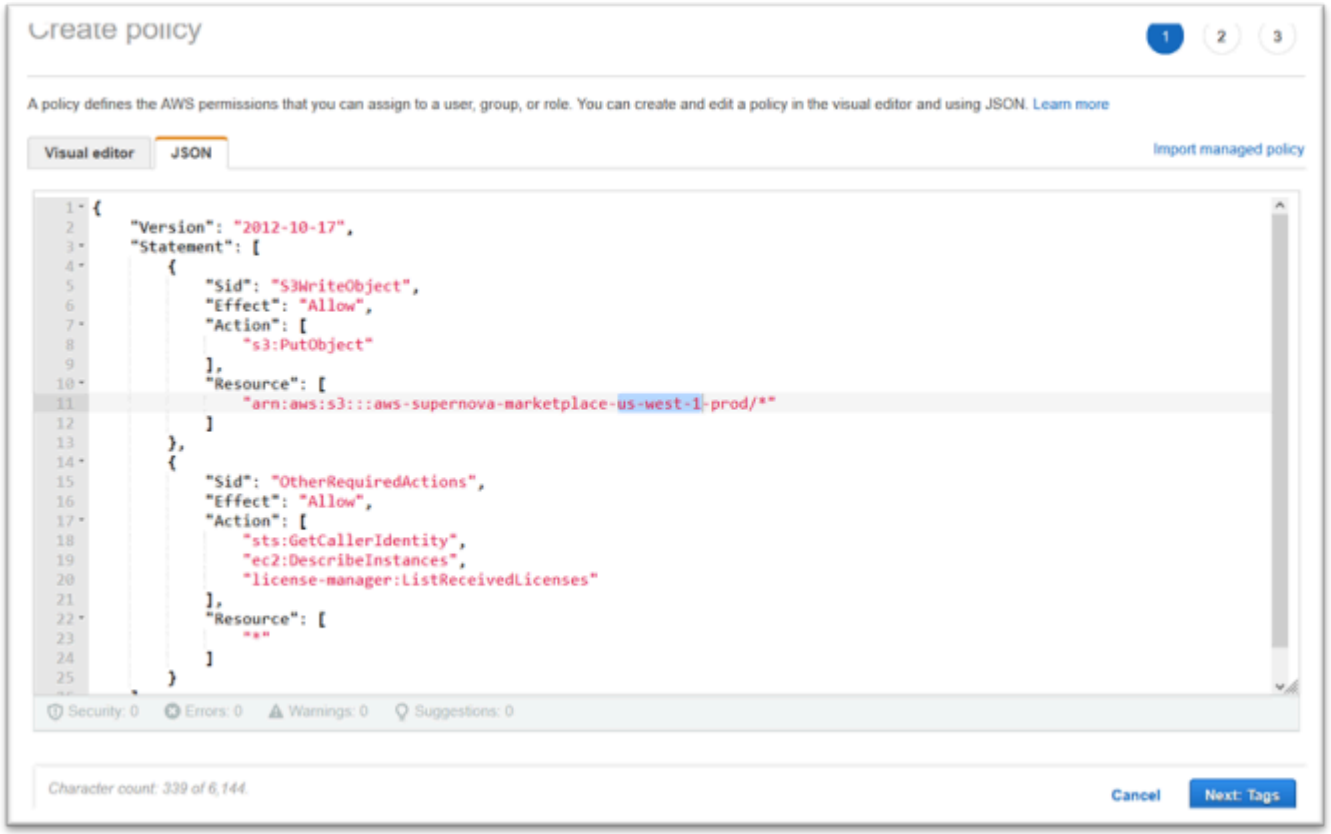

3. Selecione a guia JSON.


4. Substitua o seguinte JSON pelo local `us-west-1` em Região da AWS que o endpoint do Amazon S3 foi definido e, em seguida, copie e cole o JSON no editor de políticas.

```
{
    "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",
```

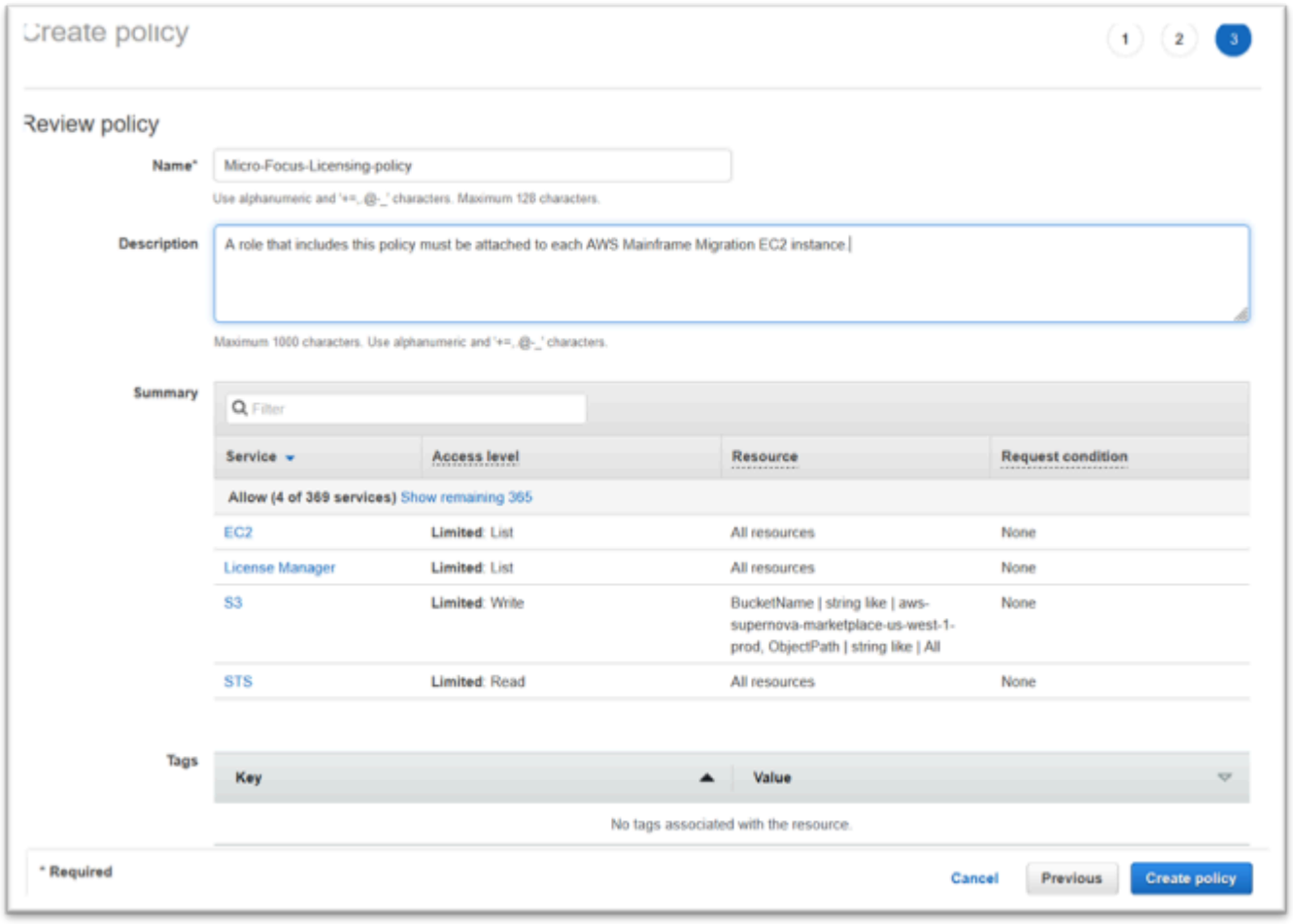
```
            "Effect": "Allow",
            "Action": [
                "sts:GetCallerIdentity",
                "ec2:DescribeInstances",
                "license-manager:ListReceivedLicenses"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

> **Note**
>
> As ações do Sid `OtherRequiredActions` não oferecem suporte a permissões de nível de recurso e precisam ser especificadas `*` no elemento de recurso.




5. Escolha Próximo: etiquetas.

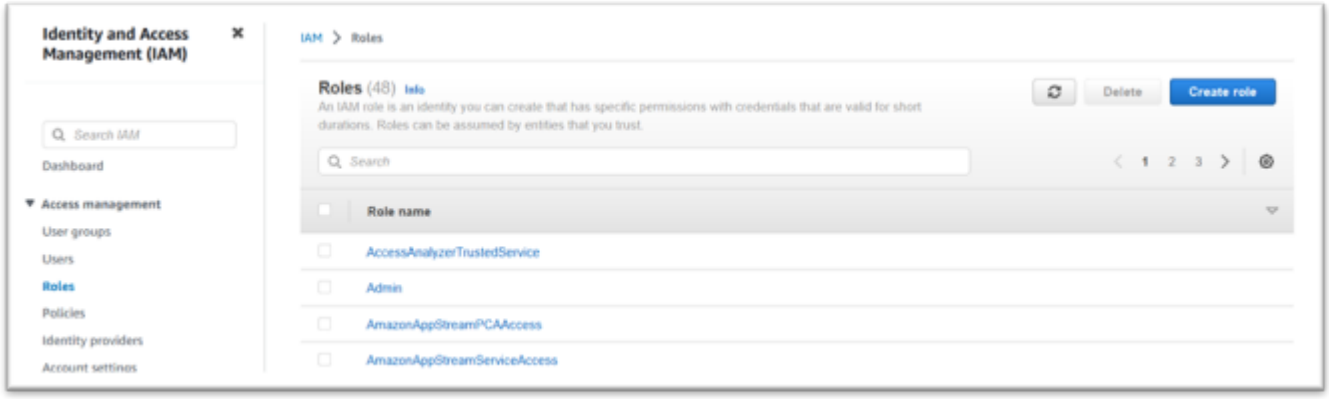

6. Opcionalmente, insira qualquer tag e escolha Próximo: Revisar.
7. Insira um nome para a política, por exemplo, “Política de licenciamento da Micro-Focus”. Opcionalmente, insira uma descrição, por exemplo, “Uma função que inclua essa política deve ser anexada a cada instância de modernização de AWS mainframe do Amazon EC2”.



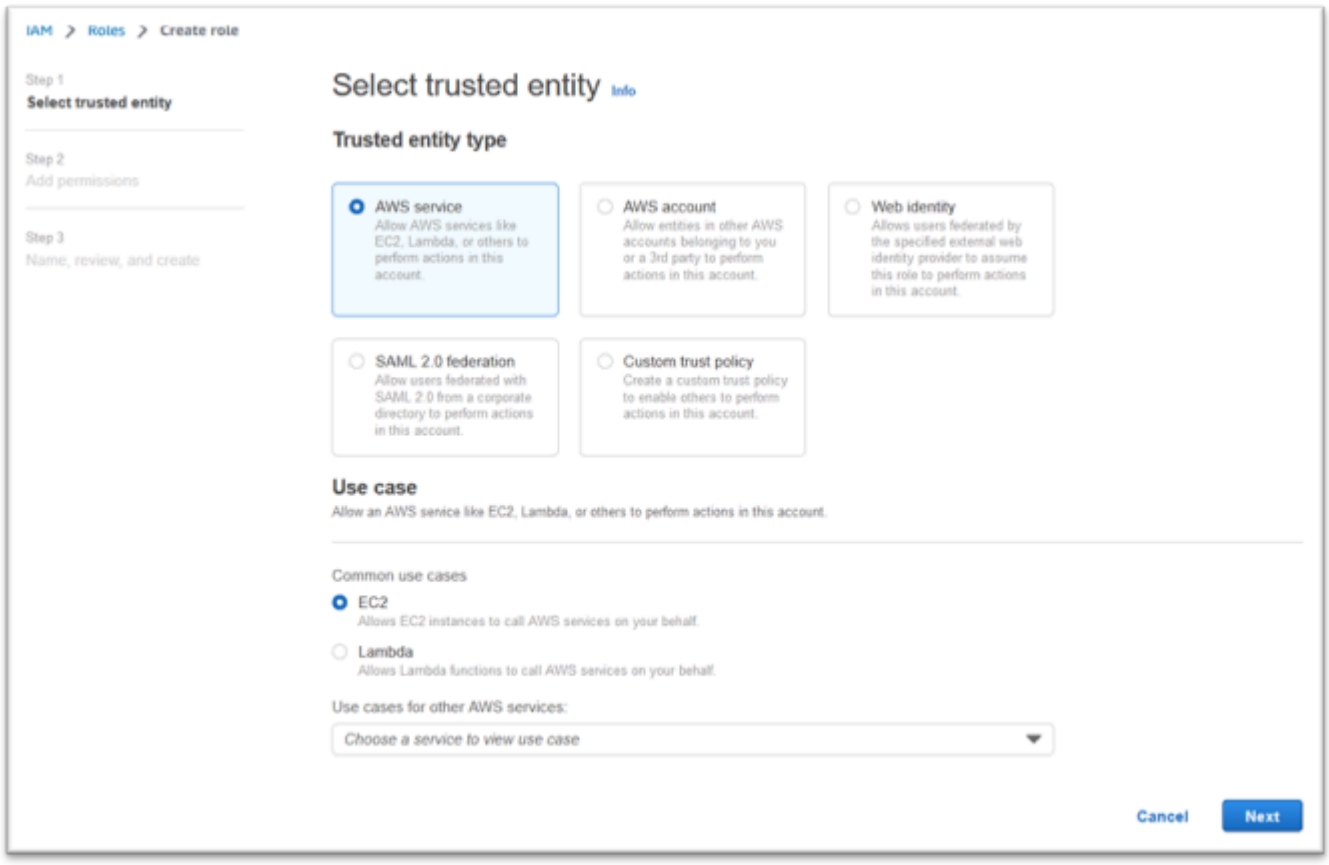

8. Escolha Create Policy.

## Criar a função de IAM

1. Navegue até IAM no AWS Management Console.
2. Escolha Perfis e, depois, Criar perfil.

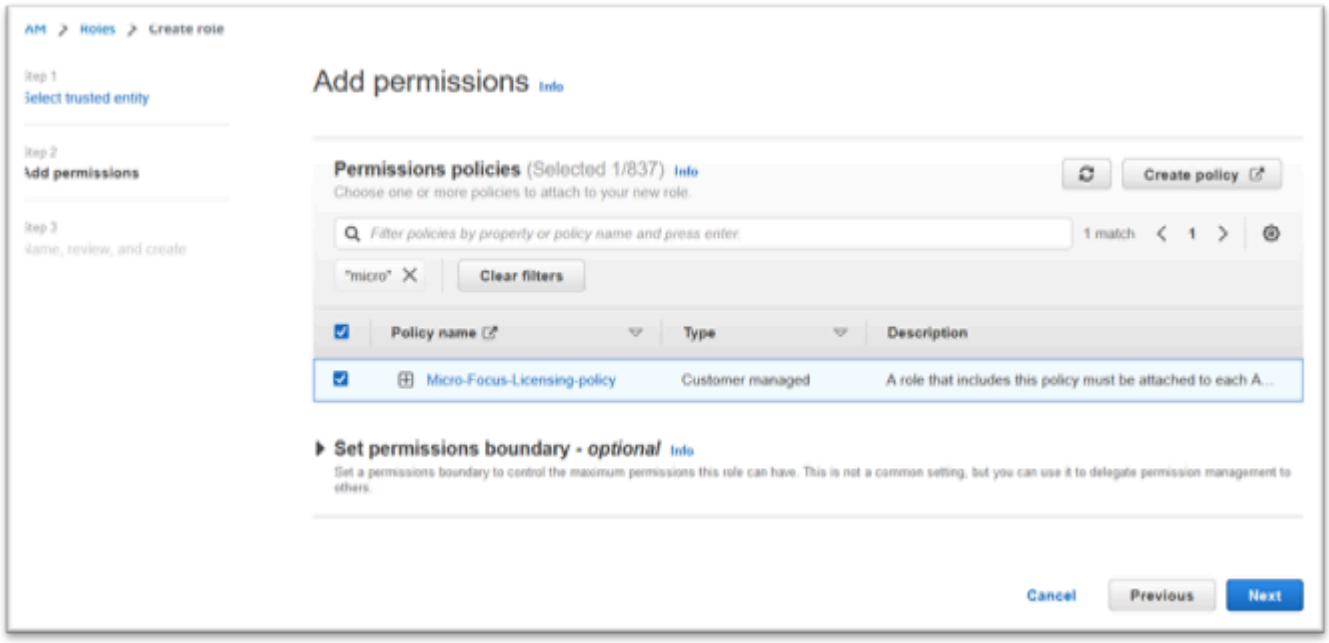



3. Deixe o tipo de entidade confiável como serviço AWS  e escolha o caso de uso comum do EC2.

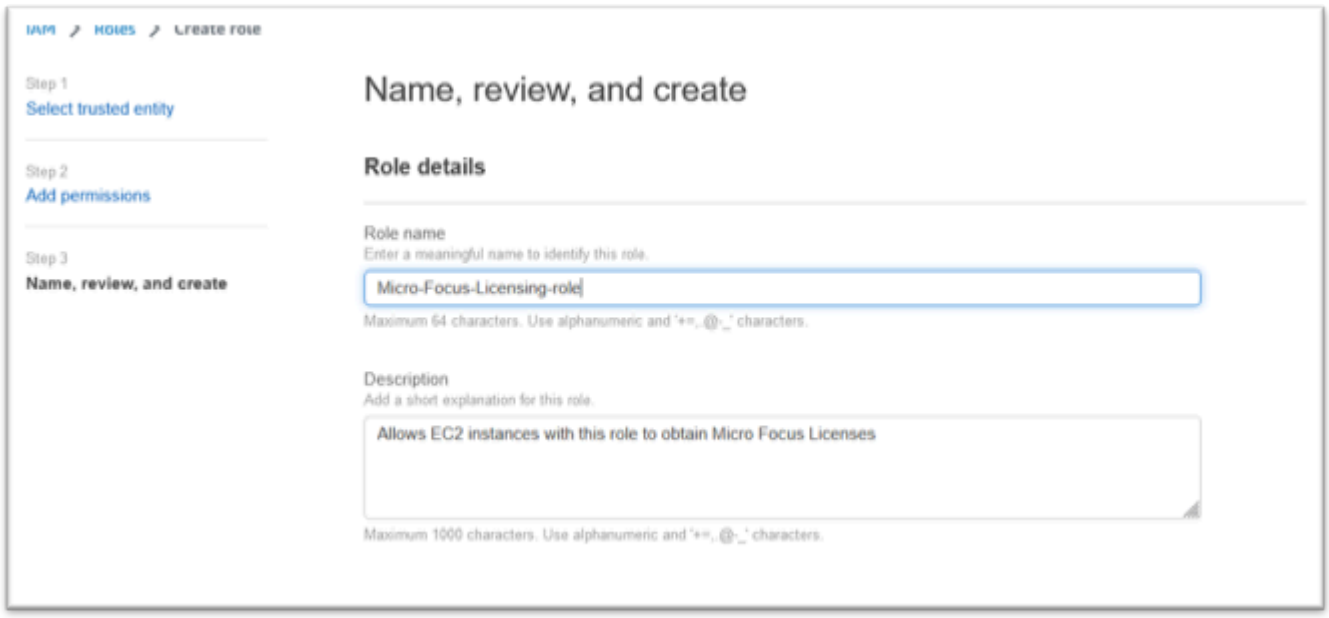



4. Escolha Próximo.
5. Digite “Micro” no filtro e pressione enter para aplicar o filtro.

6. Escolha a política que acabou de ser criada, por exemplo, a “Política de licenciamento da Micro-Focus”.
7. Escolha Próximo.




8. Insira o nome da função, por exemplo, “Micro-Focus-Licensing-Role”.
9. Substitua a descrição por uma de sua preferência, por exemplo, “Permite que instâncias do Amazon EC2 com essa função obtenham licenças da Micro Focus”.

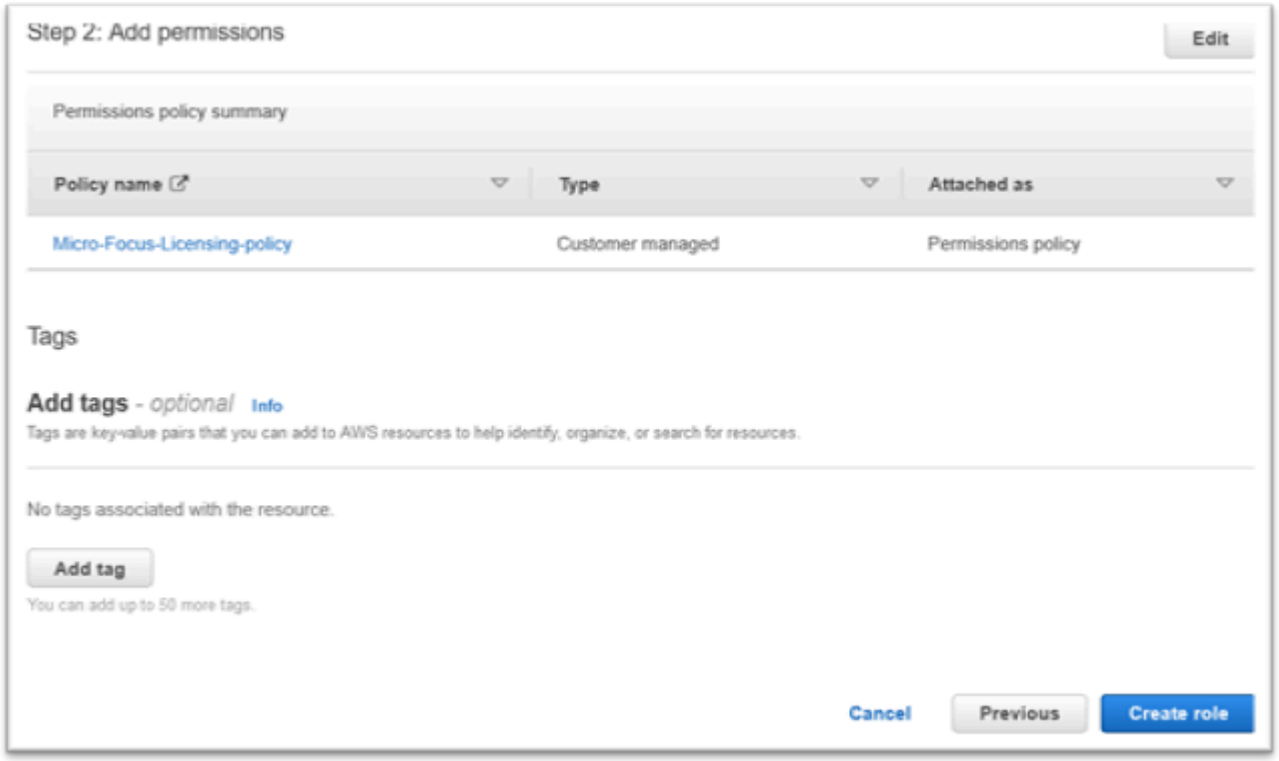



10. Em Etapa 1: Selecione entidades confiáveis, revise o JSON e confirme se ele tem os seguintes valores:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Principal": {
                "Service": [
                    "ec2.amazonaws.com"
                ]
            }
        }
    ]
}
```

> **Note**
>
> A ordem do Efeito, da Ação e do Principal não é significativa.

11. Confirme se a Etapa 2: Adicionar permissões mostra sua política de licenciamento.

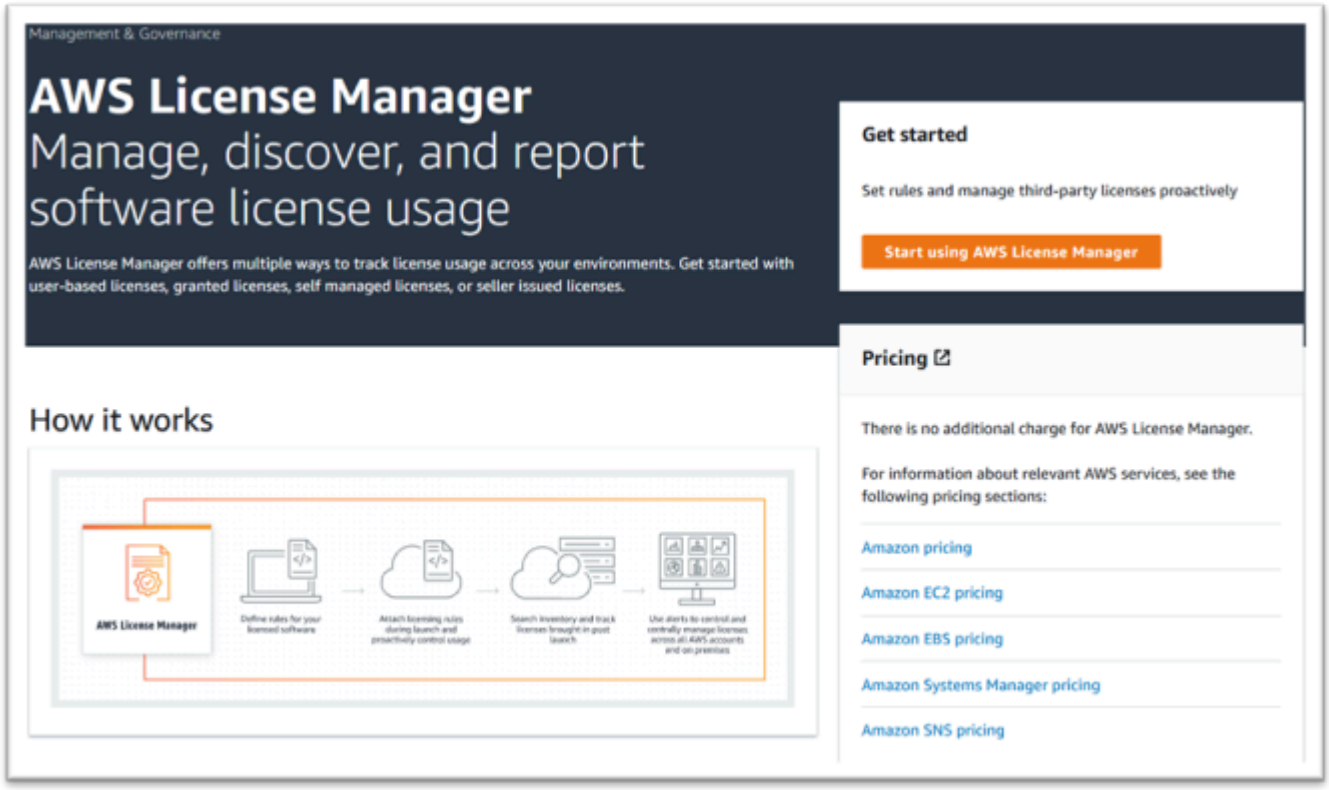



12. Selecione Criar função.

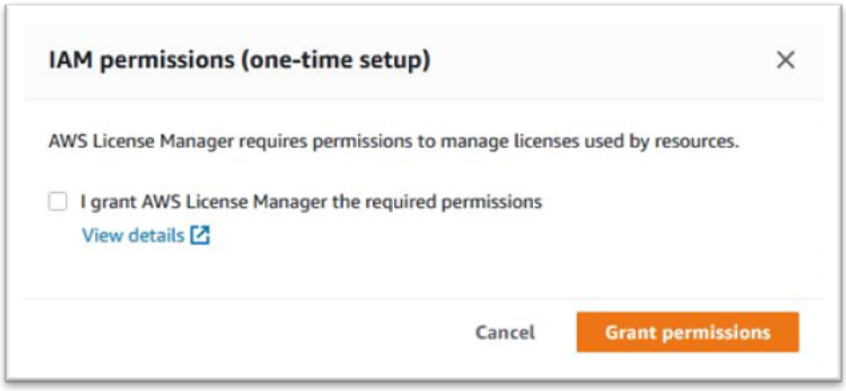
Depois que a solicitação da lista de permissões for concluída, continue com as etapas a seguir.

## Conceda ao License Manager as permissões necessárias

1. Navegue até AWS License Manager no AWS Management Console.




2. Escolha Começar a usar o AWS License Manager.
3. Se você ver o pop-up a seguir, veja os detalhes, escolha a caixa de seleção e pressione Conceder Permissões.

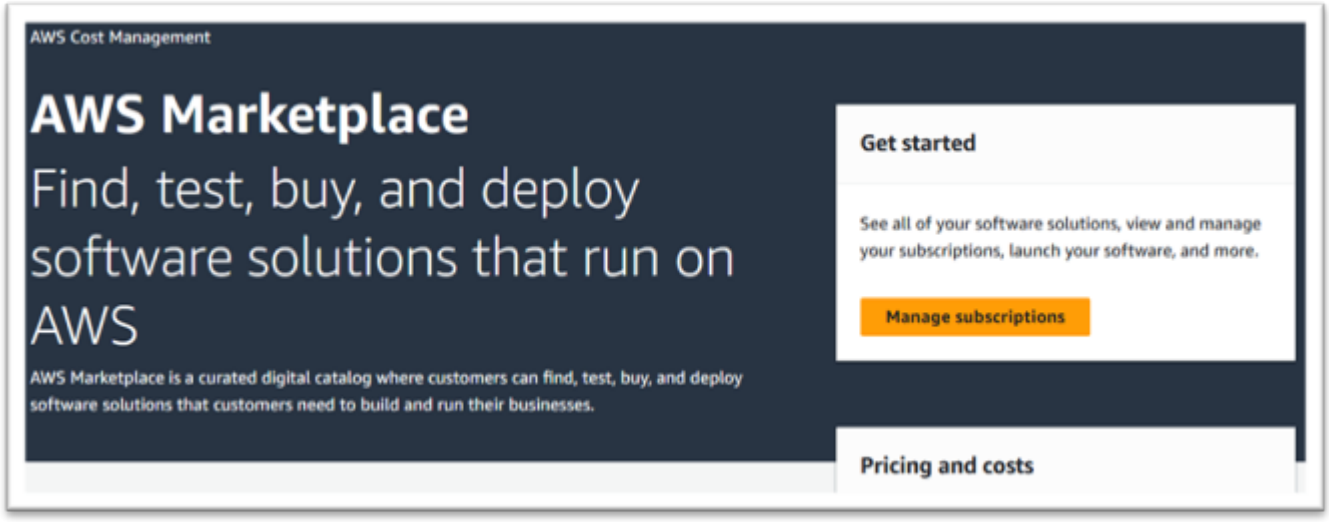



## Inscreva-se para receber as imagens de máquina da Amazon

Depois de assinar um AWS Marketplace produto, você pode iniciar uma instância a partir da AMI do produto.

1. Navegue até AWS Marketplace Assinaturas no. AWS Management Console

2. Escolha Manage subscriptions (Gerenciar assinaturas).



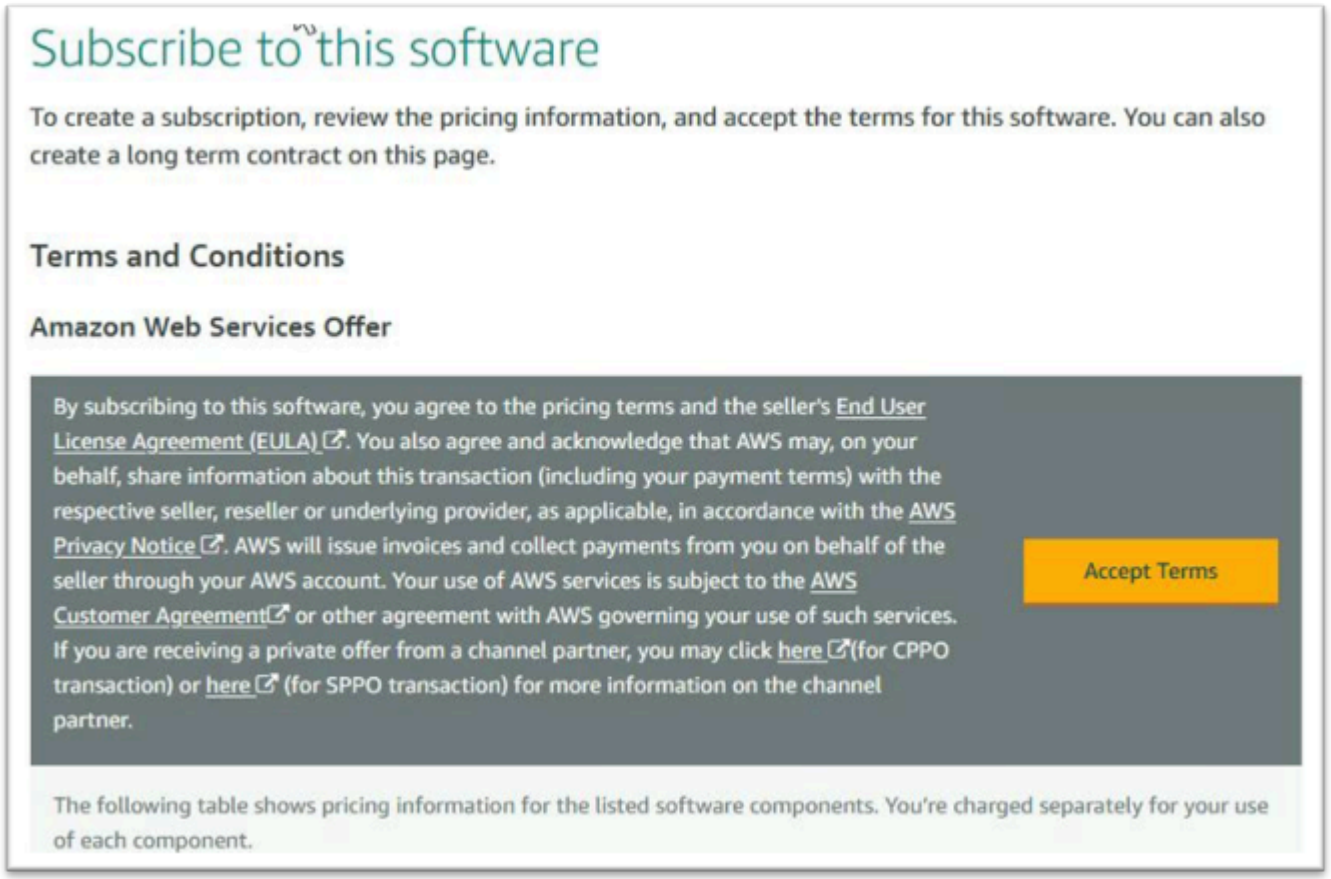

3. Copie e cole um dos links a seguir na barra de endereço do navegador.

> **Note**
>
> Escolha apenas um link para um dos produtos que você foi autorizado a usar.

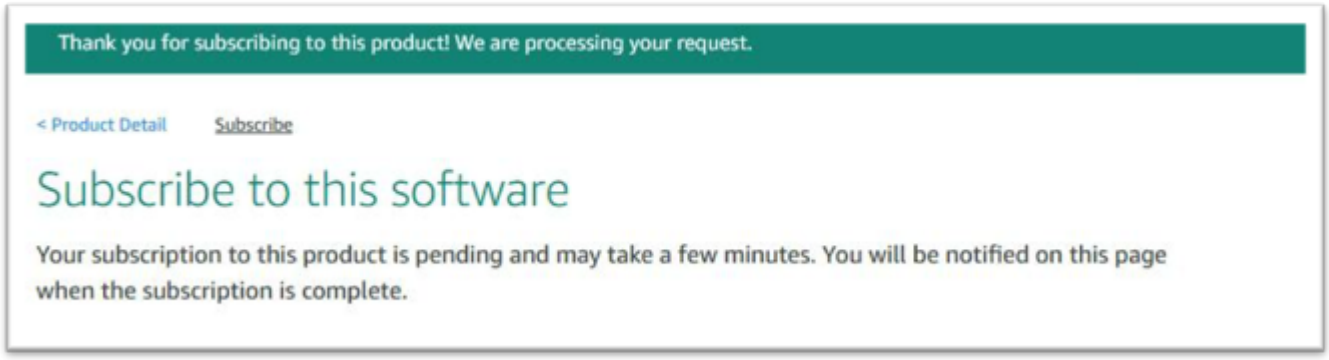
- Servidor corporativo: https://aws.amazon.com/marketplace/pp/prodview-g5emev63l7blc
- Servidor corporativo para Windows: https://aws.amazon.com/marketplace/pp/prodview-lwybsiyikbhc2
- Desenvolvedor corporativo: https://aws.amazon.com/marketplace/pp/prodview-77qmpr42yzxwk
- Desenvolvedor corporativo com Visual Studio 2022: https://aws.amazon.com/marketplace/pp/prodview-m4l3lqiszo6cm
- Analisador corporativo: https://aws.amazon.com/marketplace/pp/prodview-tttheylcmcihm
- Ferramentas de compilação empresarial para Windows: https://aws.amazon.com/marketplace/pp/prodview-2rw35bbt6uozi
- Procedimentos armazenados corporativos: https://aws.amazon.com/marketplace/pp/prodview-zoeyqnsdsj6ha
- Procedimentos armazenados corporativos com o SQL Server 2019: https://aws.amazon.com/marketplace/pp/prodview-ynfklquwubnz4

4. Escolha Continue to Subscribe (Continuar para assinar).

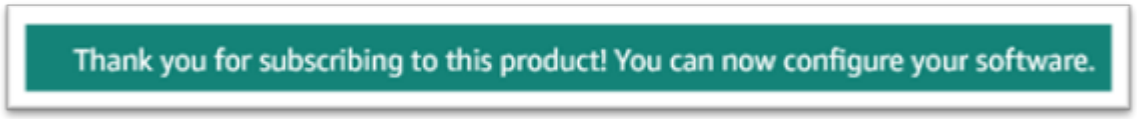

5. Se os Termos e Condições forem aceitáveis, escolha Aceitar Termos.




6. A assinatura pode levar alguns minutos para ser processada.

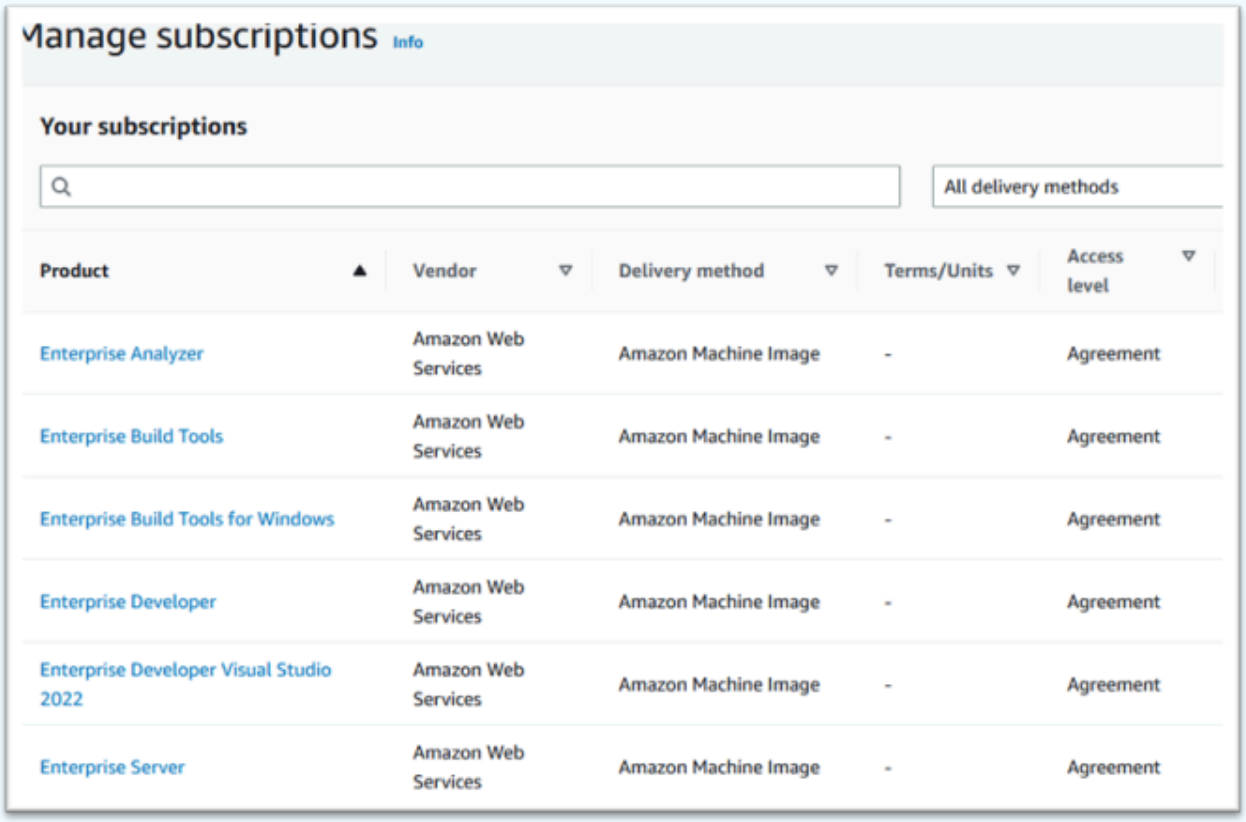



7. Depois que a mensagem de agradecimento for exibida, copie e cole o próximo link da etapa 3 para continuar adicionando assinaturas.



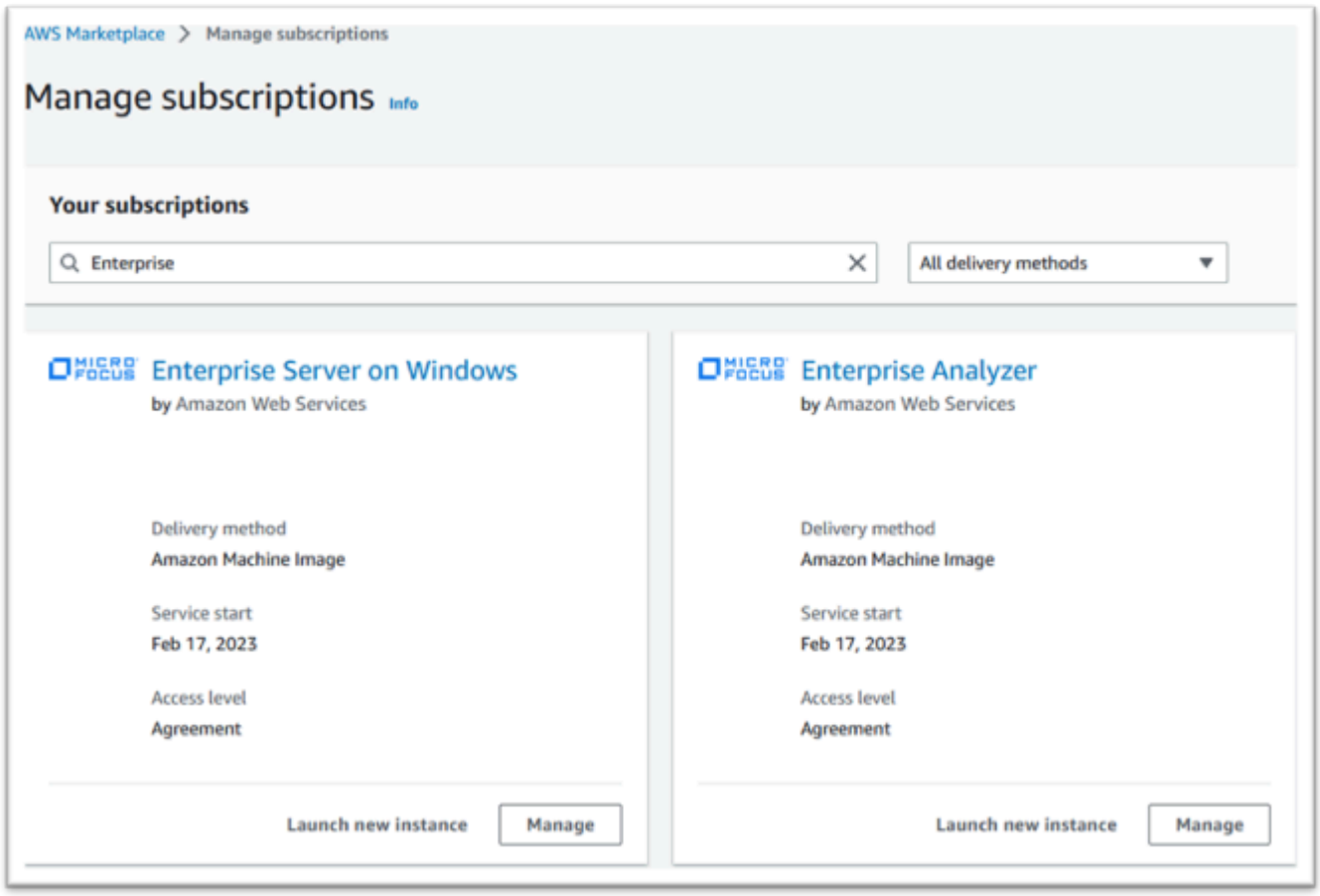

8. Pare quando Gerenciar assinaturas mostrar todas as suas AMIs inscritas.

> **Note**
>
> As preferências do painel (ícone de engrenagem) estão configuradas para mostrar a exibição como uma tabela.



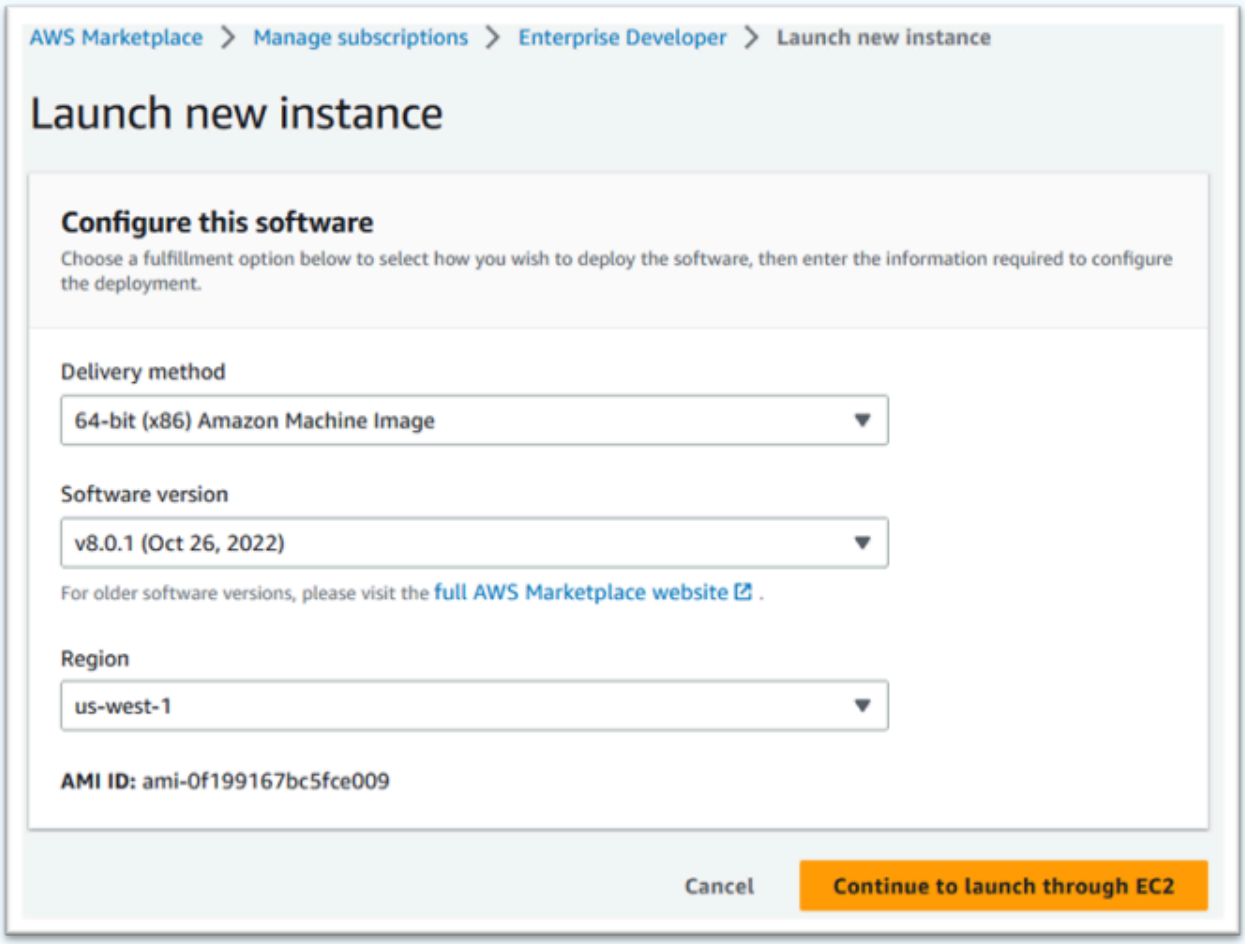

## Inicie uma instância Micro Focus de modernização de AWS mainframe

1. Navegue até AWS Marketplace Assinaturas no. AWS Management Console
2. Localize a AMI a ser executada e escolha Iniciar nova instância.




3. Na caixa de diálogo Iniciar nova instância, verifique se a região da lista de permissões está selecionada.
4. Pressione Continuar para iniciar por meio do EC2.

> **Note**
>
> O exemplo a seguir mostra o lançamento de uma AMI para desenvolvedores corporativos, mas o processo é o mesmo para todas as AMIs de modernização de AWS mainframe.

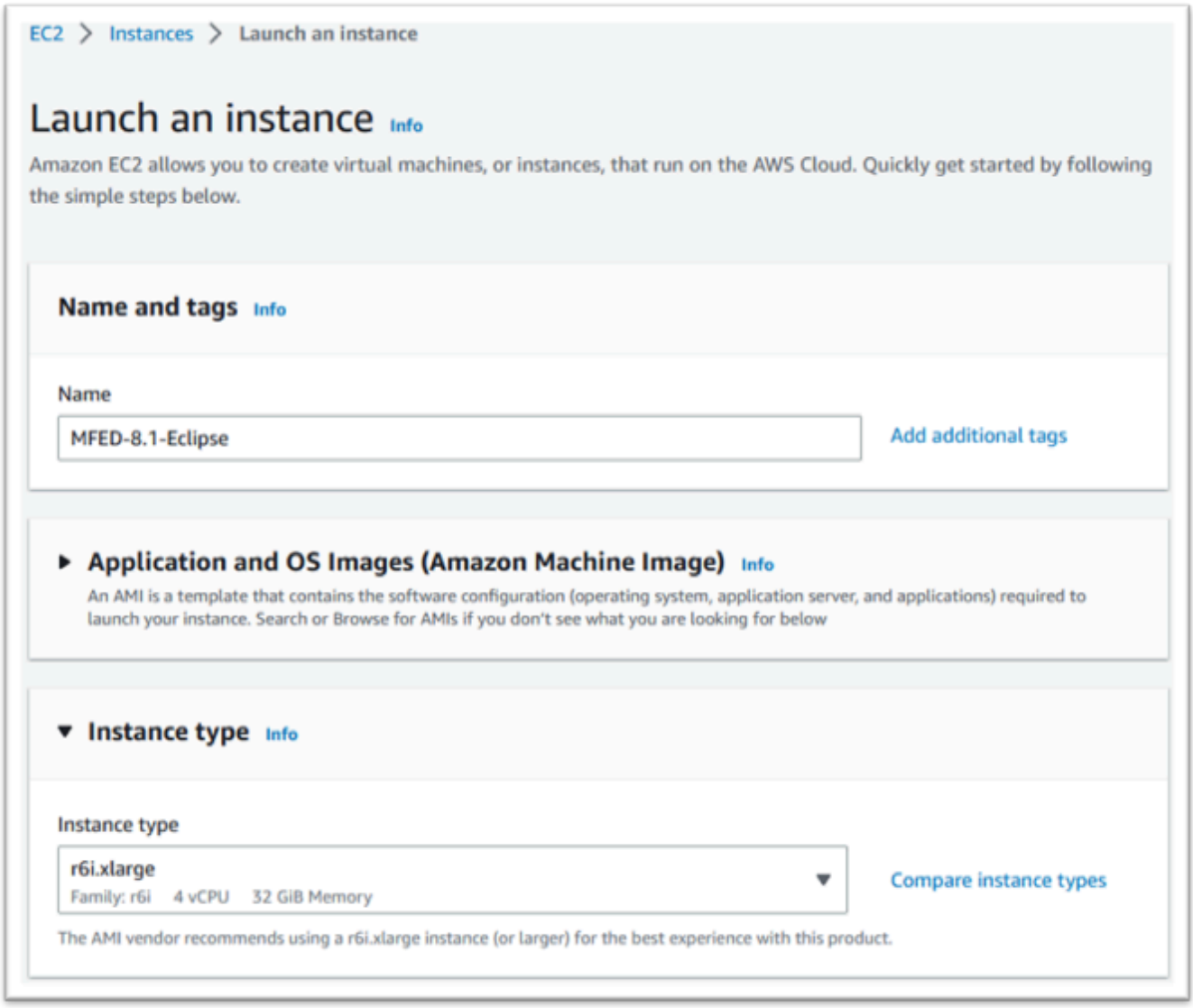



5. Digite um nome para o servidor.
6. Escolha um tipo de instância.

   O tipo de instância selecionado deve ser determinado pelo desempenho do projeto e pelos requisitos de custo. A seguir estão os pontos de partida sugeridos:

   - Para o Enterprise Analyzer, um r6i.xlarge
   - Para desenvolvedores corporativos, um r6i.large
   - Para uma instância autônoma do Enterprise Server, um r6i.xlarge
   - Para o Micro Focus Performance Availability Cluster (PAC) com escalabilidade horizontal, um r6i.large

> **Note**
>
> A seção Imagens do aplicativo e do sistema operacional foi reduzida para a captura de tela.



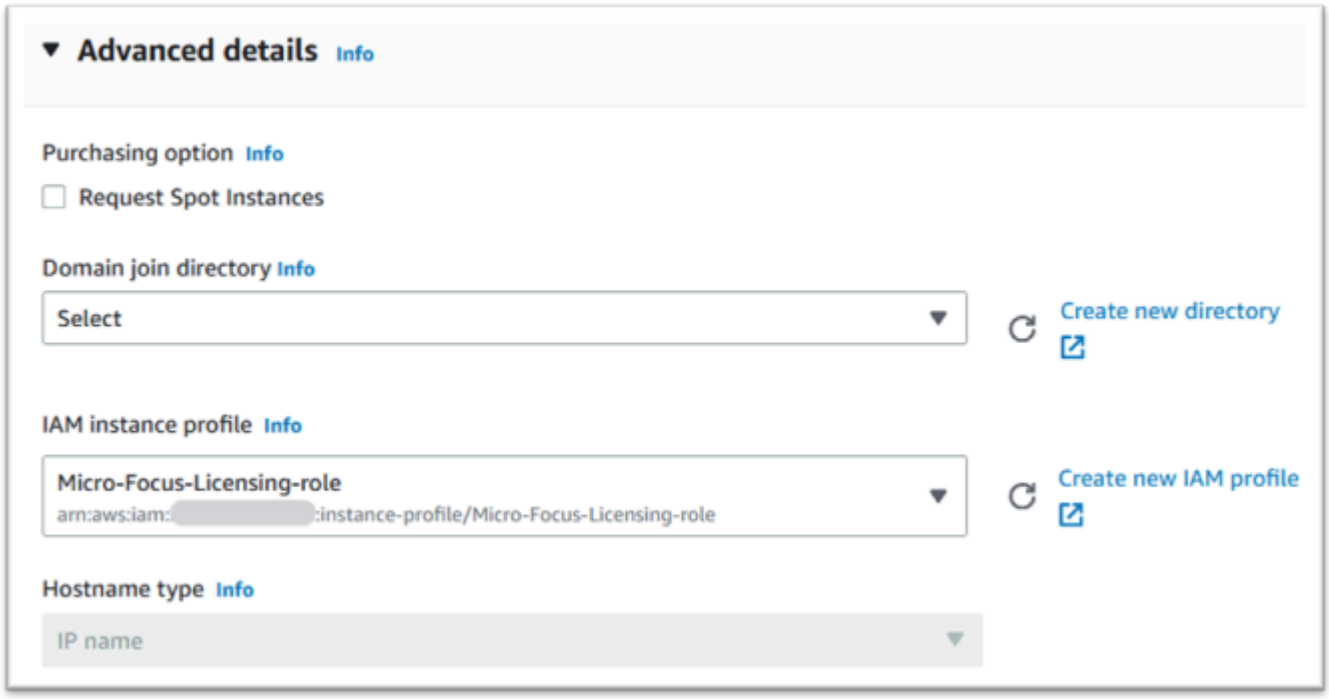

7. Escolha ou crie (e salve) um par de chaves (não exibido).

   Para obter mais informações sobre pares de chaves para instâncias do Linux, consulte Pares de chaves do Amazon EC2 e instâncias do Linux.

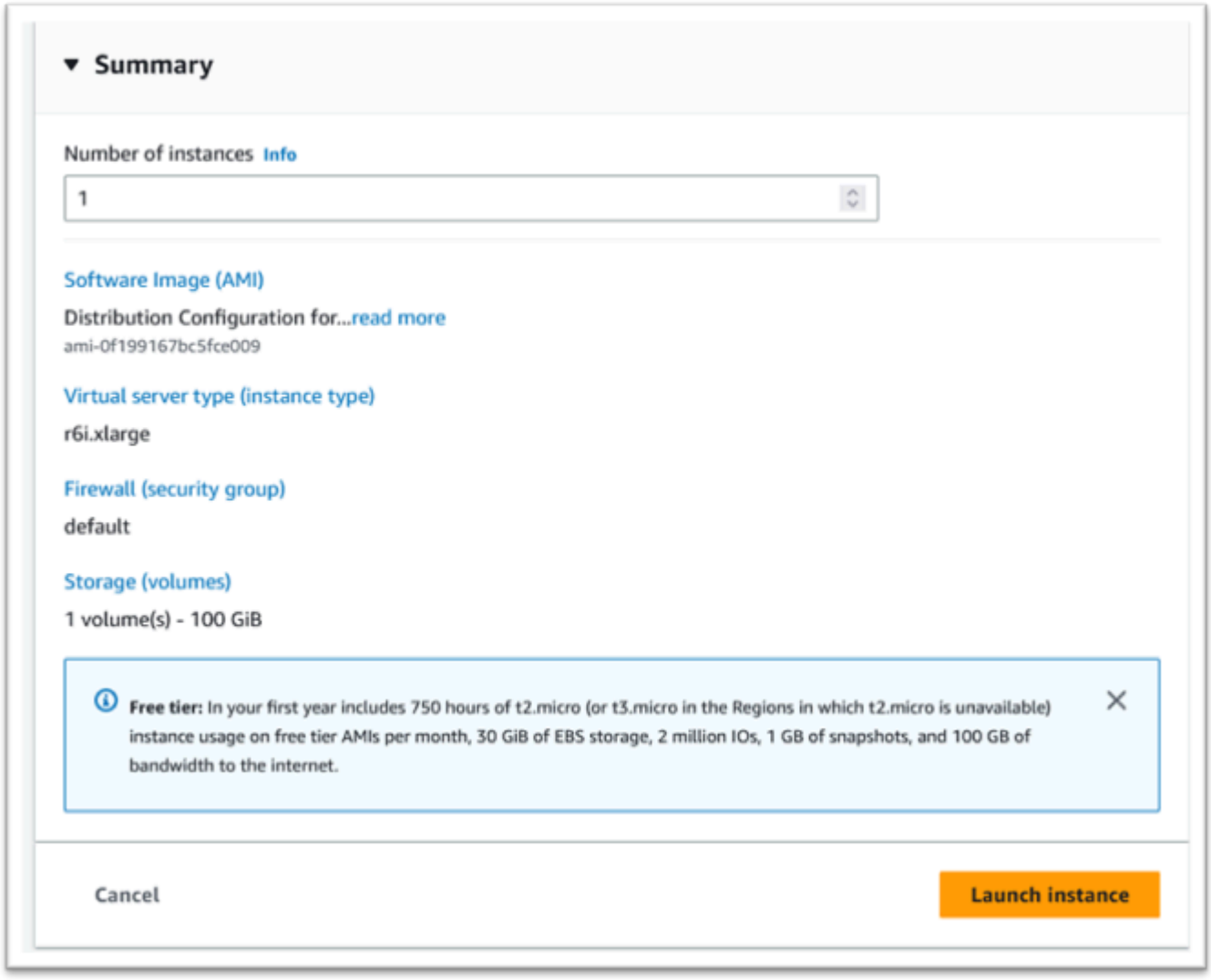
Para obter mais informações sobre pares de chaves para instâncias do Windows, consulte Pares de chaves do Amazon EC2 e instâncias do Windows.

8. Edite as configurações de rede e escolha a VPC permitida e a sub-rede apropriada.
9. Escolha Criar um novo grupo de segurança. Se for uma instância do Enterprise Server EC2, é normal permitir o tráfego TCP para as portas 86 e 10086 para administrar a configuração da Micro Focus.
10. Opcionalmente, configure o armazenamento para a instância do Amazon EC2.
11. Importante: expanda os detalhes avançados e, em Perfil da instância do IAM, escolha a função de licenciamento criada anteriormente, por exemplo, “Micro-Focus-Licensing-Role”.

> **Note**
>
> Se essa etapa for perdida, depois que a instância for criada, você poderá modificar a função do IAM na opção Segurança do menu Ação da instância EC2.

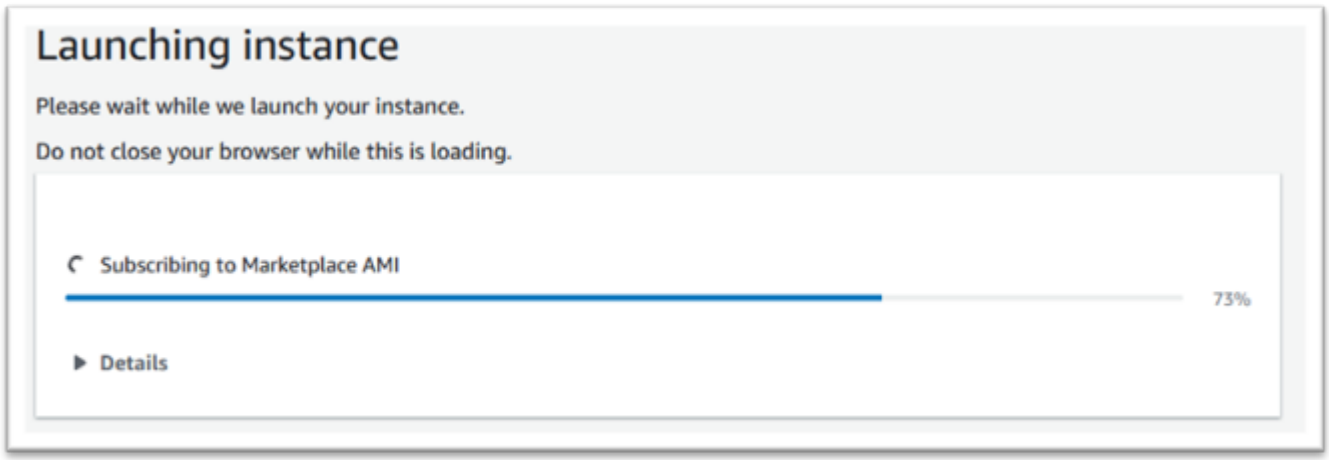



12. Revise o resumo e envie a Iniciar instância.

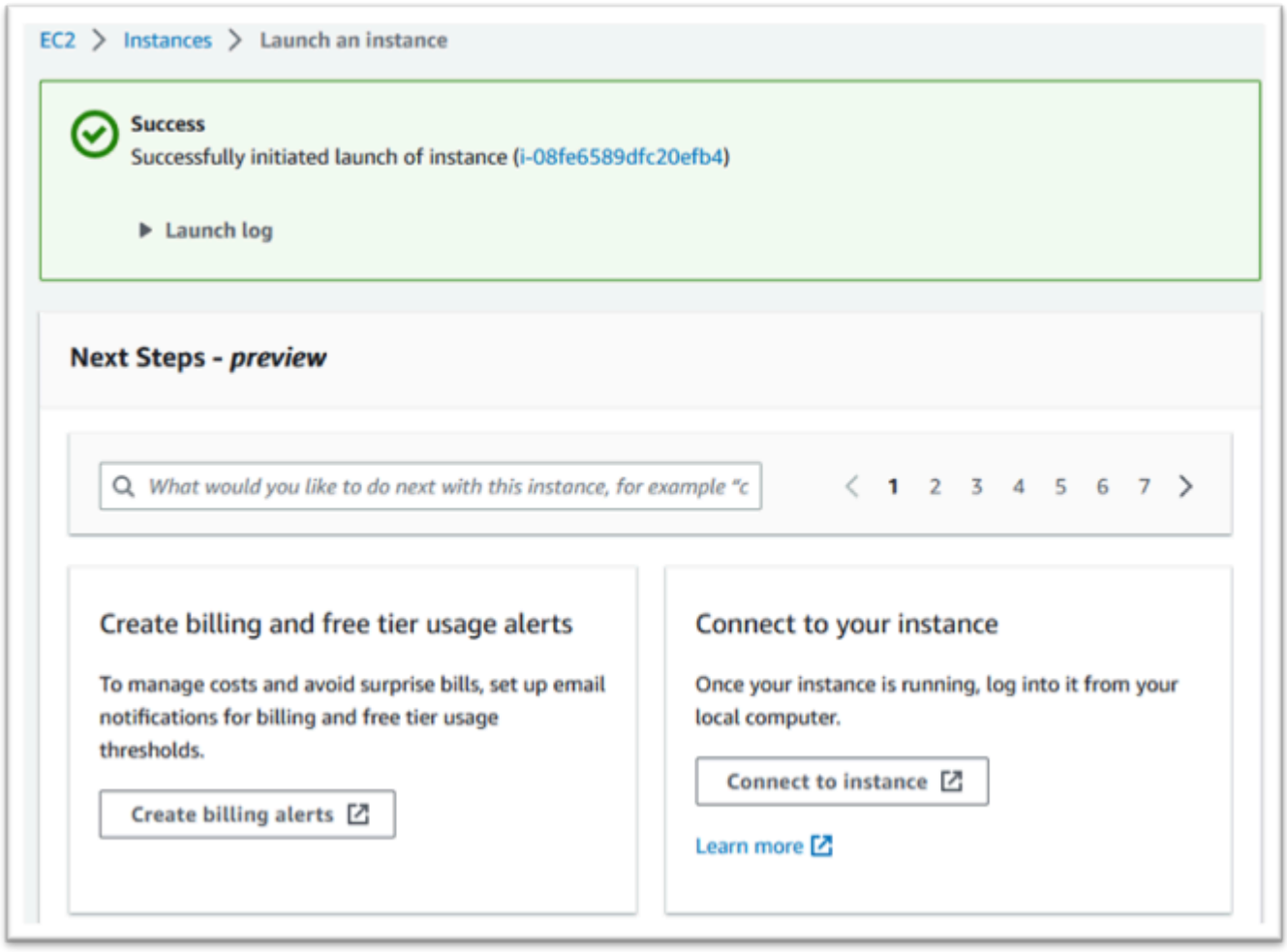



13. A inicialização da instância falhará se um tipo de servidor virtual inválido for escolhido.

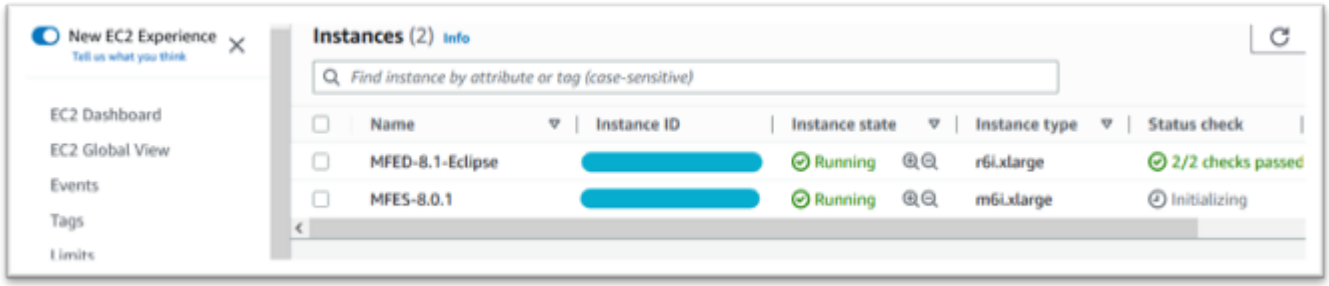
    Se isso acontecer, escolha Editar configuração da instância e altere o tipo de instância.



14. Depois que a mensagem “Sucesso” for exibida, escolha Conectar-se à instância para obter os detalhes da conexão.




15. Como alternativa, navegue até EC2 no AWS Management Console.
16. Escolha Instâncias para ver o status da nova instância.




## Sub-rede ou VPC sem acesso à Internet

Faça essas alterações adicionais se a sub-rede ou a VPC não tiver acesso de saída à Internet.

O gerenciador de licenças exige acesso aos seguintes serviços da AWS:

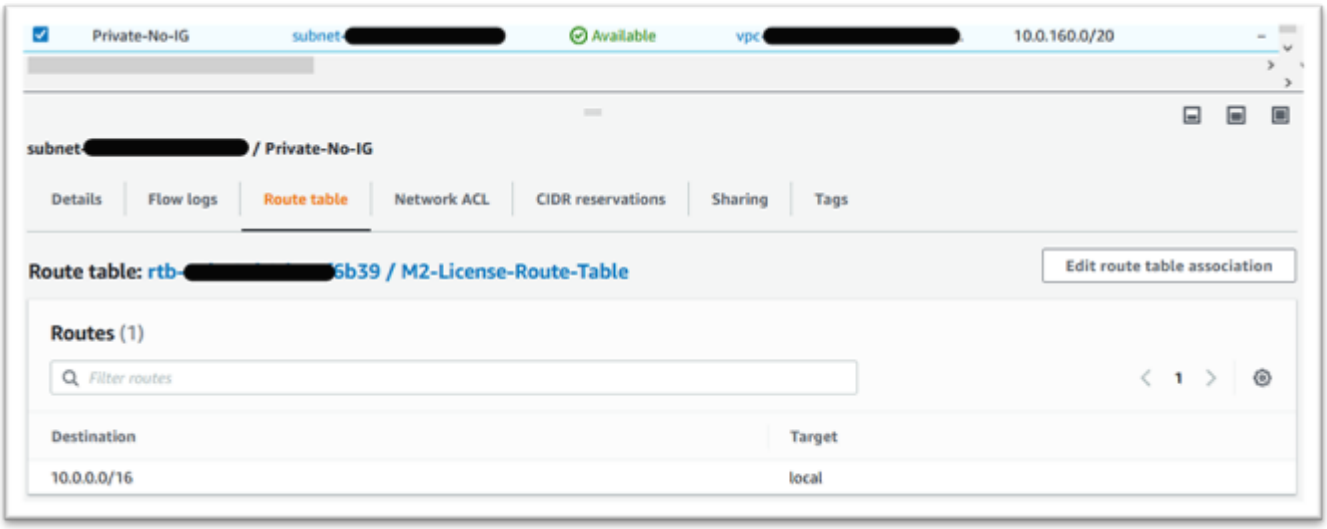
- com.amazonaws.*region*.s3
- com.amazonaws.*region*.ec2
- com.amazonaws.*region*.license-manager
- com.amazonaws.*region*.sts

As etapas anteriores definiram o com.amazonaws. serviço *region* .s3 como um endpoint de gateway. Esse endpoint precisa de uma entrada na tabela de rotas para qualquer sub-rede sem acesso à Internet.

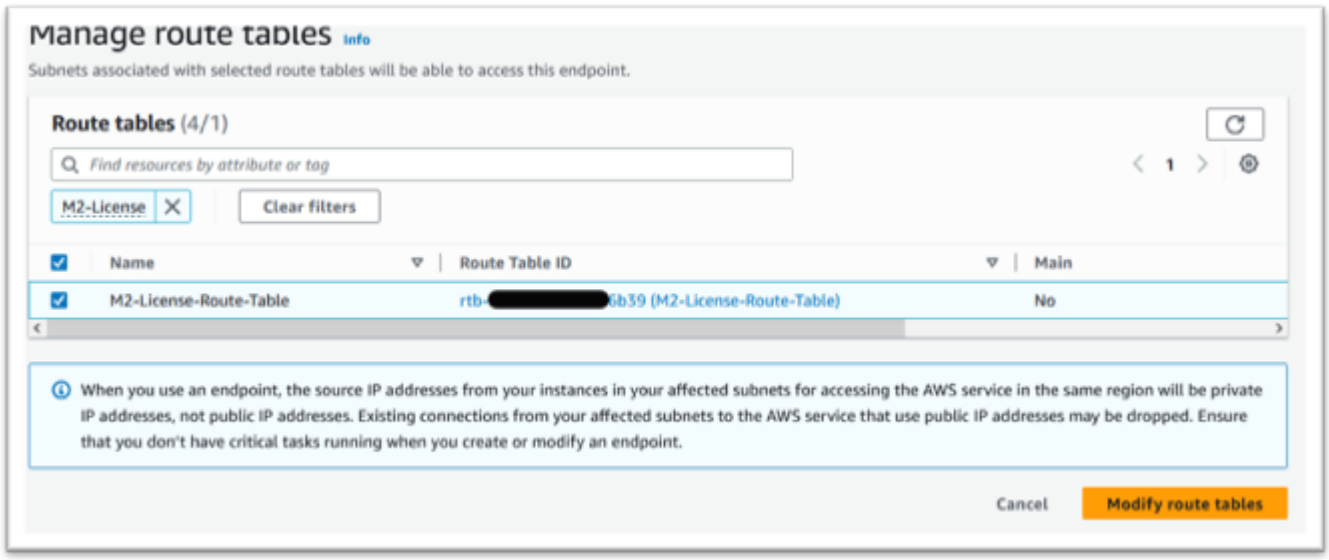
Os três serviços adicionais serão definidos como endpoints de interface.

## Adicione a entrada da tabela de rotas para o endpoint do Amazon S3

1. Navegue até VPC em AWS Management Console e escolha Sub-redes.
2. Selecione a sub-rede na qual as instâncias do Amazon EC2 serão criadas e selecione a guia Tabela de rotas.
3. Observe alguns dígitos finais do ID da tabela de rotas. Por exemplo, o 6b39 na imagem abaixo.




4. No painel de navegação, escolha Endpoints.
5. Escolha o endpoint criado anteriormente e, em seguida, Gerenciar tabelas de rotas, na guia Tabelas de rotas do endpoint ou no menu suspenso Ações.
6. Escolha a tabela de rotas usando os dígitos identificados anteriormente e pressione Modificar tabelas de rotas.

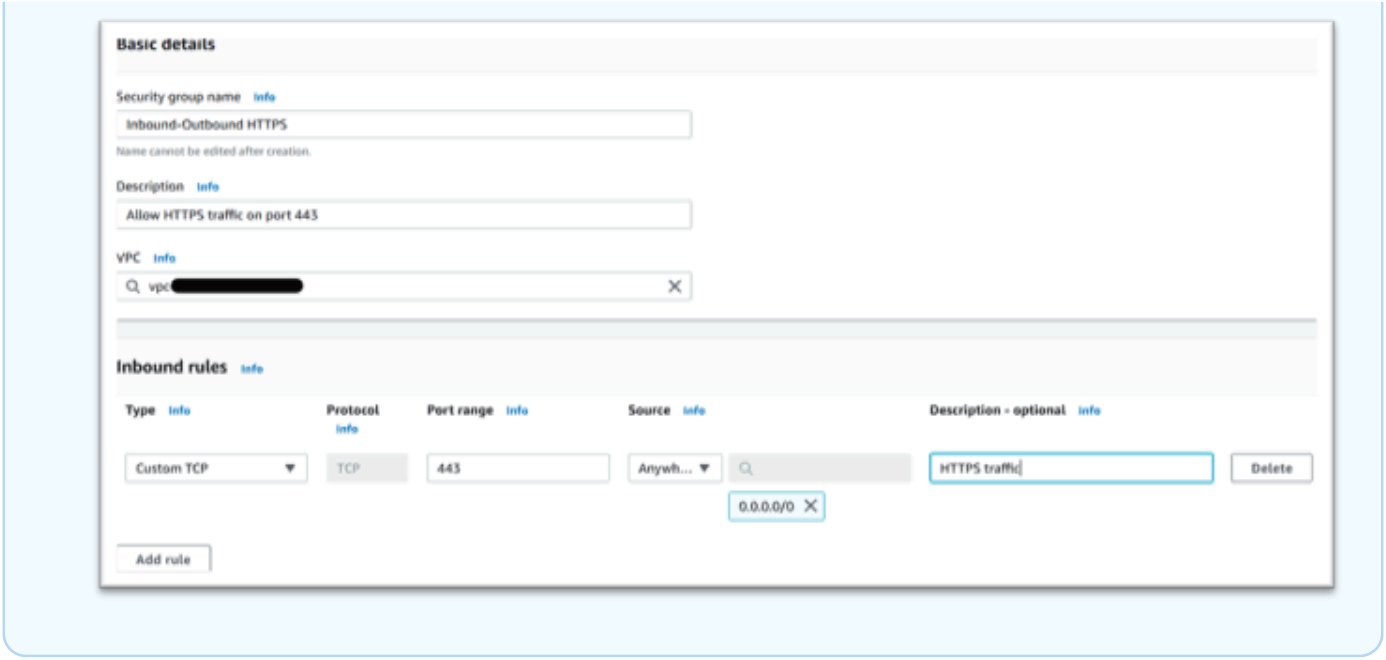



## Defina o grupo de segurança necessário

Os serviços Amazon EC2 e License Manager se comunicam via HTTPS pela porta 443. AWS STS Essa comunicação é bidirecional e requer regras de entrada e saída para permitir que a instância se comunique com os serviços.

1. Navegue até o Amazon VPC na seção AWS Management Console
2. Localize Grupos de segurança na barra de navegação e selecione Criar grupo de segurança.
3. Insira o nome e a descrição do grupo de segurança, por exemplo, “HTTPS de entrada e saída”.
4. Pressione o X na área de seleção da VPC para remover a VPC padrão e escolha a VPC que contém o endpoint do S3.
5. Adicione uma regra de entrada que permita o tráfego TCP na porta 443 de qualquer lugar.

> **Note**
>
> As regras de entrada (e saída) podem ser restringidas ainda mais limitando a Fonte. Para obter mais informações, consulte Controle o tráfego para seus AWS recursos usando grupos de segurança no Guia do usuário da Amazon VPC.

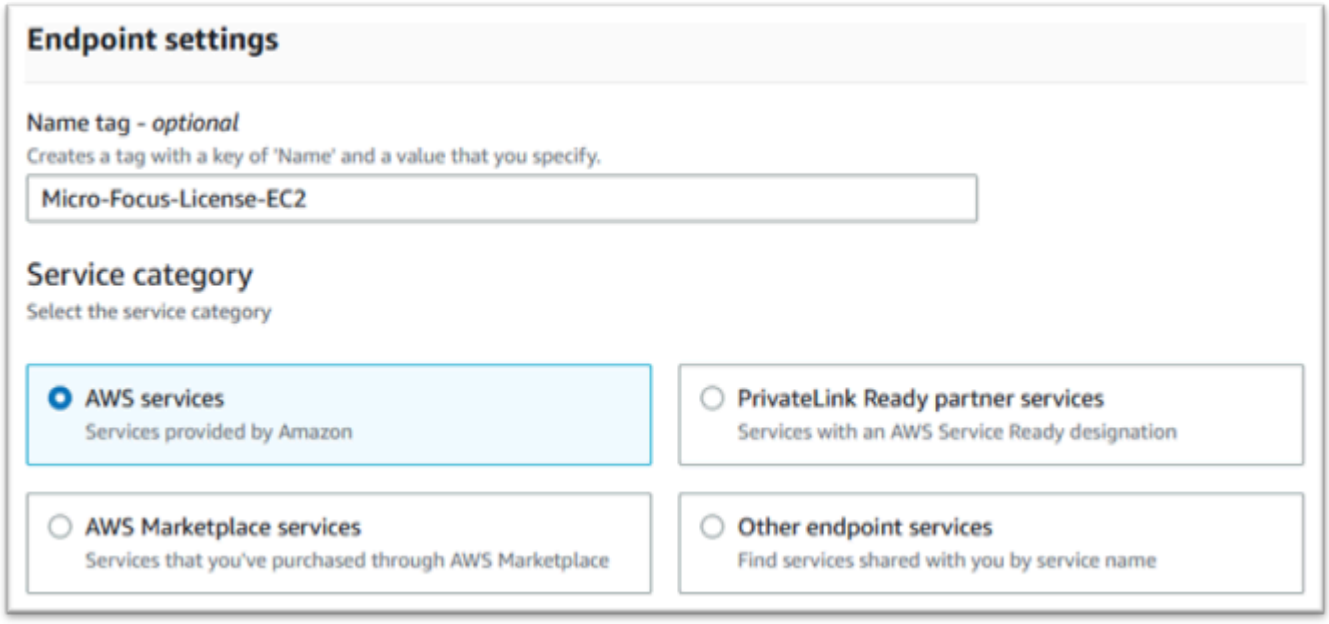



6. Pressione Criar grupo de segurança.

## Criar os endpoints de serviço

Repita esse processo três vezes — uma vez para cada serviço.

1. Navegue até Amazon VPC em AWS Management Console e escolha Endpoints.
2. Pressione Criar endpoint.
3. Insira um nome, por exemplo, “Micro-Focus-License-EC2”, “Micro-Focus-License-STS” ou “Micro-Focus-License-Manager”.
4. Escolha a categoria de serviço AWS Services.




5. Em Serviços, procure o serviço de interface correspondente, que é um dos seguintes:

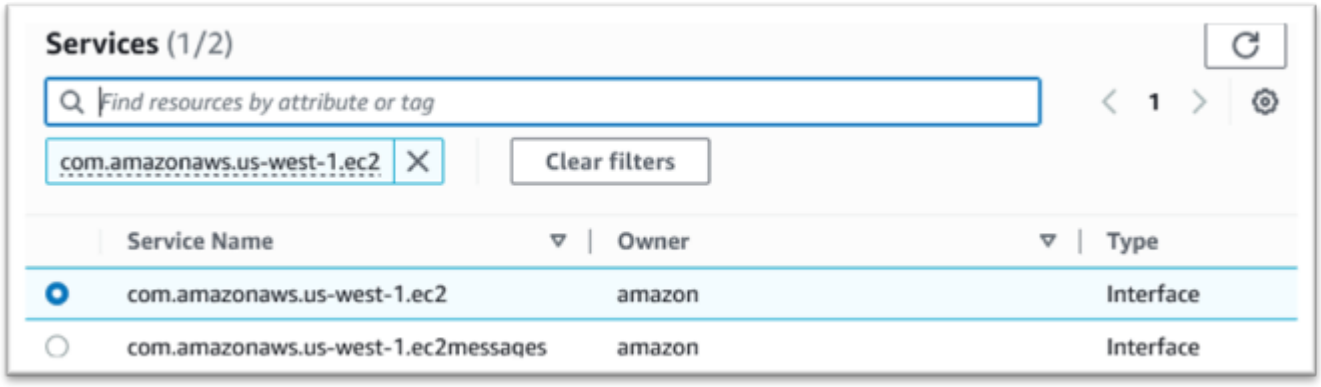
   - “com.amazonaws.*region*.ec2”
   - “com.amazonaws.*region*.sts”
   - “com.amazonaws.*region*.license-manager”

   Por exemplo: .

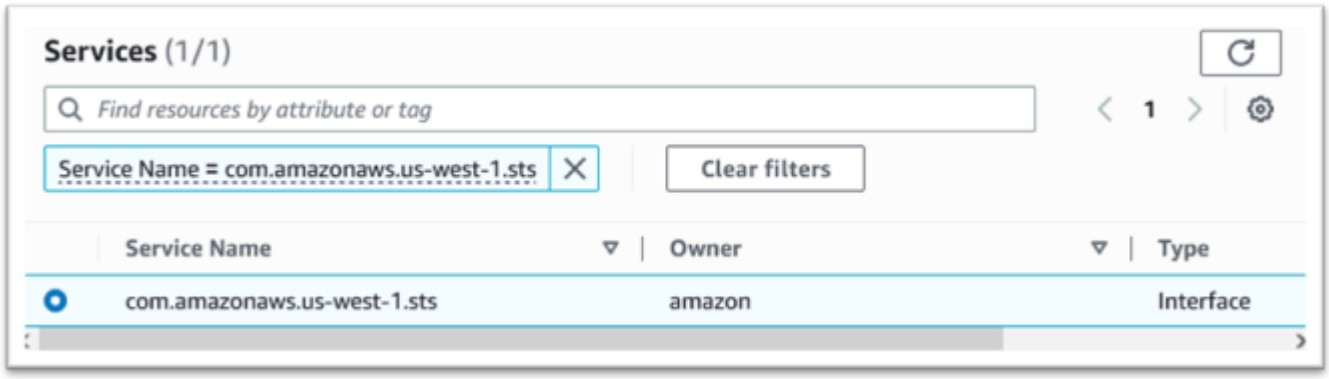
   - “com.amazonaws.us-west-1.ec2”
   - “com.amazonaws.us-west-1.sts”
   - “com.amazonaws.us-west-1.license-manager”

6. Escolha o serviço de interface correspondente.

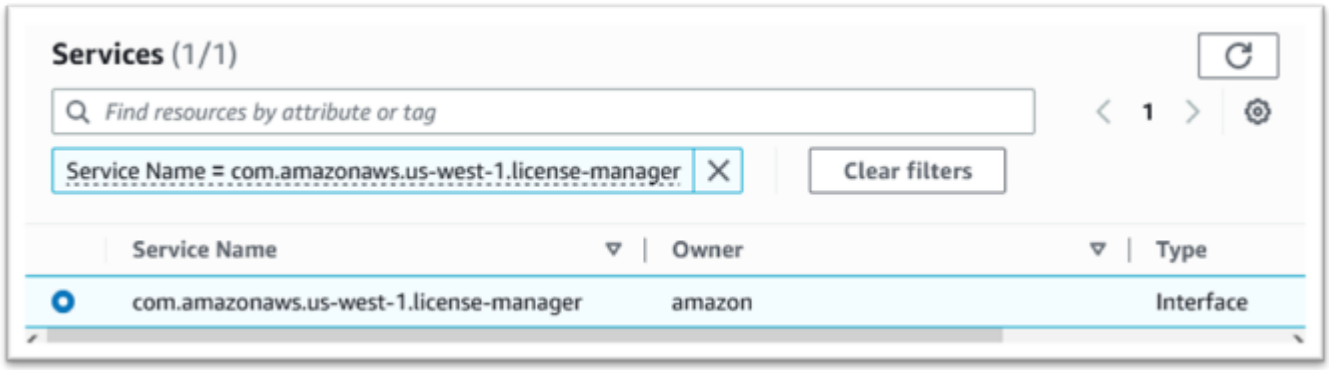
   com.amazonaws.***region***.ec2:



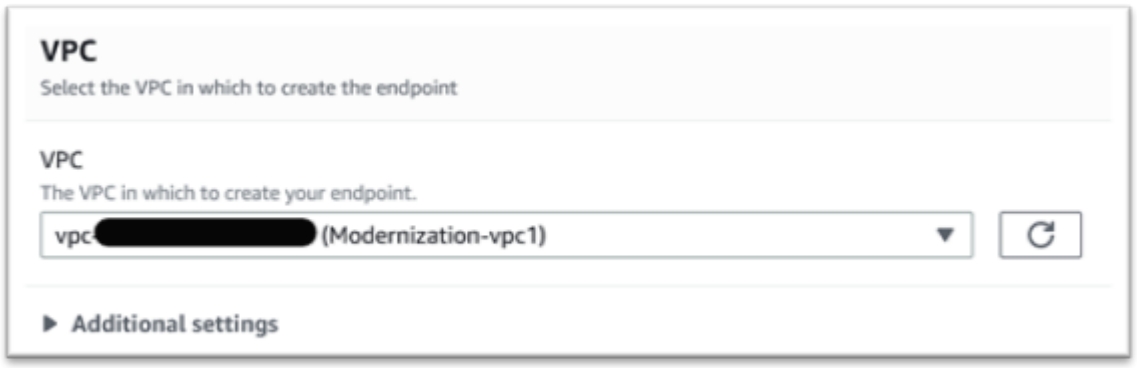

com.amazonaws.***region***.sts:



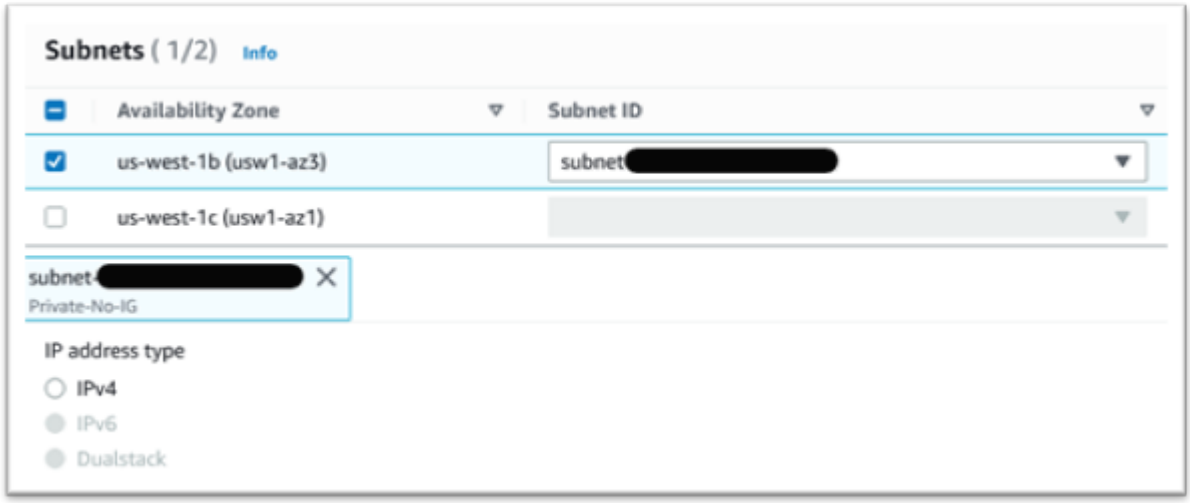

com.amazonaws.***region***.license-manager:



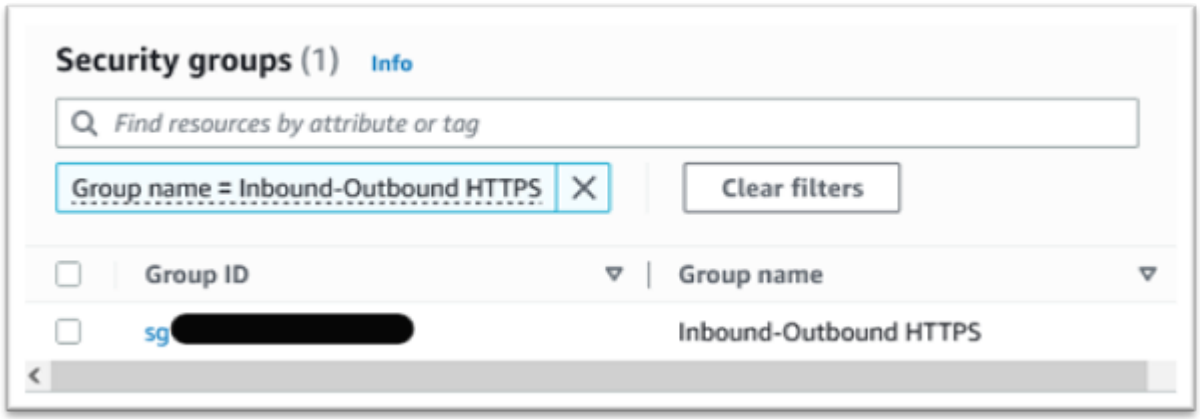

7. Para VPC, escolha a VPC para a instância.




8. Escolha a Zona de disponibilidade e as Sub-redes para a VPC.




9. Escolha Criar para criar o grupo de segurança.




10. Em Política, escolha Acesso total.

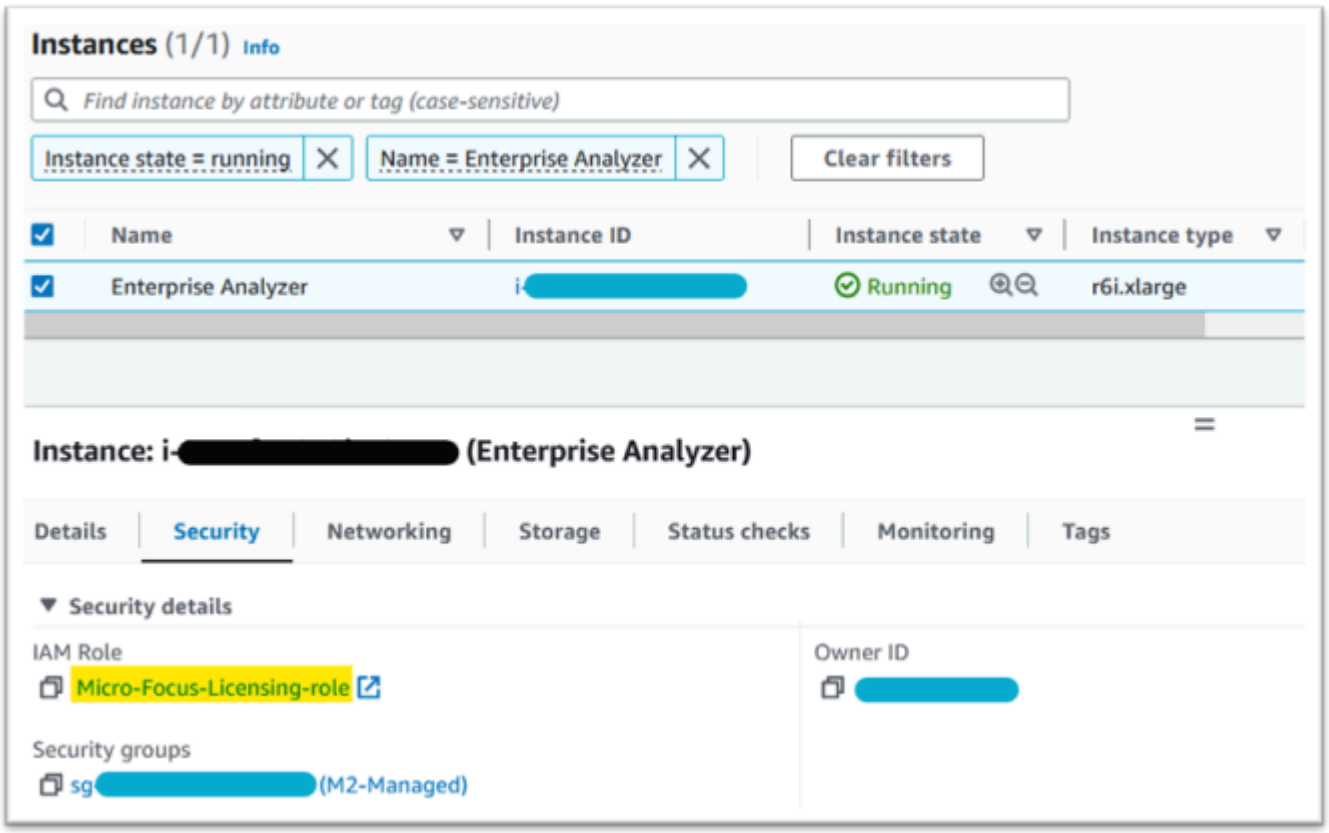

11. Escolha Criar endpoint.
12. Repita este processo para as interfaces restantes.

## Solução de problemas de licença

Se você tiver problemas para acessar ou usar as AMIs, as informações a seguir podem ajudá-lo.

### Verifique se a instância do Amazon EC2 tem a função de licenciamento do IAM

Isso pode ser verificado na guia Segurança dos detalhes da instância do Amazon EC2. Isso pode ser alterado usando a Opção de Segurança do menu suspenso Ações.




## Use o Reachability Analyzer

Encontre o Reachability Analyzer na página do console. AWS Network Manager

Crie e analise um caminho entre a instância do Amazon EC2 criada a partir da AMI e o Amazon S3 VPC Endpoint.

Se a instância do Amazon EC2 não tiver acesso à Internet, repita a análise do caminho para todos os 4 endpoints.

Para obter mais informações sobre o Reachability Analyzer, consulte Introdução ao Reachability Analyzer no guia do Reachability Analyzer.

## Execute o daemon de licença

No Windows Enterprise Developer, use o seguinte comando em um prompt de comando:

```
“C:\Program Files (x86)\Micro Focus\Enterprise Developer\AdoptOpenJDK\bin\java” -jar
 "C:\Program Files (x86)\Micro Focus\Licensing\aws-license-daemon.jar"
```

e examine a saída. Ignore as mensagens do SLF4J e procure a primeira exceção.

No Enterprise Analyzer, use o seguinte comando em um prompt de comando:

```
"C:\Program Files (x86)\Micro Focus\AdoptOpenJDK\bin\java" -jar "C:\Program Files
 (x86)\Micro Focus\Licensing\aws-license-daemon.jar"
```

e examine a saída. Ignore as mensagens do SLF4J e procure a primeira exceção.

No Linux, execute:

```
java -jar /var/microfocuslicensing/bin/aws-license-daemon.jar
```

Ignore as mensagens do SLF4J e procure a primeira exceção.

Por exemplo, se o recurso Amazon S3 não estiver disponível, a exceção será a seguinte:

```
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
```

A mensagem de exceção indica qual recurso não está disponível. Compare os valores de configuração com os mostrados neste tópico.

# Tutorial: Configurar o AppStream 2.0 para uso com o Micro Focus Enterprise Analyzer e o Micro Focus Enterprise Developer

O AWS Mainframe Modernization fornece várias ferramentas por meio do Amazon AppStream 2.0. O AppStream 2.0 é um serviço de streaming de aplicativos totalmente gerenciado e seguro que permite transmitir aplicativos de desktop aos usuários sem precisar reescrevê-los. O AppStream 2.0 fornece aos usuários acesso instantâneo aos aplicativos de que precisam com uma experiência de usuário responsiva e fluida no dispositivo de sua escolha. O uso do AppStream 2.0 para hospedar ferramentas específicas do mecanismo de execução oferece às equipes de aplicativos do cliente a capacidade de usar as ferramentas diretamente de seus navegadores da web, interagindo com arquivos de aplicativos armazenados em buckets do Amazon S3 ou repositórios do CodeCommit.

Para obter informações sobre o suporte de navegadores no AppStream 2.0, consulte Requisitos do sistema e suporte a recursos (navegador da Web) no Guia de administração do Amazon AppStream 2.0. Se você tiver problemas ao usar o AppStream 2.0, consulte Solução de problemas do usuário do AppStream 2.0 no Guia de administração do Amazon AppStream 2.0.

Este documento é destinado aos membros da equipe de operações do cliente. Ele descreve como configurar frotas e pilhas do Amazon AppStream 2.0 para hospedar as ferramentas Micro Focus Enterprise Analyzer e Micro Focus Enterprise Developer usadas com o AWS Mainframe Modernization. O Micro Focus Enterprise Analyzer geralmente é usado durante a fase de avaliação e o Micro Focus Enterprise Developer geralmente é usado durante a fase de migração e modernização da abordagem do AWS Mainframe Modernization. Se você planeja usar o Enterprise Analyzer e o Enterprise Developer, deve criar frotas e pilhas separadas para cada ferramenta. Cada ferramenta requer sua própria frota e pilha porque seus termos de licenciamento são diferentes.

> **Important**
>
> As etapas deste tutorial são baseadas no modelo AWS CloudFormation disponível para download cfn-m2-appstream-fleet-ea-ed.yml.

Tópicos



## Pré-requisitos

- Faça o download do modelo: cfn-m2-appstream-fleet-ea-ed.yml.
- Obtenha o ID da sua VPC e do grupo de segurança padrão. Para obter mais informações sobre o VPC padrão, consulte VPCs padrão no Guia do usuário do Amazon VPC. Para obter mais

informações sobre o grupo de segurança padrão, consulte Grupos de segurança padrão e personalizados no Guia do usuário do Amazon EC2 para instâncias Linux.

- Certifique-se de que você tenha as seguintes permissões:
  - criar pilhas, frotas e usuários no AppStream 2.0.
  - crie pilhas em AWS CloudFormation usando um modelo.
  - crie buckets e faça upload de arquivos para buckets no Amazon S3.
  - baixe as credenciais (`access_key_id` e `secret_access_key`) do IAM.

## Etapa 1: obter as imagens do AppStream 2.0

Nesta etapa, você compartilha as imagens do AppStream 2.0 para Enterprise Analyzer e Enterprise Developer com sua conta AWS.

1. Abra o console do AWS Mainframe Modernization em https://console.aws.amazon.com/m2/.
2. Na navegação à esquerda, selecione Ferramentas.
3. Em Análise, desenvolvimento e criação de ativos, escolha Compartilhar ativos com minha contaAWS.

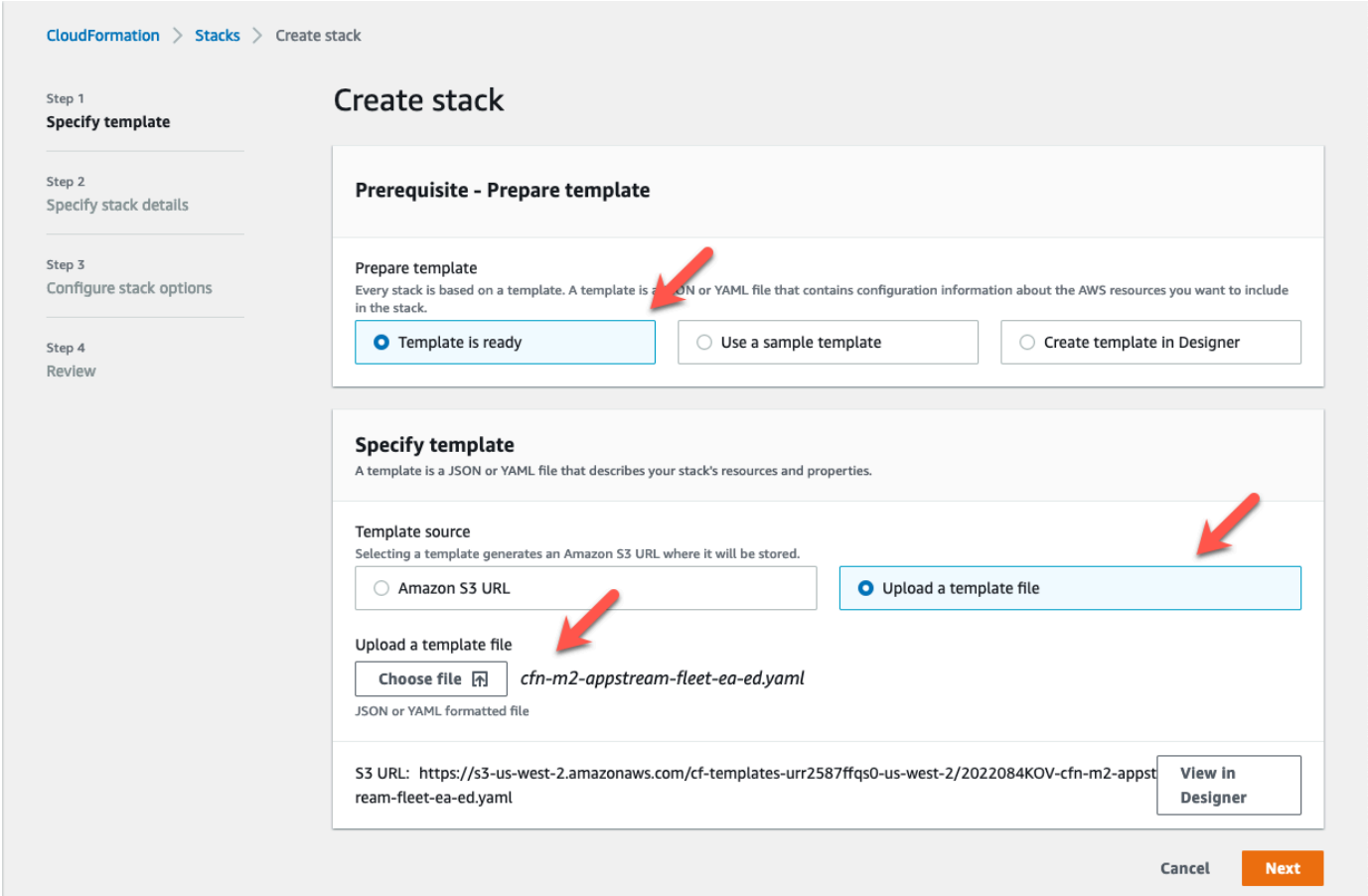
## Etapa 2: criar a pilha usando o modelo AWS CloudFormation

Nesta etapa, você usa o modelo AWS CloudFormation baixado para criar uma pilha e uma frota do AppStream 2.0 para executar o Micro Focus Enterprise Analyzer. Você pode repetir essa etapa posteriormente para criar outra pilha e frota do AppStream 2.0 para executar o Micro Focus Enterprise Developer, pois cada ferramenta exige sua própria frota e pilha no AppStream 2.0. Para obter mais informações sobre pilhas AWS CloudFormation, consulte Trabalhar com pilhas no Guia do UsuárioAWS CloudFormation.

> **Note**
>
> AWS Mainframe Modernization adiciona uma taxa adicional ao preço padrão do AppStream 2.0 para o uso do Enterprise Analyzer e do Enterprise Developer. Para obter mais00 informações, consulte Preço do AWS Mainframe Modernization.

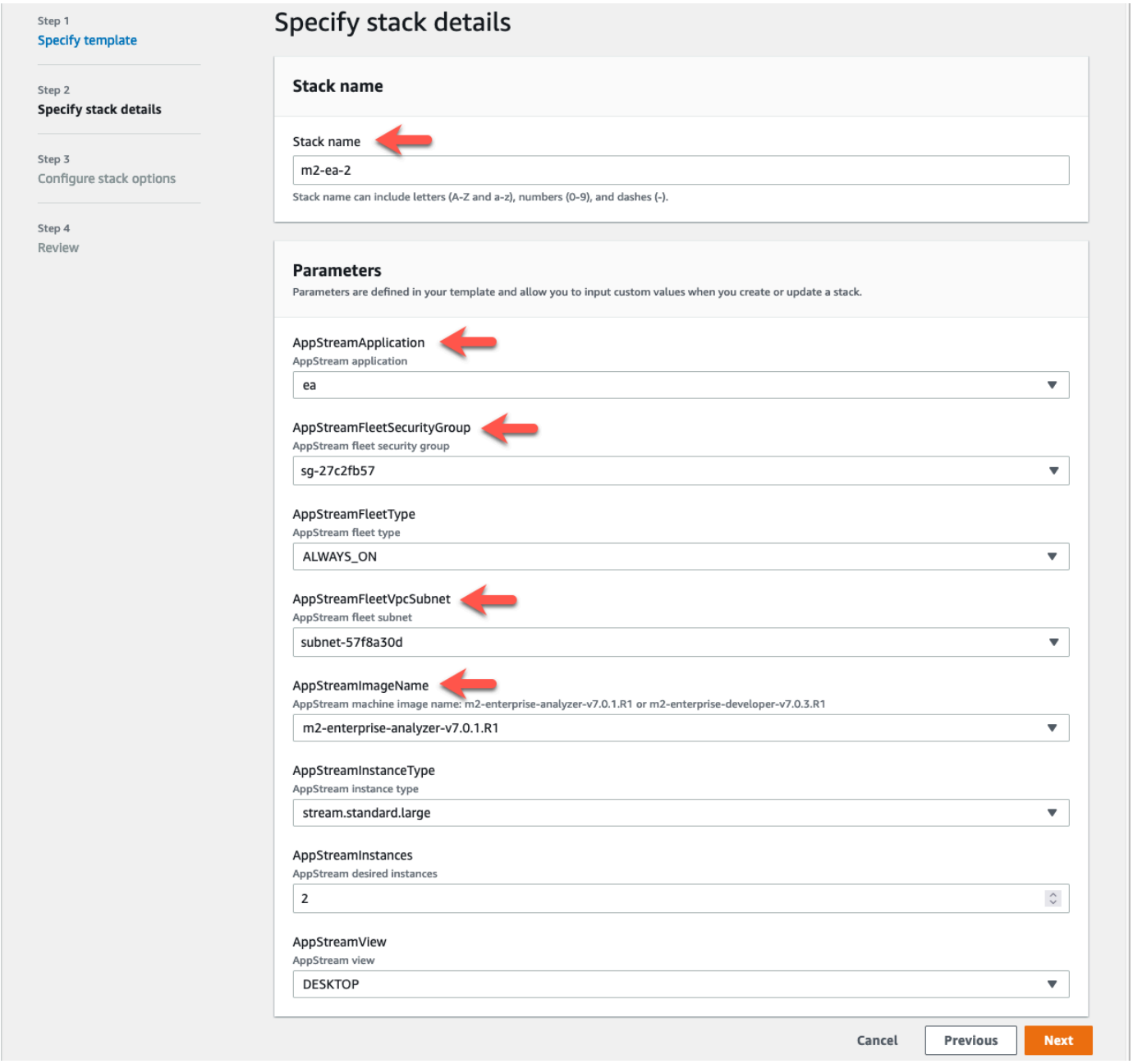
1. Baixe o modelo cfn-m2-appstream-fleet-ea-ed.yml, se necessário.

2. Abra o console AWS CloudFormation e escolha Criar pilha e com novos recursos (padrão).
3. Em Pré-requisito - Preparar modelo, selecione O modelo está pronto.
4. Em Especificar modelo, selecione Upload de um arquivo de modelo.
5. Em Carregar um arquivo de modelo, escolha Escolher arquivo e carregue o modelo [cfn-m2-appstream-fleet-ea-ed.yml](cfn-m2-appstream-fleet-ea-ed.yml).
6. Escolha Next (Próximo).




7. Em Especificar detalhes da pilha, insira as seguintes informações

   - Em Nome da pilha, insira um nome de sua escolha. Por exemplo, **`m2-ea`**.
   - No aplicativo AppStream, escolha ea.
   - No AppStreamFleetSecurityGroup, escolha o grupo de segurança padrão da sua VPC padrão.
   - No AppStreamFleetVPCSubnet, escolha uma sub-rede dentro da sua VPC padrão.
   - Em AppStreamImageName, escolha a imagem que começa com `m2-enterprise-analyzer`. Essa imagem contém a versão atualmente suportada da ferramenta Micro Focus Enterprise Analyzer.
   - Aceite os padrões para os outros campos e selecione Próximo.




8. Aceite todos os padrões e selecione Próximo novamente.
9. Na revisão, certifique-se de que todos os parâmetros sejam o que você pretende.
10. Role até a parte inferior, escolha Eu reconheço que o AWS CloudFormation pode criar recursos IAM com nomes personalizados e escolha Criar pilha.

A pilha e a frota demoram entre 20 e 30 minutos para serem criadas. Você pode escolher Atualizar para ver os AWS CloudFormation eventos à medida que eles ocorrem.

# Etapa 3: criar um usuário no AppStream 2.0

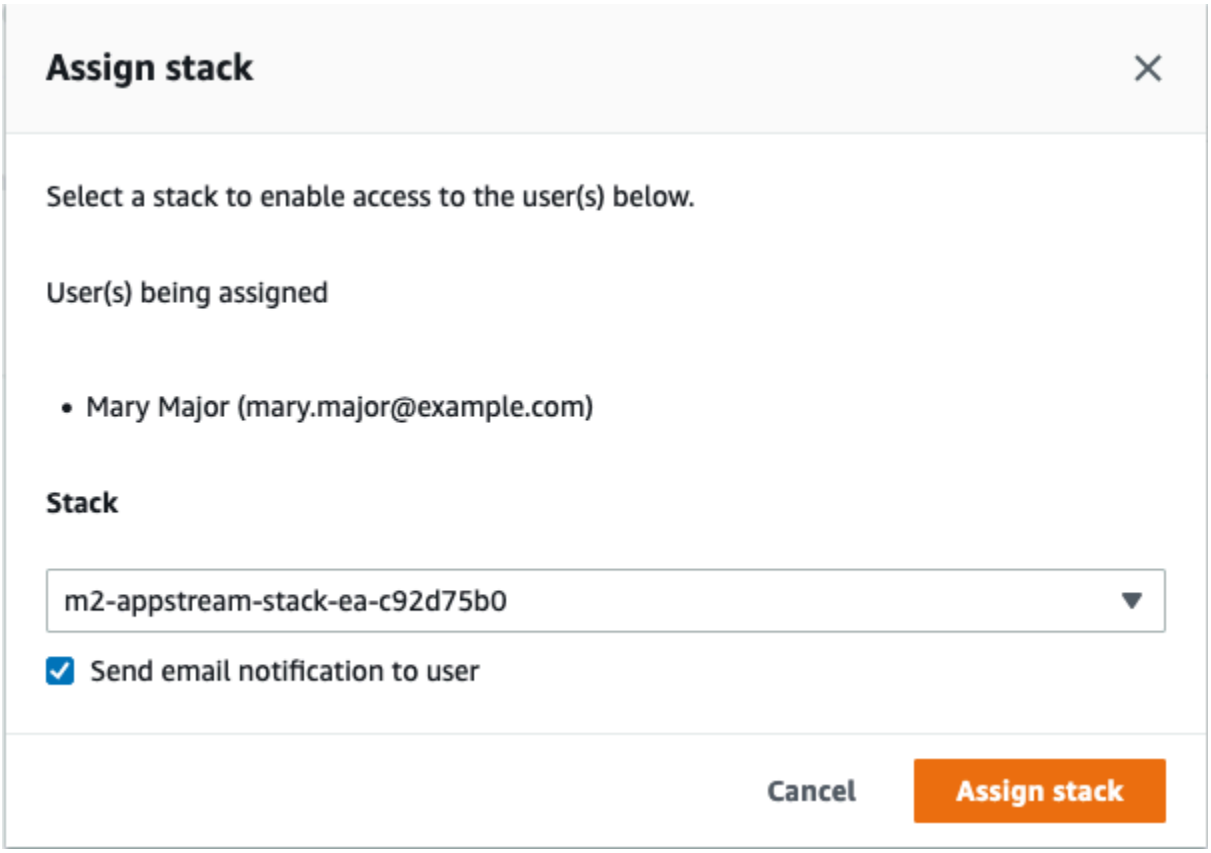
Enquanto aguarda o AWS CloudFormation terminar de criar a pilha, você pode criar um ou mais usuários no AppStream 2.0. Esses usuários usarão o Enterprise Analyzer no AppStream 2.0. Você precisará especificar um endereço de e-mail para cada usuário e garantir que cada usuário tenha permissões suficientes para criar buckets no Amazon S3, fazer upload de arquivos em um bucket e vincular a um bucket para mapear seu conteúdo.

1. Abra o console do AppStream 2.0.
2. Na navegação à esquerda, selecione Grupo de usuários.
3. Escolha Criar usuário.
4. Forneça um endereço de e-mail em que o usuário possa receber um convite por e-mail para usar o AppStream 2.0, um nome e sobrenome e escolha Criar usuário.
5. Repita, se necessário, para criar mais usuários. O endereço de e-mail de cada usuário deve ser exclusivo.

Para obter mais informações sobre a criação de usuários do Amazon AppStream 2.0, consulte Grupos de usuários do Amazon AppStream 2.0 no Guia de administração do Amazon AppStream 2.0.

Quando o AWS CloudFormation terminar de criar a pilha, você poderá atribuir o usuário que criou à pilha, como segue:

1. Abra o console do AppStream 2.0.
2. Clique no nome de usuário.
3. Escolha Ação e, em seguida, Atribuir pilha.
4. Em Atribuir pilha, escolha a pilha que começa com `m2-appstream-stack-ea`.
5. Escolha Assign stack.




Atribuir um usuário a uma pilha faz com que o AppStream 2.0 envie um e-mail para o usuário no endereço que você forneceu. Esse e-mail contém um link para a página de login do AppStream 2.0.

## Etapa 4: fazer login no AppStream 2.0

Nesta etapa, você faz login no AppStream 2.0 usando o link no e-mail enviado pelo AppStream 2.0 para o usuário no qual você criou Etapa 3: criar um usuário no AppStream 2.0.

1. Faça login no AppStream 2.0 usando o link fornecido no e-mail enviado pelo AppStream 2.0.
2. Altere sua senha, se solicitado. A tela do AppStream 2.0 que você vê é semelhante à seguinte:


3. Escolha Desktop.
4. Na barra de tarefas, escolha Pesquisar e digite **`D:`** para navegar até a Pasta Pessoal.

> **Note**
>
> Se você pular essa etapa, poderá receber uma `Device not ready` mensagem de erro ao tentar acessar a Pasta Pessoal.

A qualquer momento, se tiver problemas para fazer login no AppStream 2.0, você pode reiniciar sua frota do AppStream 2.0 e tentar fazer login novamente, usando as etapas a seguir.

1. Abra o console do AppStream 2.0.
2. No painel de navegação à esquerda, escolha Frotas.
3. Escolha a frota que você está tentando usar.
4. Escolha Ação e, em seguida, escolha Parar.
5. Espere a frota parar.
6. Escolha Ação e, em seguida, escolha Iniciar.

Esse processo pode levar cerca de 10 minutos.

## Etapa 5: verificar os buckets no Amazon S3 (opcional)

Uma das tarefas concluídas pelo modelo AWS CloudFormation que você usou para criar a pilha foi criar dois buckets no Amazon S3, que são necessários para salvar e restaurar os dados do usuário e as configurações do aplicativo nas sessões de trabalho. Esses baldes são os seguintes:

- O nome começa com `appstream2-`. Esse bucket mapeia dados para sua pasta inicial no AppStream 2.0 (`D:\PhotonUser\My Files\Home Folder`).

  > **Note**
  >
  > A Pasta Inicial é exclusiva para um determinado endereço de e-mail e é compartilhada entre todas as frotas e pilhas em uma determinada conta AWS. O nome da Pasta Pessoal é um hash SHA256 do endereço de e-mail do usuário e é armazenado em um caminho baseado nesse hash.

- O nome começa com `appstream-app-settings-`. Esse bucket contém informações da sessão do usuário para o AppStream 2.0 e inclui configurações como favoritos do navegador, perfis de conexão do IDE e do aplicativo e personalizações da interface do usuário. Para obter mais informações, consulte Como funciona a persistência das configurações do aplicativo no Guia de administração do Amazon AppStream 2.0.

Para verificar se os buckets foram criados, siga estas etapas:

1. Abra o console do Amazon S3.
2. Na navegação à esquerda, escolha Buckets.
3. Em Localizar compartimentos por nome, insira **`appstream`** para filtrar a lista.

Não é necessário realizar nenhuma ação adicional se você ver os compartimentos. Esteja ciente de que os baldes existem. Se você não vir os compartimentos, o modelo AWS CloudFormation não terminou de ser executado ou ocorreu um erro. Vá para o console do AWS CloudFormation e examine as mensagens de criação de pilha.

## Próximas etapas

Agora que a infraestrutura do AppStream 2.0 está configurada, você pode configurar e começar a usar o Enterprise Analyzer. Para obter mais informações, consulte Tutorial: Configurar o Enterprise

Analyzer no AppStream 2.0. Também é possível configurar o Enterprise Developer. Para obter mais informações, consulte Tutorial: Configurar o Micro Focus Enterprise Developer no AppStream 2.0.

### Limpar os recursos

O procedimento para limpar a pilha e as frotas criadas é descrito em Criar uma frota e pilha do AppStream 2.0.

Quando os objetos do AppStream 2.0 são excluídos, o administrador da conta também pode, se apropriado, limpar os buckets do Amazon S3 para as configurações do aplicativo e as pastas iniciais.

> **Note**
>
> A pasta inicial de um determinado usuário é exclusiva em todas as frotas, portanto, talvez seja necessário mantê-la se outras pilhas do AppStream 2.0 estiverem ativas na mesma conta.

Por fim, o AppStream 2.0 atualmente não permite que você exclua usuários usando o console. Em vez disso, você deve usar a API de serviço com a CLI. Para obter mais informações, consulte Administração do grupo de usuários no Guia de administração do Amazon AppStream 2.0.

## Tutorial: Configurar o Enterprise Analyzer no AppStream 2.0

Este tutorial descreve como configurar o Micro Focus Enterprise Analyzer para analisar um ou mais aplicativos de mainframe. A ferramenta Enterprise Analyzer fornece vários relatórios com base em sua análise do código-fonte do aplicativo e das definições do sistema.

Essa configuração foi projetada para promover a colaboração em equipe. A instalação usa um bucket do Amazon S3 para compartilhar o código-fonte com discos virtuais. Isso faz uso do Rclone) na máquina Windows. Com uma instância comum do Amazon RDS executando o PostgreSQL, qualquer membro da equipe pode acessar todos os relatórios solicitados.

Os membros da equipe também podem montar o disco virtual com suporte do Amazon S3 em suas máquinas pessoais e atualizar o bucket de origem a partir de suas estações de trabalho. Eles podem potencialmente usar scripts ou qualquer outra forma de automação em suas máquinas se estiverem conectados a outros sistemas internos locais.

A configuração é baseada nas imagens do AppStream 2.0 do Windows que o AWS Mainframe Modernization compartilha com o cliente. A configuração também se baseia na criação de frotas e

pilhas do AppStream 2.0, conforme descrito em Tutorial: Configurar o AppStream 2.0 para uso com o Micro Focus Enterprise Analyzer e o Micro Focus Enterprise Developer.

> **Important**
>
> As etapas deste tutorial pressupõem que você configurou o AppStream 2.0 com o modelo de AWS CloudFormation cfn-m2-appstream-fleet-ea-ed.yml que pode ser baixado. Para obter mais informações, consulte Tutorial: Configurar o AppStream 2.0 para uso com o Micro Focus Enterprise Analyzer e o Micro Focus Enterprise Developer.
> Para executar as etapas neste tutorial, você deve ter configurado a frota e a pilha do Enterprise Analyzer e elas devem estar em execução.

Para obter uma descrição completa dos recursos e resultados do Enterprise Analyzer, consulte a documentação do Enterprise Analyzer no site da Micro Focus.

## Conteúdo da imagem

Além do próprio aplicativo Enterprise Analyzer, a imagem contém as seguintes ferramentas e bibliotecas.

Ferramentas de terceiros

- Python
- Rclone
- pgAdmin
- git-scm
- Driver ODBC PostgreSQL

Bibliotecas em `C:\Users\Public`

- Código-fonte do BankDemo e definição do projeto para Enterprise Developer: `m2-bankdemo-template.zip`.
- Pacote de instalação do MFA para o mainframe: `mfa.zip`. Para obter mais informações, consulte Visão geral do acesso ao mainframe na documentação do Micro Focus Enterprise Developer.
- Arquivos de comando e configuração do Rclone (instruções para seu uso nos tutoriais): `m2-rclone.cmd` e `m2-rclone.conf`.

Tópicos



## Pré-requisitos

- Faça upload do código-fonte e das definições do sistema para o aplicativo do cliente que você deseja analisar em um bucket do S3. As definições do sistema incluem CICS CSD, definições de objetos do DB2 e assim por diante. Você pode criar uma estrutura de pastas dentro do bucket que faça sentido para a forma como você deseja organizar os artefatos do aplicativo. Por exemplo, quando você descompacta a amostra BankDemo, ela tem a seguinte estrutura:

```
demo
     |--> jcl
     |--> RDEF
     |--> transaction
     |--> xa
```

- Criar uma instância de banco de dados do Amazon RDS que executa PostgreSQL e conectar-se a ela Essa instância armazenará os dados e os resultados produzidos pelo Enterprise Analyzer. Você pode compartilhar essa instância com todos os membros da equipe do aplicativo. Além disso, crie um esquema vazio chamado `m2_ea` (ou qualquer outro nome adequado) no banco de dados. Defina credenciais para usuários autorizados que lhes permitam criar, inserir, atualizar e excluir itens nesse esquema. Você pode obter o nome do banco de dados, o URL do endpoint do servidor e a porta TCP no console do Amazon RDS ou no administrador da conta.
- Certifique-se de ter configurado o acesso programático ao seu Conta da AWS. Para obter mais informações, consulte Acessar o  para o Referência geral da Amazon Web Services no .

## Etapa 1: Configurar o

1. Inicie uma sessão com o AppStream 2.0 com a URL que você recebeu na mensagem de e-mail de boas-vindas do AppStream 2.0.
2. Use seu e-mail como ID de usuário e defina sua senha permanente.
3. Selecione a pilha do Enterprise Analyzer.
4. Na página de menu do AppStream 2.0, escolha Desktop para acessar a área de trabalho do Windows que a frota está transmitindo.

## Etapa 2: Criar a pasta virtual baseada no Amazon S3 no Windows

> **Note**
>
> Se você já usou o Rclone durante a pré-visualização do AWS Mainframe Modernization, você deve atualizar `m2-rclone.cmd` para a versão mais recente localizada em `C:\Users\Public`.

1. Copie os `m2-rclone.conf` arquivos `m2-rclone.cmd` e fornecidos em `C:\Users\Public` sua pasta pessoal `C:\Users\PhotonUser\My Files\Home Folder` usando o Explorador de Arquivos.
2. Atualize os parâmetros de `m2-rclone.conf` configuração com sua chave de acesso AWS e o segredo correspondente, bem como seu Região da 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. Mo `m2-rclone.cmd`, faça as seguintes alterações:
   - Altere `your-s3-bucket` para o nome do seu bucket do Amazon S3. Por exemplo, `m2-s3-mybucket`.

- Altere `your-s3-folder-key` para sua chave de bucket do Amazon S3. Por exemplo, `myProject`.
- Altere `your-local-folder-path` para o caminho do diretório em que você deseja que os arquivos do aplicativo sejam sincronizados a partir do bucket do Amazon S3 que os contém. Por exemplo, `D:\PhotonUser\My Files\Home Folder\m2-new`. Esse diretório sincronizado deve ser um subdiretório da pasta inicial para que o AppStream 2.0 faça backup e restaure adequadamente no início e no final da sessão.

```
: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. Abra um prompt de comando do Windows, toque em CD, `C:\Users\PhotonUser\My Files\Home Folder` se necessário, e execute`m2-rclone.cmd`. Esse script de comando executa um loop contínuo, sincronizando o bucket e a chave do Amazon S3 com a pasta local a cada 10 segundos. Você pode ajustar o tempo limite conforme necessário. Você deve ver o código-fonte do aplicativo localizado no bucket do Amazon S3 no Windows File Explorer.

Para adicionar novos arquivos ao conjunto em que você está trabalhando ou para atualizar os existentes, faça o upload dos arquivos para o bucket do Amazon S3 e eles serão sincronizados com seu diretório na próxima iteração definida em `m2-rclone.cmd`. Da mesma forma, se você quiser excluir alguns arquivos, exclua-os do bucket do Amazon S3. A próxima operação de sincronização os excluirá do seu diretório local.

## Etapa 3: Criar uma fonte de ODBC para a instância do Amazon RDS

1. Para iniciar a ferramenta EA_Admin, navegue até o menu seletor de aplicativos no canto superior esquerdo da janela do navegador e escolha MF EA_Admin.
2. No menu Administrar, escolha Fontes de dados ODBC e escolha Adicionar na guia DSN do usuário.
3. Na caixa de diálogo Create New Data Source (Criar nova fonte de dados), escolha Simba Athena ODBC Driver (Driver ODBC do Simba Athena) e, em seguida, selecione Finish (Concluir).

4. Na caixa de diálogo Configuração do driver ODBC PostgreSQL Unicode (psqloDBC), defina e anote o nome da fonte de dados que você deseja. Preencha os seguintes parâmetros com os valores da instância do RDS que você criou anteriormente:

   Descrição

   Descrição opcional para ajudá-lo a identificar essa conexão de banco de dados rapidamente.

   Banco de dados

   O banco de dados do Amazon RDS que você criou anteriormente.

   Servidor

   O endpoint do Amazon RDS.

   Porta

   A porta do Amazon RDS.

   Nome de usuário

   Conforme definido na instância do Amazon RDS.

   Senha

   Conforme definido na instância do Amazon RDS.

5. Escolha Testar para validar se a conexão com o Amazon RDS foi bem-sucedida e, em seguida, escolha Salvar para salvar seu novo DSN de usuário.
6. Espere até ver a mensagem que confirma a criação do espaço de trabalho adequado e, em seguida, escolha OK para finalizar com as fontes de dados ODBC e fechar a ferramenta EA_Admin.
7. Navegue novamente até o menu do seletor de aplicativos e escolha Enterprise Analyzer para iniciar a ferramenta. Selecione Create new (Criar novo).
8. Na janela de configuração do espaço de trabalho, insira o nome do espaço de trabalho e defina sua localização. O espaço de trabalho pode ser o disco baseado no Amazon S3, se você trabalhar com essa configuração, ou sua pasta inicial, se preferir.
9. Escolha Escolher outro banco de dados para se conectar à sua instância do Amazon RDS.
10. Escolha o ícone do Postgre nas opções e, depois, escolha OK.

11. Para as configurações do Windows, em Opções — Definir parâmetros de conexão, insira o nome da fonte de dados que você criou. Insira também o nome do banco de dados, o nome do esquema, o nome do usuário e a senha. Escolha OK.
12. Aguarde até que o Enterprise Analyzer crie todas as tabelas, índices, etc. necessários para armazenar os resultados. Essa etapa pode levar alguns minutos. O Enterprise Analyzer confirma quando o banco de dados e o espaço de trabalho estão prontos para uso.
13. Navegue novamente até o menu do seletor de aplicativos e escolha Enterprise Analyzer para iniciar a ferramenta.
14. A janela de inicialização do Enterprise Analyzer aparece no novo local selecionado do espaço de trabalho. Escolha OK.
15. Navegue até seu repositório no painel esquerdo, selecione o nome do repositório e escolha Adicionar arquivos/pastas ao seu espaço de trabalho. Selecione a pasta em que o código do aplicativo está armazenado para adicioná-lo ao espaço de trabalho. Você pode usar o código de exemplo anterior do BankDemo, se quiser. Quando o Enterprise Analyzer solicitar que você verifique esses arquivos, escolha Verificar para iniciar o relatório de verificação inicial do Enterprise Analyzer. A conclusão pode levar alguns minutos, dependendo do tamanho do aplicativo.
16. Expanda seu espaço de trabalho para ver os arquivos e pastas que você adicionou ao espaço de trabalho. Os tipos de objetos e os relatórios de complexidade ciclomática também são visíveis no quadrante superior do painel Visualizador de gráficos.

Agora você pode usar o Enterprise Analyzer para todas as tarefas necessárias.

## Sessões subsequentes

1. Inicie uma sessão com o AppStream 2.0 com a URL que você recebeu na mensagem de e-mail de boas-vindas do AppStream 2.0.
2. Faça login com seu e-mail e senha permanente.
3. Selecione a pilha do Enterprise Analyzer.
4. Inicie `Rclone` para se conectar ao disco suportado pelo Amazon S3 se você usar essa opção para compartilhar os arquivos do espaço de trabalho.
5. Inicie o Enterprise Analyzer para fazer suas tarefas.

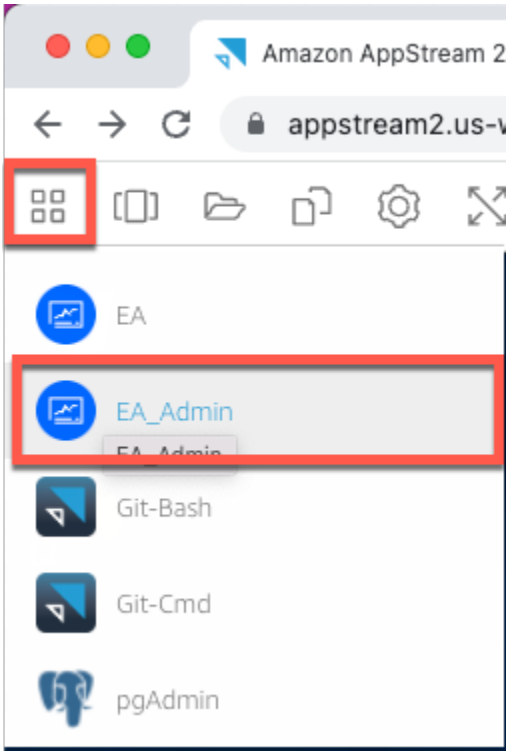
# Solução de problemas de conexão do espaço de trabalho

Ao tentar se reconectar ao seu espaço de trabalho do Enterprise Analyzer, você pode ver um erro como este:

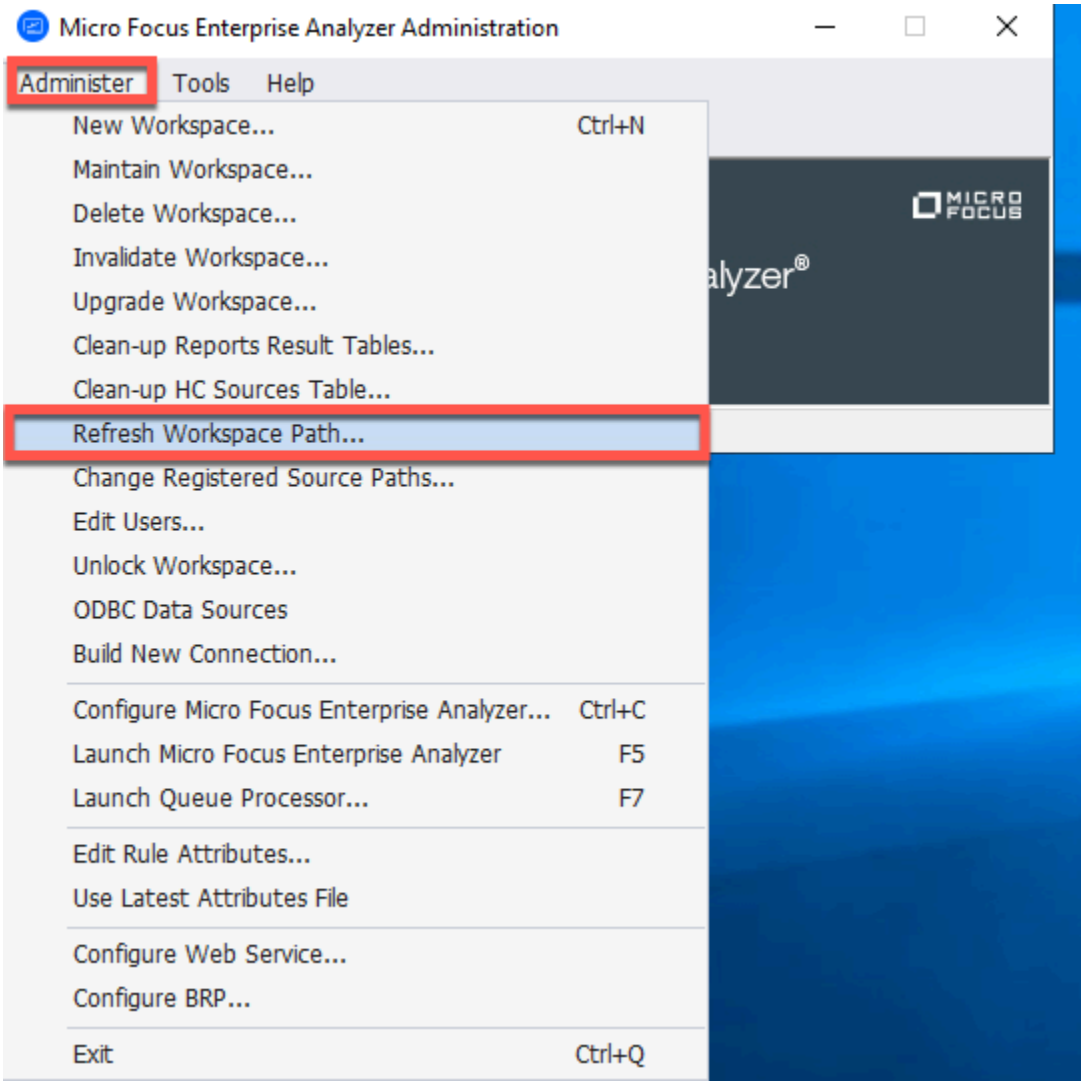
```
Cannot access the workspace directory D:\PhotonUser\My Files\Home Folder\EA_BankDemo.
 The workspace has been created on a non-shared disk of the EC2AMAZ-E6LC33H computer.
 Would you like to correct the workspace directory location?
```

Para resolver esse problema, escolha OK para limpar a mensagem e conclua as etapas a seguir.

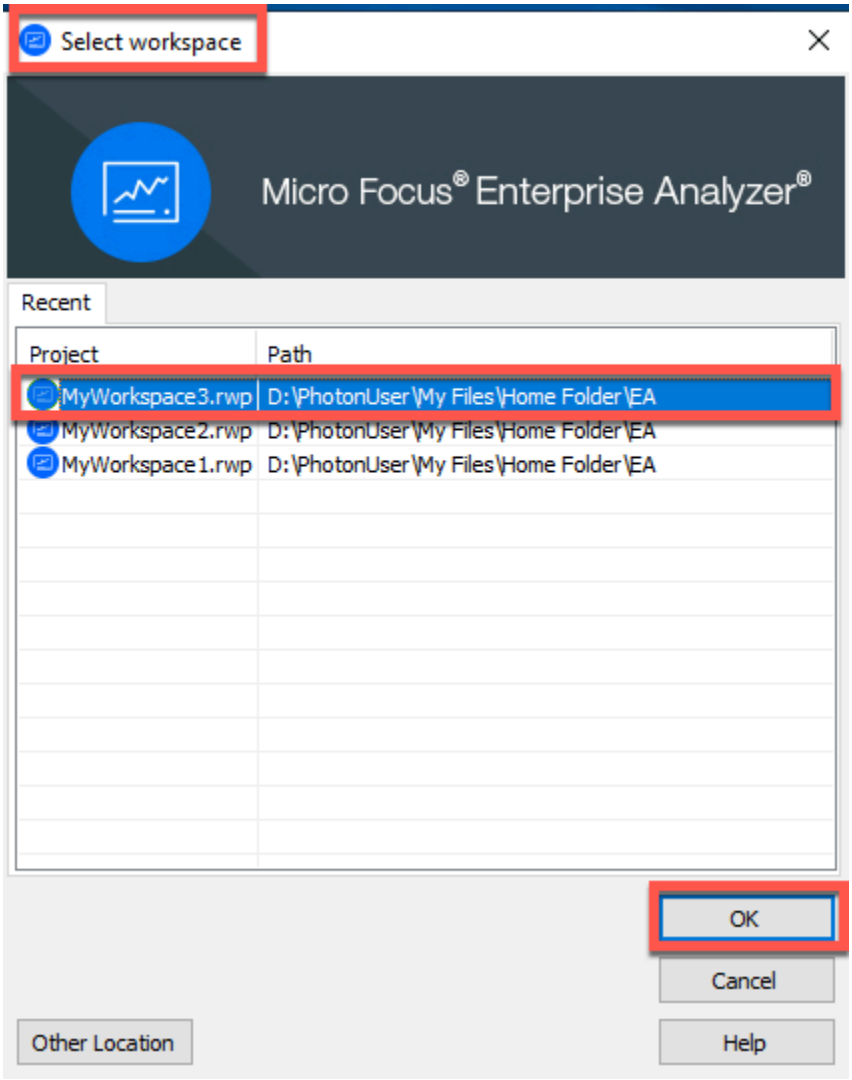
1. No AppStream 2.0, escolha o ícone Iniciar aplicação na barra de ferramentas e, depois, escolha EA_Admin para iniciar a ferramenta de administração do Micro Focus Enterprise Analyzer.



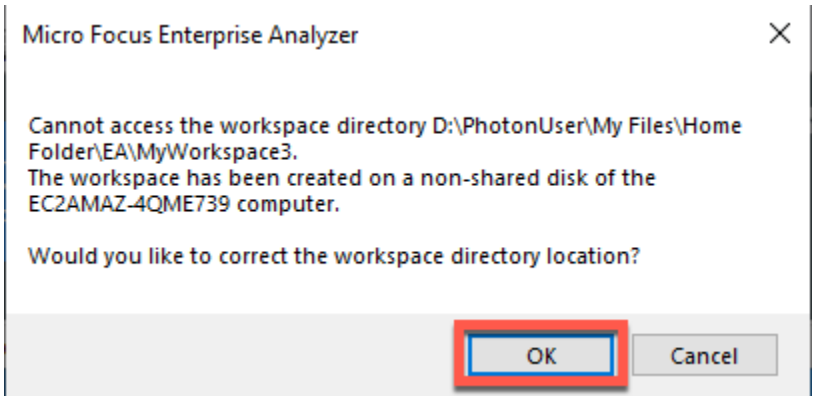

2. No menu Administrar, escolha Atualizar caminho do espaço de trabalho….



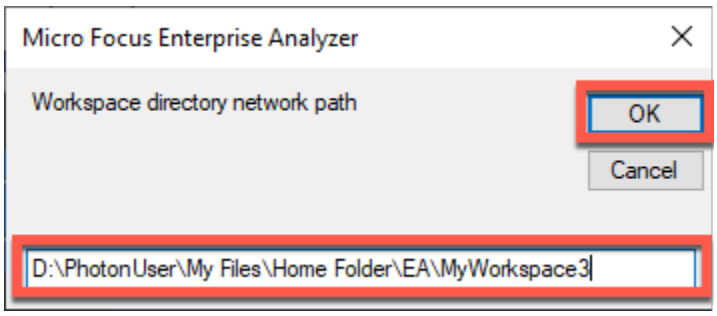

3. Em Selecionar espaço de trabalho, escolha o espaço de trabalho desejado e, em seguida, escolha OK.



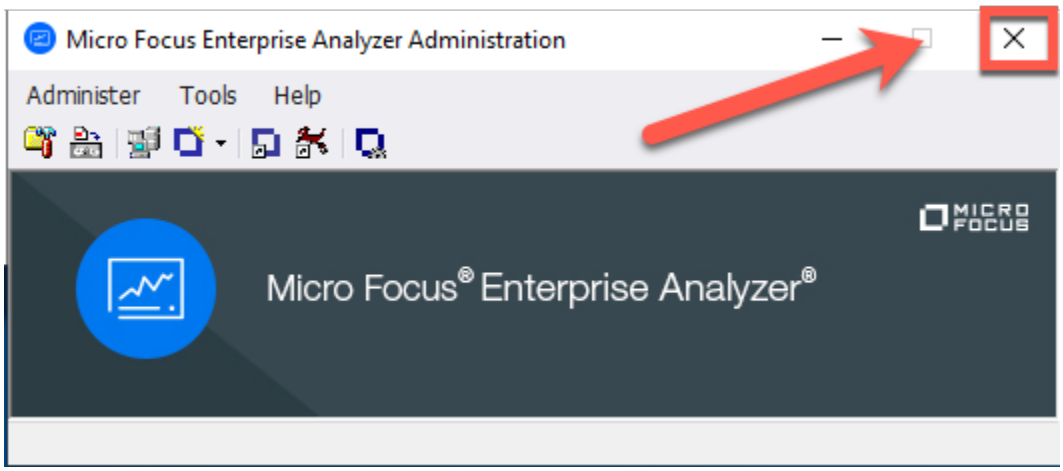

4. Escolha OK para confirmar a mensagem de erro.



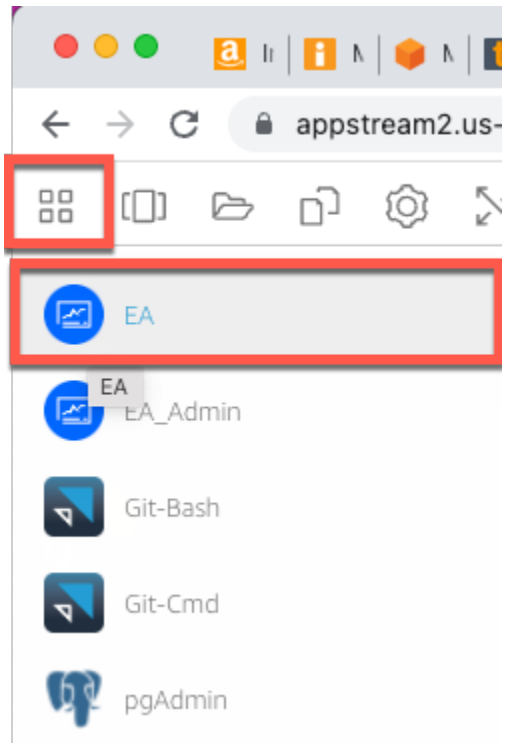

5. Em Caminho de rede do diretório Workspace, insira o caminho correto para seu espaço de trabalho, por exemplo, `D:\PhotonUser\My Files\Home Folder\EA\MyWorkspace3`.




6. Feche a ferramenta de administração do Micro Focus Enterprise Analyzer.




7. No AppStream 2.0, escolha o ícone Iniciar aplicação na barra de ferramentas e, em seguida, escolha EA para iniciar o Micro Focus Enterprise Analyzer.



8. Repita as etapas 3 e 4.

O Micro Focus Enterprise Analyzer agora deve abrir com o espaço de trabalho existente.

## Limpeza de recursos

Se os recursos criados durante este tutorial não forem mais necessários, exclua-os para não incorrer em cobranças adicionais. Execute as etapas a seguir:

- Use a ferramenta EA_Admin para excluir o espaço de trabalho.
- Exclua os buckets do S3 que você criou para este tutorial. Para obter mais informações, consulte Chaves de bucket do Amazon S3 no Guia do usuário do Amazon S3.
- Escolha o banco de dados do RDS que você criou para este tutorial. Para ter mais informações, consulte Excluir uma instância de banco de dados.

# Tutorial: Configurar o Micro Focus Enterprise Developer no AppStream 2.0

Este tutorial descreve como configurar o Micro Focus Enterprise Developer para um ou mais aplicativos de mainframe para mantê-los, compilá-los e testá-los usando os recursos do Enterprise Developer. A configuração é baseada nas imagens Windows do AppStream 2.0 que o AWS Mainframe Modernization compartilha com o cliente e na criação de frotas e pilhas do AppStream 2.0, conforme descrito em Tutorial: Configurar o AppStream 2.0 para uso com o Micro Focus Enterprise Analyzer e o Micro Focus Enterprise Developer.

> **Important**
>
> As etapas deste tutorial pressupõem que você configurou o AppStream 2.0 usando o modelo de AWS CloudFormation cfn-m2-appstream-fleet-ea-ed.yaml que pode ser baixado. Para obter mais informações, consulte Tutorial: Configurar o AppStream 2.0 para uso com o Micro Focus Enterprise Analyzer e o Micro Focus Enterprise Developer.
> Você deve executar as etapas dessa configuração quando a frota e a pilha do Enterprise Developer estiverem em funcionamento.

Para obter uma descrição completa dos recursos e resultados do Enterprise Developer v7, confira sua documentação on-line atualizada (v7.0) no site da Micro Focus.

## Conteúdo da imagem

Além do próprio Enterprise Developer, a imagem contém a imagem que contém Rumba (um emulador TN3270). Ele também contém as seguintes ferramentas e bibliotecas.

Ferramentas de terceiros

- Python
- Rclone
- pgAdmin
- git-scm
- Driver ODBC PostgreSQL

Bibliotecas em `C:\Users\Public`

- Código-fonte do BankDemo e definição do projeto para Enterprise Developer: `m2-bankdemo-template.zip`.
- Pacote de instalação do MFA para o mainframe: `mfa.zip`. Para obter mais informações, consulte Visão geral do acesso ao mainframe na documentação do Micro Focus Enterprise Developer.
- Arquivos de comando e configuração do Rclone (instruções para seu uso nos tutoriais): `m2-rclone.cmd` e `m2-rclone.conf`.

Se você precisar acessar o código-fonte que ainda não está carregado nos repositórios do CodeCommit, mas que está disponível em um bucket do Amazon S3, por exemplo, para realizar o carregamento inicial do código-fonte no git, siga o procedimento para criar um disco virtual do Windows conforme descrito em Tutorial: Configurar o Enterprise Analyzer no AppStream 2.0.

Tópicos

- Pré-requisitos
- Etapa 1: Configuração por usuários individuais do Enterprise Developer
- Etapa 2: criar a pasta virtual baseada no Amazon S3 no Windows (opcional)
- Etapa 3: clonar o repositório
- Sessões subsequentes

- Limpeza de recursos

## Pré-requisitos

- Um ou mais repositórios do CodeCommit carregados com o código-fonte do aplicativo a ser mantido. A configuração do repositório deve corresponder aos requisitos do pipeline de CI/CD acima para criar sinergias por meio da combinação das duas ferramentas.
- Cada usuário deve ter credenciais para o repositório ou repositórios do CodeCommit definidos pelo administrador da conta de acordo com as informações em Autenticação e controle de acesso do AWS CodeCommit. A estrutura dessas credenciais é revisada em Autenticação e controle de acesso do AWS CodeCommit, e a referência completa para autorizações do IAM para o CodeCommit está na referência de permissões do CodeCommit: o administrador pode definir políticas distintas do IAM para funções distintas, ter credenciais específicas para a função de cada repositório e limitar as autorizações do usuário ao conjunto específico de tarefas que ele precisa realizar em um determinado repositório. história. Portanto, para cada mantenedor do repositório do CodeCommit, o administrador da conta gerará um usuário primário e concederá a esse usuário permissões para acessar o repositório ou repositórios necessários selecionando a política ou as políticas adequadas do IAM para acesso ao CodeCommit.

## Etapa 1: Configuração por usuários individuais do Enterprise Developer

1. Obtenha suas credenciais do IAM:
    1. Conecte ao console do AWS em https://console.aws.amazon.com/iam.
    2. Siga o procedimento descrito na etapa 3 de Configuração para usuários HTTPS usando credenciais do Git no Guia do usuárioAWS CodeCommit.
    3. Copie as credenciais de login específicas do CodeCommit que o IAM gerou para você, mostrando, copiando e colando essas informações em um arquivo seguro no seu computador local ou escolhendo Fazer download das credenciais para baixar essas informações como um arquivo .CSV. Você precisa dessas informações para se conectar ao CodeCommit.
2. Inicie uma sessão com o AppStream 2.0 com base na URL recebida no e-mail de boas-vindas. Use seu e-mail como nome de usuário e crie sua senha.
3. Selecione sua pilha de desenvolvedores corporativos.
4. Na página do menu, escolha Desktop para acessar a área de trabalho do Windows transmitida pela frota.

# Etapa 2: criar a pasta virtual baseada no Amazon S3 no Windows (opcional)

Se houver necessidade do Rclone (veja acima), crie a pasta virtual baseada no Amazon S3 no Windows: (opcional se todos os artefatos do aplicativo vierem exclusivamente do acesso ao CodeCommit).

> **Note**
>
> Se você já usou o Rclone durante a pré-visualização do AWS Mainframe Modernization, você deve atualizar `m2-rclone.cmd` para a versão mais recente localizada em `C:\Users\Public`.

1. Copie os `m2-rclone.conf` arquivos `m2-rclone.cmd` e fornecidos em `C:\Users\Public` sua pasta pessoal `C:\Users\PhotonUser\My Files\Home Folder` usando o Explorador de Arquivos.
2. Atualize os parâmetros de `m2-rclone.conf` configuração com sua chave de acesso AWS e o segredo correspondente, bem como seu Região da 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. Mo `m2-rclone.cmd`, faça as seguintes alterações:

   - Altere `your-s3-bucket` para o nome do seu bucket do Amazon S3. Por exemplo, `m2-s3-mybucket`.
   - Altere `your-s3-folder-key` para sua chave de bucket do Amazon S3. Por exemplo, `myProject`.
   - Altere `your-local-folder-path` para o caminho do diretório em que você deseja que os arquivos do aplicativo sejam sincronizados a partir do bucket do Amazon S3 que os contém. Por exemplo, `D:\PhotonUser\My Files\Home Folder\m2-new`. Esse diretório

sincronizado deve ser um subdiretório da pasta inicial para que o AppStream 2.0 faça backup e restaure adequadamente no início e no final da sessão.

```
: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. Abra um prompt de comando do Windows, toque em CD, `C:\Users\PhotonUser\My Files \Home Folder` se necessário, e execute`m2-rclone.cmd`. Esse script de comando executa um loop contínuo, sincronizando o bucket e a chave do Amazon S3 com a pasta local a cada 10 segundos. Você pode ajustar o tempo limite conforme necessário. Você deve ver o código-fonte do aplicativo localizado no bucket do Amazon S3 no Windows File Explorer.

Para adicionar novos arquivos ao conjunto em que você está trabalhando ou para atualizar os existentes, faça o upload dos arquivos para o bucket do Amazon S3 e eles serão sincronizados com seu diretório na próxima iteração definida em `m2-rclone.cmd`. Da mesma forma, se você quiser excluir alguns arquivos, exclua-os do bucket do Amazon S3. A próxima operação de sincronização os excluirá do seu diretório local.

## Etapa 3: clonar o repositório

1. Navegue até o menu seletor de aplicativos no canto superior esquerdo da janela do navegador e selecione Enterprise Developer.
2. Conclua a criação do espaço de trabalho exigido pelo Enterprise Developer em sua pasta inicial escolhendo `C:\Users\PhotonUser\My Files\Home Folder` (aka `D: \PhotonUser\My Files\Home Folder`) como local para o espaço de trabalho.
3. No Enterprise Developer, clone seu repositório do CodeCommit acessando o Project Explorer, clique com o botão direito do mouse e escolha Importar, Importar..., Git, Projetos do URI do Git Clone. Em seguida, insira suas credenciais de login específicas do CodeCommit e preencha a caixa de diálogo do Eclipse para importar o código.

O repositório git do CodeCommit agora está clonado em seu espaço de trabalho local.

Seu espaço de trabalho do Enterprise Developer agora está pronto para iniciar o trabalho de manutenção em seu aplicativo. Em particular, você pode usar a instância local do Microfocus Enterprise Server (ES) integrada ao Enterprise Developer para depurar e executar interativamente seu aplicativo para validar suas alterações localmente.

> **Note**
>
> O ambiente local do Enterprise Developer, incluindo a instância local do Enterprise Server, é executado no Windows, enquanto o AWS Mainframe Modernization é executada no Linux. Recomendamos que você execute testes complementares no ambiente Linux fornecido pelo AWS Mainframe Modernization depois de confirmar o novo aplicativo no CodeCommit e reconstruí-lo para esse destino e antes de implantar o novo aplicativo em produção.

## Sessões subsequentes

Ao selecionar uma pasta que está sob gerenciamento do AppStream 2.0, como a pasta inicial para a clonagem do seu repositório do CodeCommit, ela será salva e restaurada de forma transparente em todas as sessões. Conclua as seguintes etapas na próxima vez que precisar trabalhar com o aplicativo:

1. Inicie uma sessão com o AppStream 2.0 com base na URL recebida no e-mail de boas-vindas.
2. Faça login com seu e-mail e senha permanente.
3. Selecione a pilha Enterprise Developer.
4. Inicie `Rclone` para se conectar (veja acima) ao disco baseado no Amazon S3 quando essa opção for usada para compartilhar os arquivos do espaço de trabalho.
5. Inicie o Enterprise Developer para fazer seu trabalho.

## Limpeza de recursos

Se os recursos criados durante este tutorial não forem mais necessários, exclua-os para que você não continue sendo cobrado por eles. Execute as etapas a seguir:

- Exclua o repositório do CodeCommit que você criou para este tutorial. Para obter mais informações, consulte Excluir um repositório do CodeCommit no Guia do usuário AWS CodeCommit.

- Exclua o banco de dados que você criou para este tutorial. Para ter mais informações, consulte Excluir uma instância de banco de dados.

# Configure a automação para sessões de streaming do Micro Focus Enterprise Analyzer e do Micro Focus Enterprise Developer

Você pode executar automaticamente um script no início e no final da sessão para permitir uma automação específica para o contexto do seu cliente. Para obter mais informações sobre esse recurso do AppStream 2.0, consulte Usar scripts de sessão para gerenciar a experiência de streaming dos usuários do AppStream 2.0 no Guia de administração do Amazon AppStream 2.0.

Esse recurso exige que você tenha pelo menos as seguintes versões das imagens do Enterprise Analyzer e do Enterprise Developer:

- `m2-enterprise-analyzer-v8.0.4.R1`
- `m2-enterprise-developer-v8.0.4.R1`

Tópicos

- Configure a automação no início da sessão
- Configure a automação no final da sessão

## Configure a automação no início da sessão

Se você quiser executar um script de automação quando os usuários se conectarem ao AppStream 2.0, crie seu script e dê um nome `m2-user-setup.cmd` a ele. Armazene o script da AppStream 2.0 na pasta base para o usuário. As imagens do AppStream 2.0 fornecidas pelo AWS Mainframe Modernization procuram um script com esse nome nesse local e o executam se ele existir.

> **Note**
>
> A duração do script não pode exceder o limite definido pelo AppStream 2.0, que atualmente é de 60 segundos. Para obter mais informações, consulte Executar scripts antes do início das sessões de streaming no Guia de administração do Amazon AppStream 2.0.

## Configure a automação no final da sessão

Se quiser executar um script de automação quando os usuários se desconectarem do AppStream 2.0, crie o script e nomeie-o `m2-user-teardown.cmd`. Armazene o script da AppStream 2.0 na pasta base para o usuário. As imagens do AppStream 2.0 fornecidas pelo AWS Mainframe Modernization procuram um script com esse nome nesse local e o executam se ele existir.

> **Note**
>
> A duração do script não pode exceder o limite definido pelo AppStream 2.0, que atualmente é de 60 segundos. Para obter mais informações, consulte Executar scripts após o término das sessões de streaming no Guia de administração do Amazon AppStream 2.0.

# Exibir conjuntos de dados como tabelas e colunas no Enterprise Developer

Você pode acessar conjuntos de dados de mainframe implantados no AWS Mainframe Modernization usando o tempo de execução do Micro Focus. Você pode visualizar os conjuntos de dados migrados como tabelas e colunas de uma instância do Micro Focus Enterprise Developer. A visualização de conjuntos de dados dessa forma possibilita que você:

- Execute operações `SQL SELECT` nos arquivos de dados migrados.
- Exponha dados fora do aplicativo de mainframe migrado sem alterar o aplicativo.
- Filtre dados com facilidade e salve como CSV ou outros formatos de arquivo.

> **Note**
>
> As etapas 1 e 2 são atividades únicas. Repita as etapas 3 e 4 para cada conjunto de dados para criar as visualizações de banco de dados.

Tópicos

- Etapa 2: Criar o arquivo MFDBFH.cfg
- Etapa 3: Crie um arquivo de estrutura (STR) para o layout do seu caderno
- Etapa 4: Criar a visualização de banco de dados usando o arquivo de estrutura (STR)
- Etapa 5: Exibir conjuntos de dados da Micro Focus como tabelas e colunas

## Pré-requisitos

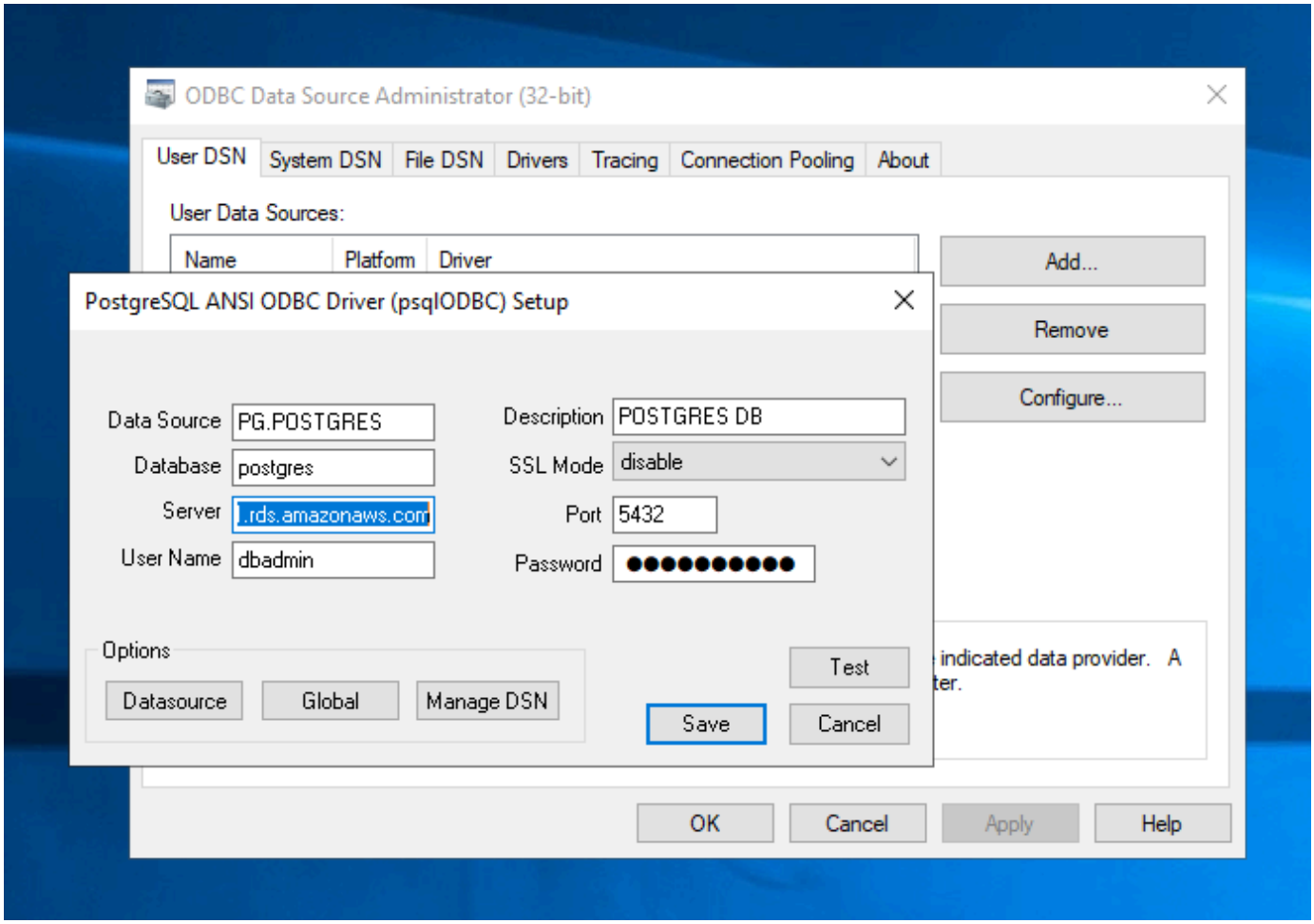
- Você deve ter acesso ao Micro Focus Enterprise Developer Desktop via AppStream 2.0.
- Você deve ter um aplicativo implantado e em execução no AWS Mainframe Modernization usando o mecanismo de tempo de execução da Micro Focus.
- Você está armazenando dados de aplicativos na edição compatível com o Aurora PostgreSQL.

## Etapa 1: Configurar a conexão ODBC com o armazenamento de dados Micro Focus (banco de dados Amazon RDS)

Nesta etapa, você configura uma conexão ODBC com o banco de dados que contém os dados que você deseja visualizar como tabelas e colunas. Esta é uma etapa única.

1. Faça login no Micro Focus Enterprise Developer Desktop usando o URL de streaming do AppStream 2.0.
2. Abra o ODBC Data Source Administrator, escolha DSN do usuário e, em seguida, escolha Adicionar.
3. Em Criar nova fonte de dados, escolha PostgreSQL ANSI e, em seguida, escolha Concluir.
4. Crie uma fonte de dados `PG.POSTGRES` fornecendo as informações necessárias do banco de dados, da seguinte forma:

```
Data Source : PG.POSTGRES
Database    : postgres
Server      : rds_endpoint.rds.amazonaws.com
Port        : 5432
User Name   : user_name
Password    : user_password
```




5. Escolha Testar para garantir que a conexão funcione. Você deverá ver a mensagem `Connection successful` se o teste for bem-sucedido.

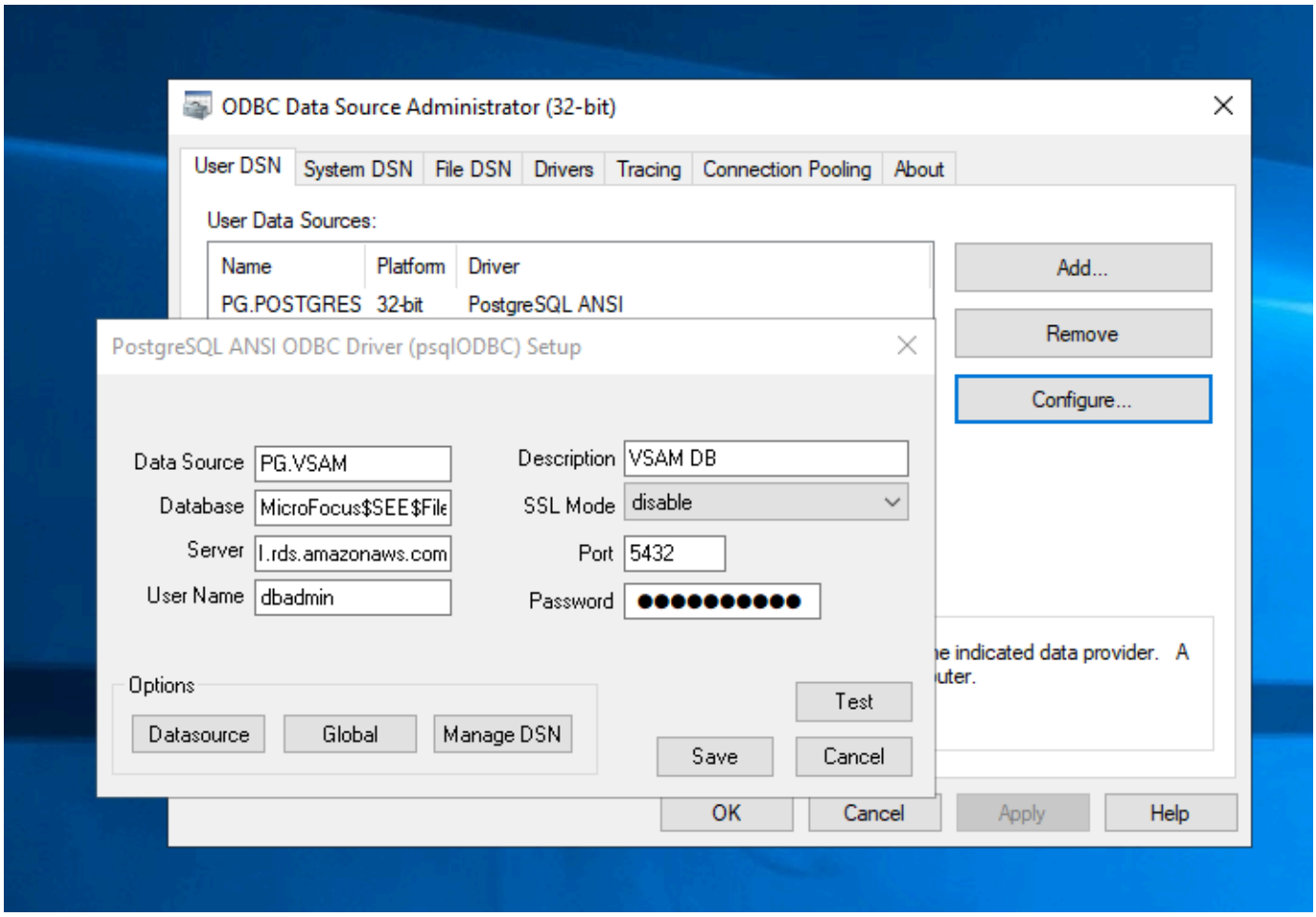
   Se o teste não for bem-sucedido, analise as informações a seguir.

   - Solução de problemas para o Amazon RDS
   - Como resolvo problemas ao me conectar à minha instância de banco de dados Amazon RDS?

6. Salve a fonte de dados.
7. Crie uma fonte de dados para `PG.VSAM`, teste a conexão e salve a fonte de dados. Forneça as seguintes informações para seu banco de dados:

```
Data Source : PG.VSAM
Database    : MicroFocus$SEE$Files$VSAM
Server      : rds_endpoint.rds.amazonaws.com
Port        : 5432
User Name   : user_name
```

```
Password    : user_password
```




## Etapa 2: Criar o arquivo MFDBFH.cfg

Nesta etapa, crie um arquivo de configuração que descreve o armazenamento de dados da Micro Focus. Esta é uma etapa única de configuração.

1. Na sua pasta pessoal, por exemplo, em `D:\PhotonUser\My Files\Home Folder\MFED \cfg\MFDBFH.cfg`, crie o arquivo MFDBFH.cfg com o conteúdo a seguir.

```
<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. Verifique a configuração do MFDBFH executando os seguintes comandos para consultar o armazenamento de dados da 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
```

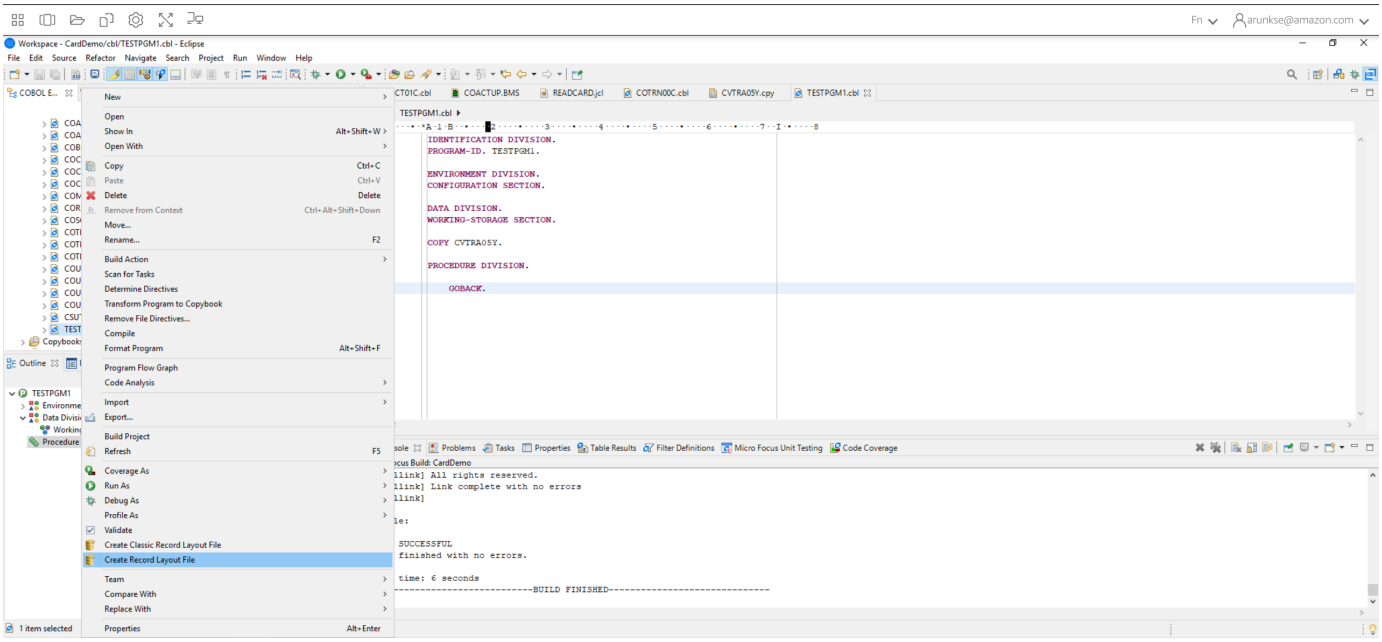
## Etapa 3: Crie um arquivo de estrutura (STR) para o layout do seu caderno

Nesta etapa, você cria um arquivo de estrutura para o layout do seu caderno para poder usá-lo posteriormente para criar visualizações de banco de dados a partir dos conjuntos de dados.

1. Compile o programa associado ao seu caderno. Se nenhum programa estiver usando o caderno, crie e compile um programa simples como o seguinte com uma instrução COPY para seu caderno.

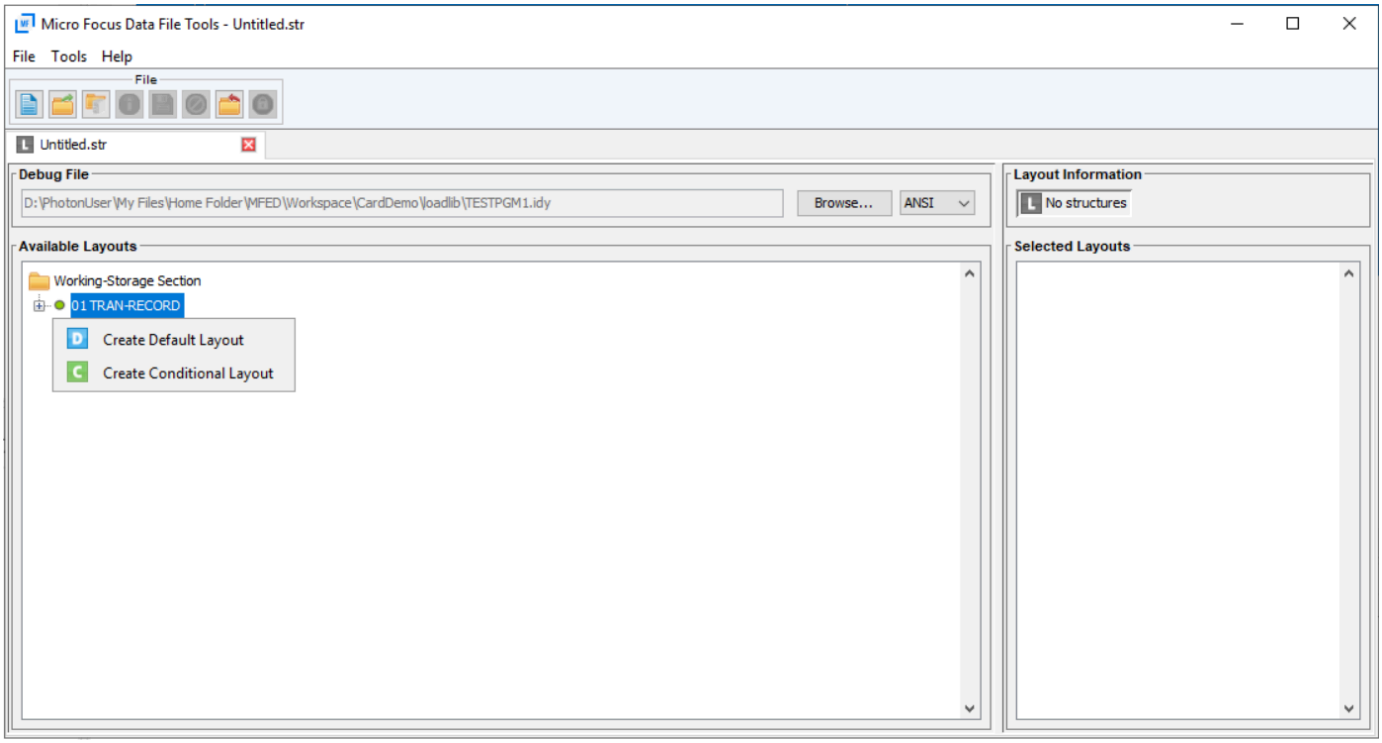
```
IDENTIFICATION DIVISION.
      PROGRAM-ID. TESTPGM1.

      ENVIRONMENT DIVISION.
      CONFIGURATION SECTION.

      DATA DIVISION.
      WORKING-STORAGE SECTION.

      COPY CVTRA05Y.

      PROCEDURE DIVISION.

      GOBACK.
```

2. Após a compilação bem-sucedida, clique com o botão direito do mouse no programa e escolha Criar arquivo de layout de registro. Isso abrirá as Ferramentas de Arquivo de Dados da Micro Focus usando o arquivo.idy gerado durante a compilação.

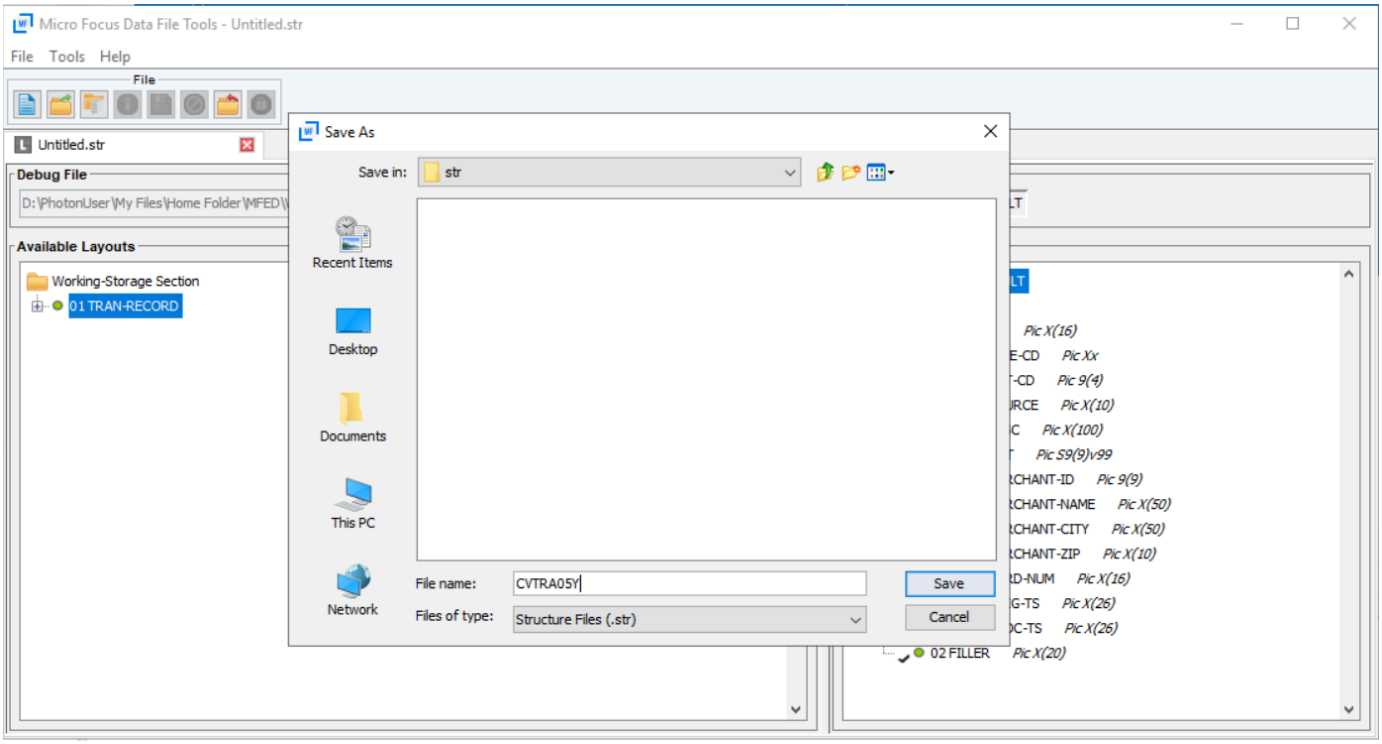



3. Clique com o botão direito na estrutura de registro e escolha Criar layout padrão (estrutura única) ou Criar layout condicional (estrutura múltipla), dependendo do layout.

   Para obter mais informações, consulte Criação de arquivos e layouts de estrutura na documentação da Micro Focus.



4. Depois de criar o layout, escolha Arquivo no menu e escolha Salvar como. Navegue e salve o arquivo em sua pasta pessoal com o mesmo nome de arquivo do seu caderno. Você pode escolher criar uma pasta chamada `str` e salvar todos os seus arquivos de estrutura lá.




## Etapa 4: Criar a visualização de banco de dados usando o arquivo de estrutura (STR)

Nesta etapa, você usa o arquivo de estrutura criado anteriormente para criar uma visualização do banco de dados para um conjunto de dados.

- Use o comando `dbfhview` para criar uma visualização do banco de dados para um conjunto de dados que já está no armazenamento de dados da Micro Focus, conforme mostrado no exemplo a seguir.

```
##
      ## The below command creates database view for VSAM file
 AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS
      ## using the STR file CVTRA05Y.str
      ##
```

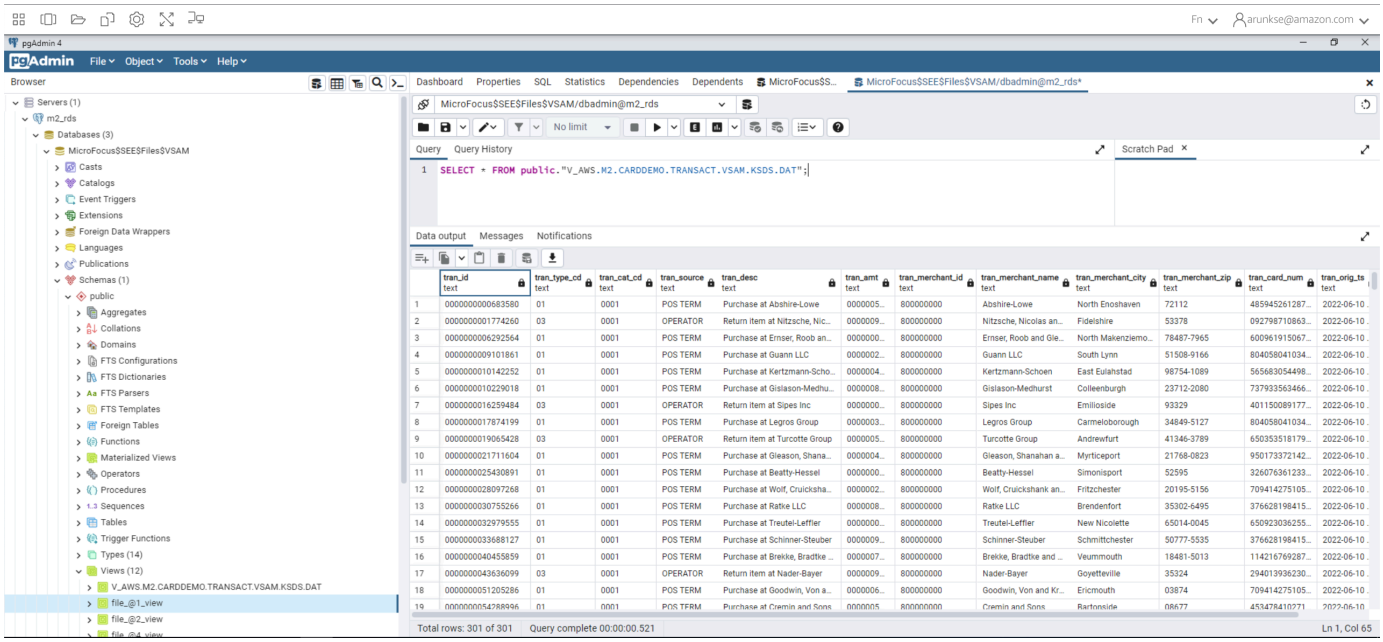
```
      dbfhview -create -struct:"D:\PhotonUser\My Files\Home Folder\MFED\str
\CVTRA05Y.str" -name:V_AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS.DAT -file:sql://
ESPACDatabase/VSAM/AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS.DAT?folder=/DATA

      ##
      ## 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
```

## Etapa 5: Exibir conjuntos de dados da Micro Focus como tabelas e colunas

Nesta etapa, conecte-se ao banco de dados usando pgAdmin para que você possa executar consultas para visualizar os conjuntos de dados, como tabelas e colunas.

- Conecte-se ao banco de dados `MicroFocus$SEE$Files$VSAM` usando o pGADmin e consulte a visualização do banco de dados que você criou na etapa 4.

```
SELECT * FROM public."V_AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS.DAT";
```




# Tutorial: Use modelos com o Micro Focus Enterprise Developer

Este tutorial descreve como usar modelos e projetos predefinidos com o Micro Focus Enterprise Developer. Ele abrange três casos de uso. Todos os casos de uso usam o código de amostra fornecido na amostra do BankDemo. Para baixar a amostra, escolha `bankdemo.zip`.

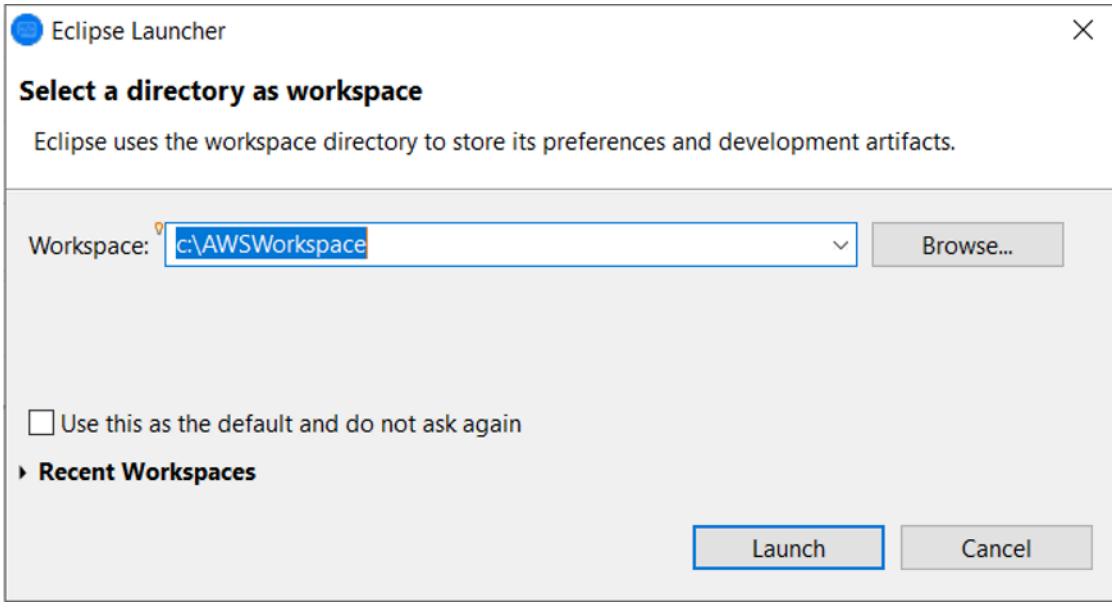
> **Important**
>
> Se você usar a versão do Enterprise Developer para Windows, os binários gerados pelo compilador poderão ser executados somente no Enterprise Server fornecido com o Enterprise Developer. Você não pode executá-los no tempo de execução do AWS Mainframe Modernization, que é baseado no Linux.

Tópicos

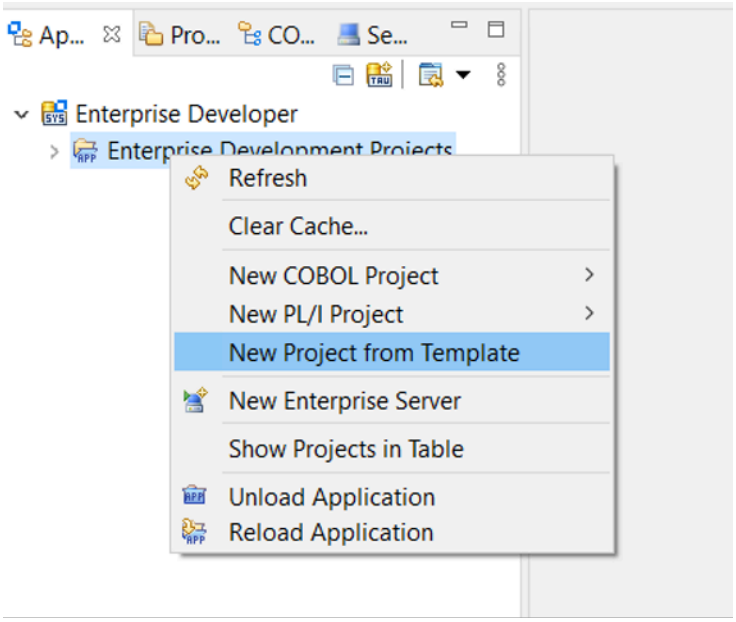
# Caso de uso 1 - Usando o modelo de projeto COBOL contendo componentes de origem

Esse caso de uso exige que você copie os componentes de origem na estrutura de diretórios do modelo como parte das etapas de pré-configuração da demonstração. No, `bankdemo.zip`isso foi alterado em relação à `AWSTemplates.zip` entrega original para evitar duas cópias da fonte.

1. Inicie o Enterprise Developer e especifique o espaço de trabalho escolhido.

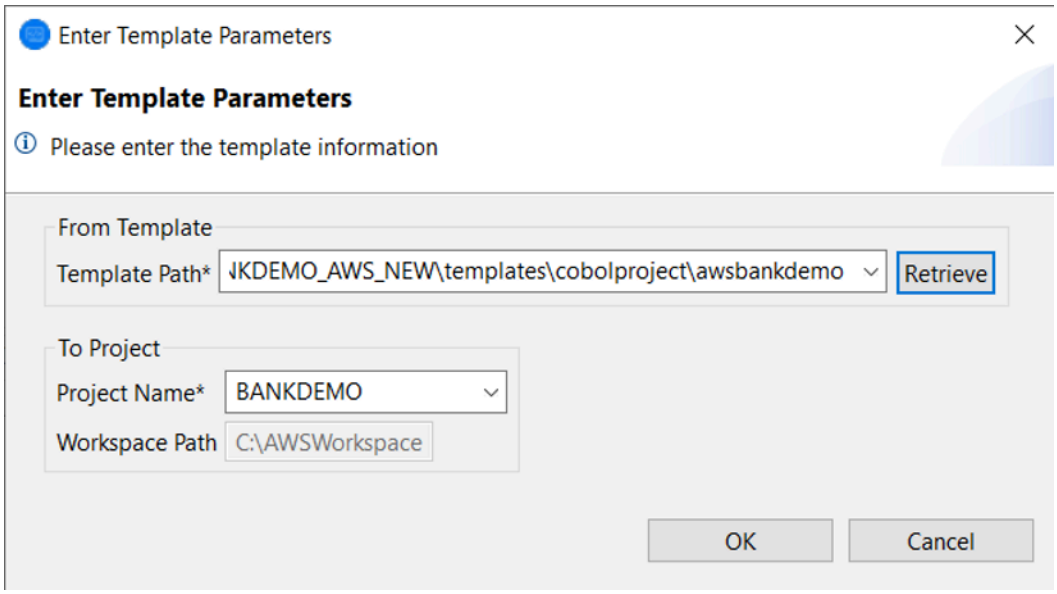



2. Na visualização do Explorador de Aplicativos, no item de visualização em árvore do Enterprise Development Project, escolha Novo projeto a partir do modelo no menu de contexto.

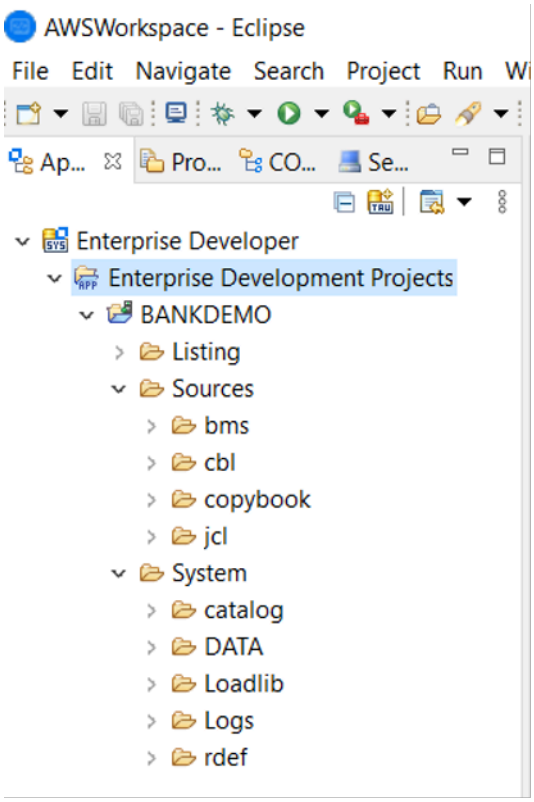


3. Insira os parâmetros do modelo conforme indicado.

> **Note**
>
> O caminho do modelo se referirá ao local onde o ZIP foi extraído.




4. Escolher OK criará um projeto Eclipse de desenvolvimento local com base no modelo fornecido, com uma estrutura completa do ambiente de origem e execução.




A estrutura `System` contém um arquivo completo de definição de recursos com as entradas necessárias para o BANKDEMO, o catálogo necessário com entradas adicionadas e os arquivos de dados ASCII correspondentes.

Como a estrutura do modelo de origem contém todos os itens de origem, esses arquivos são copiados para o projeto local e, portanto, são criados automaticamente no Enterprise Developer.

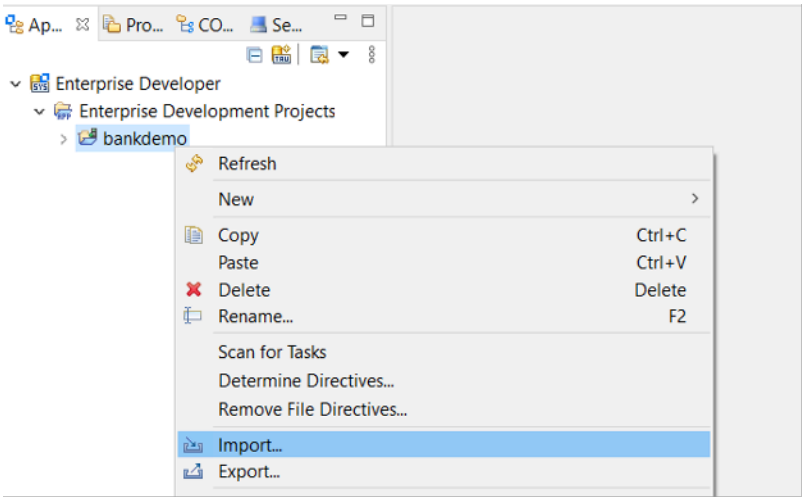
## Caso de uso 2 - Usando o modelo de projeto COBOL sem componentes de origem

As etapas 1 a 3 são idênticas Caso de uso 1 - Usando o modelo de projeto COBOL contendo componentes de origem a.

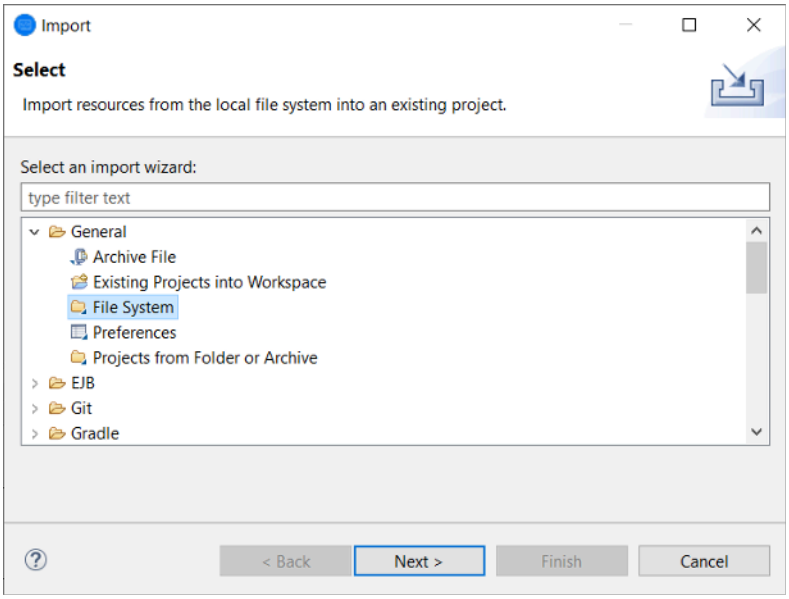
A estrutura `System` nesse caso de uso também contém um arquivo completo de definição de recursos com as entradas necessárias para o BankDemo, o catálogo necessário com entradas adicionadas e os arquivos de dados ASCII correspondentes.

No entanto, a estrutura de origem do modelo não contém nenhum componente. Você deve importá-los para o projeto a partir de qualquer repositório de origem que estiver usando.

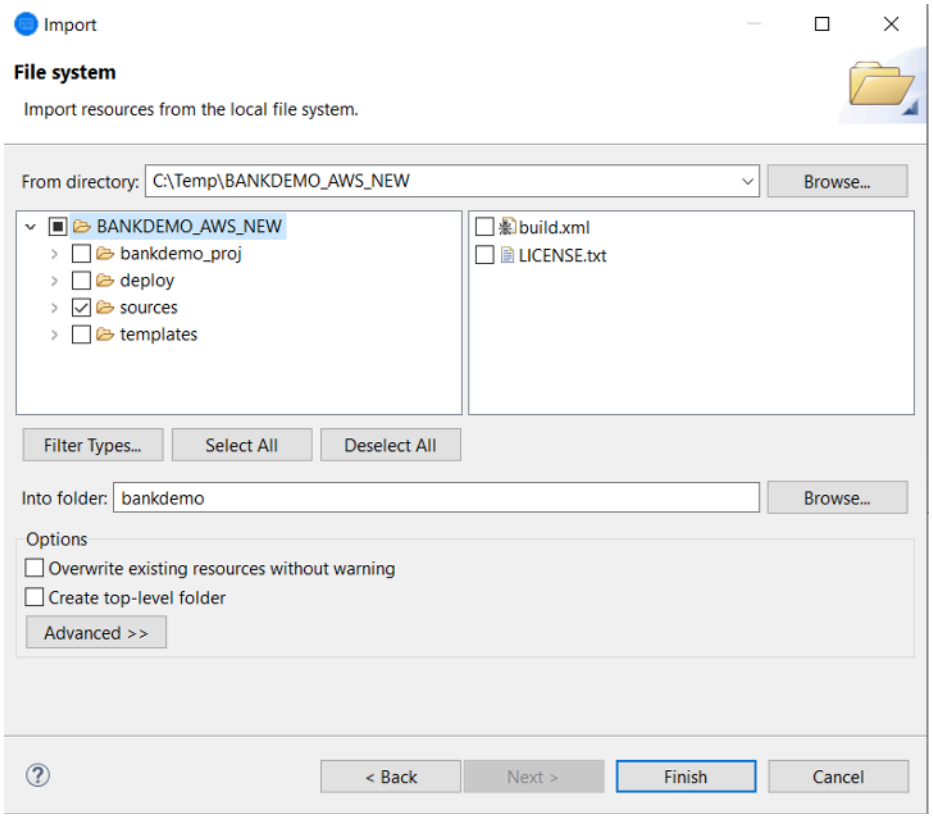
1. Escolha o nome do projeto. No menu de contexto relacionado, escolha Importar.




2. Na caixa de diálogo resultante, na seção Geral, escolha Sistema de arquivos e, em seguida, escolha Próximo.



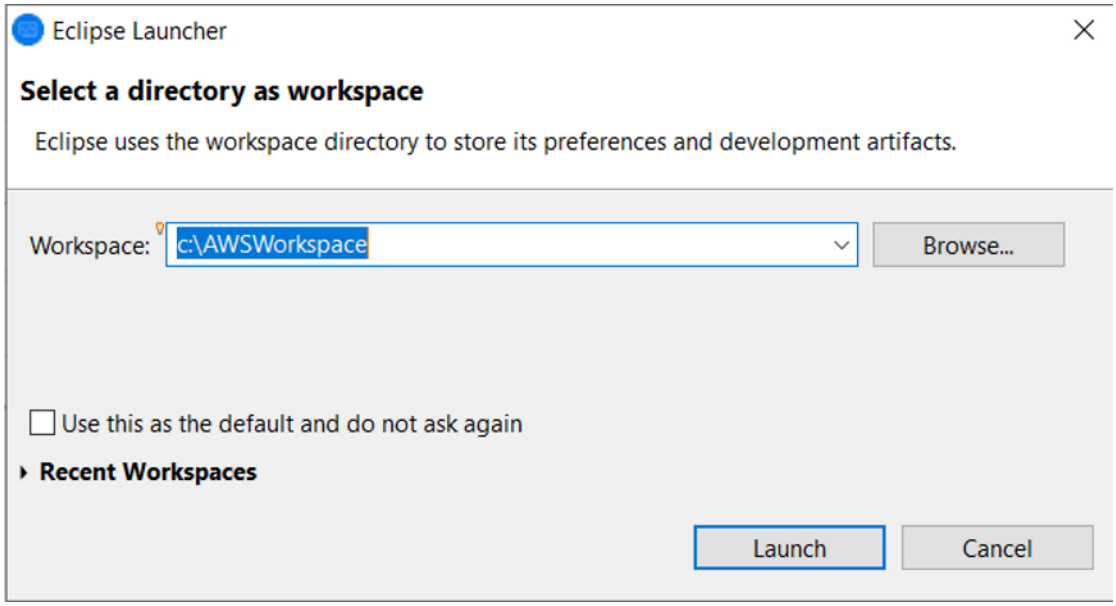

3. Preencha o campo Do diretório navegando no sistema de arquivos para apontar para a pasta do repositório. Escolha todas as pastas que você deseja importar, como `sources`. O `Into folder` campo será preenchido automaticamente. Escolha Concluir.



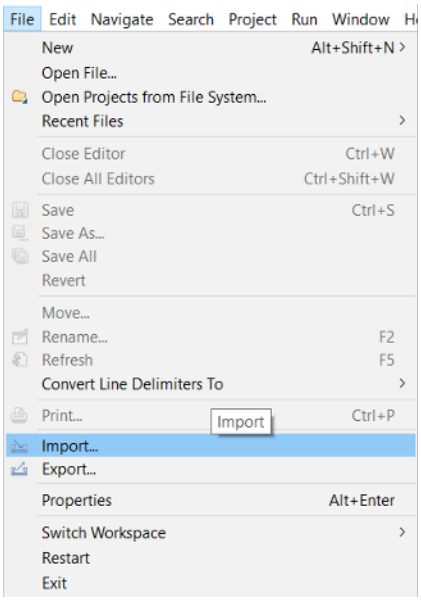

Depois que a estrutura do modelo de origem contém todos os itens de origem, eles são criados automaticamente no Enterprise Developer.

## Caso de uso 3 - Usando o projeto COBOL predefinido vinculado às pastas de origem

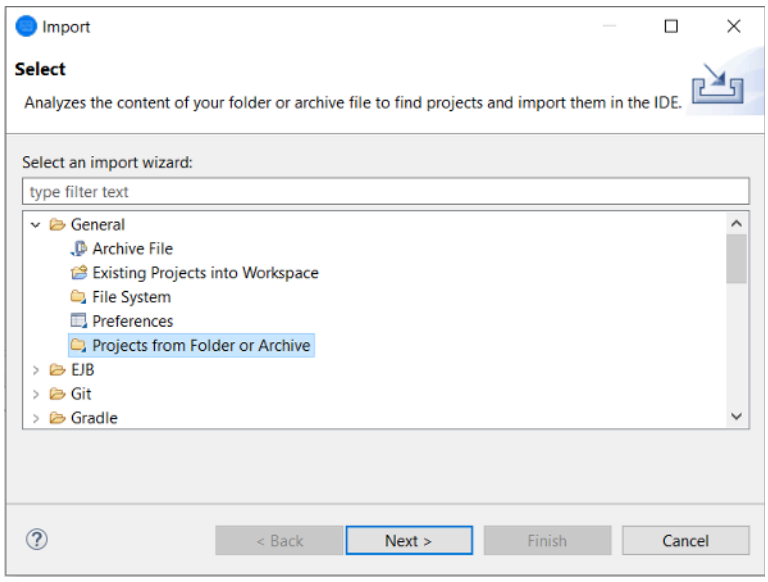
1. Inicie o Enterprise Developer e especifique o espaço de trabalho escolhido.



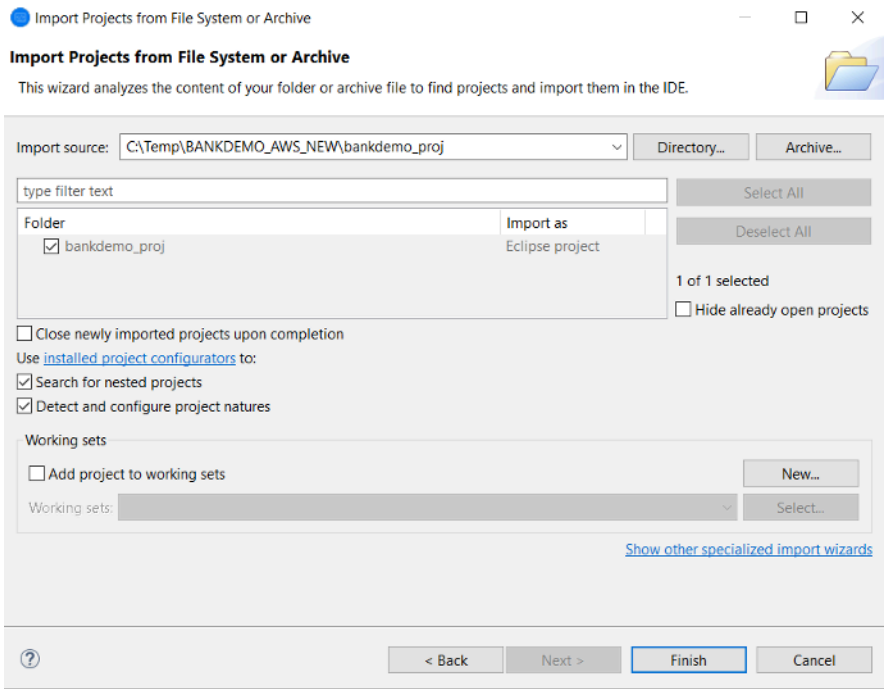

2. No menu File (Arquivo), escolha Import (Importar).




3. Na caixa de diálogo resultante, em Geral, escolha Projetos da pasta ou do arquivo e escolha Próximo.




4. Preencha a fonte de importação, escolha o diretório e navegue pelo sistema de arquivos para selecionar a pasta predefinida do projeto. O projeto contido nele tem links para as pastas de origem no mesmo repositório.

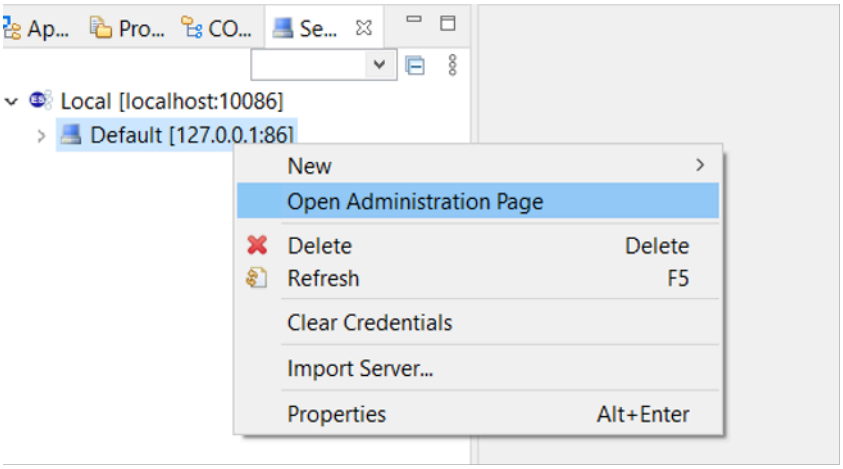



Escolha Concluir.

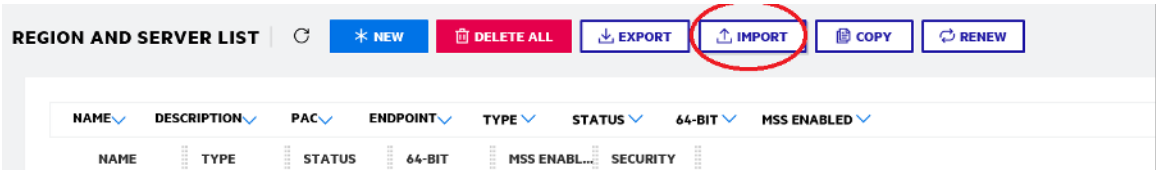
Como o projeto é preenchido pelos links para a pasta de origem, o código é criado automaticamente.

# Usando o modelo JSON de definição de região

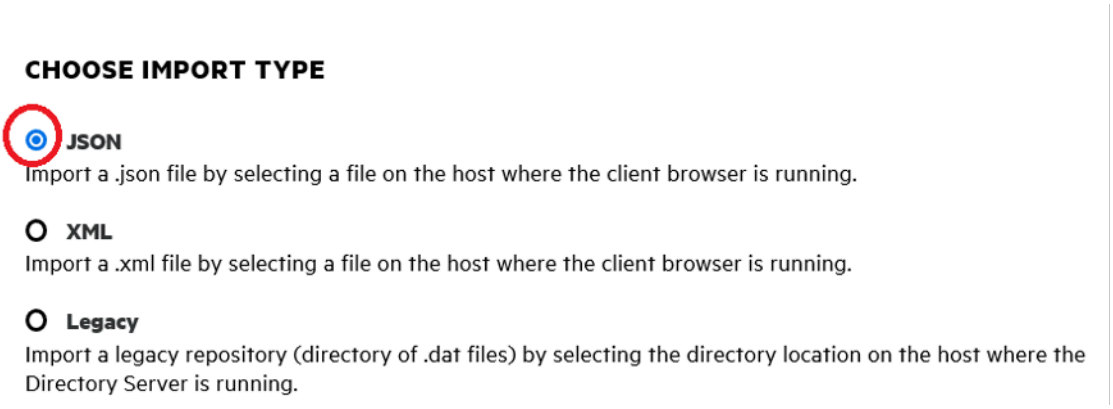
1. Alterne para a visualização Server Explorer. No menu de contexto relacionado, escolha Abrir página de administração, que inicia o navegador padrão.



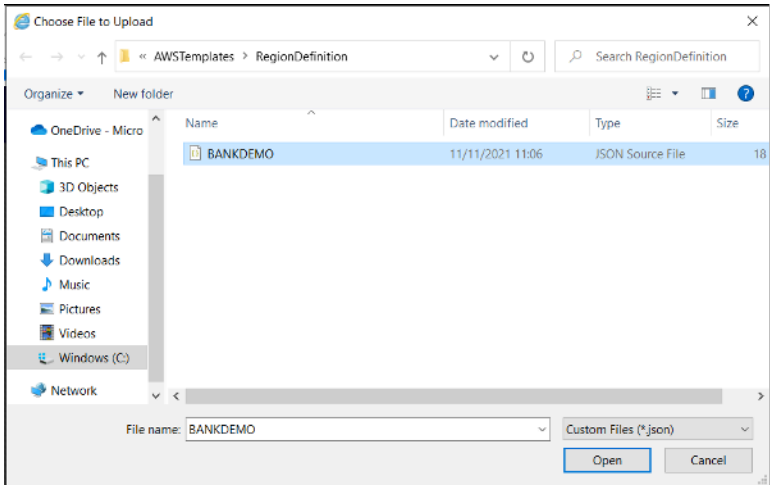

2. Na tela resultante do Enterprise Server Common Web Administration (ESCWA), escolha Importar.



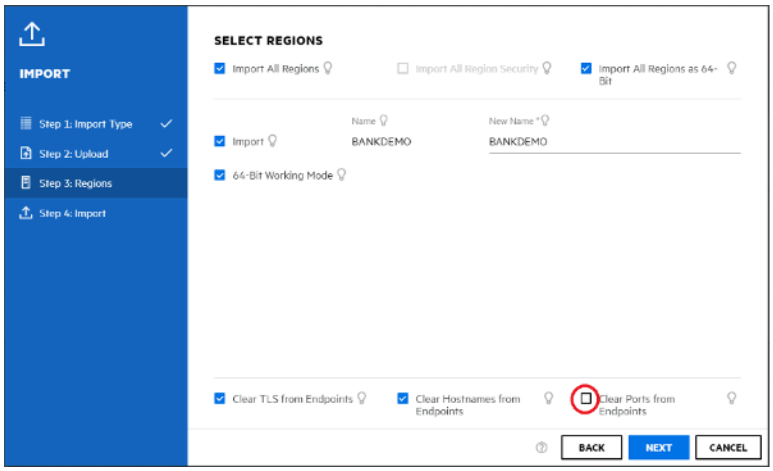

3. Escolha o tipo de importação JSON e escolha Avançar.




4. Faça o upload do `BANKDEMO.JSON` arquivo fornecido.

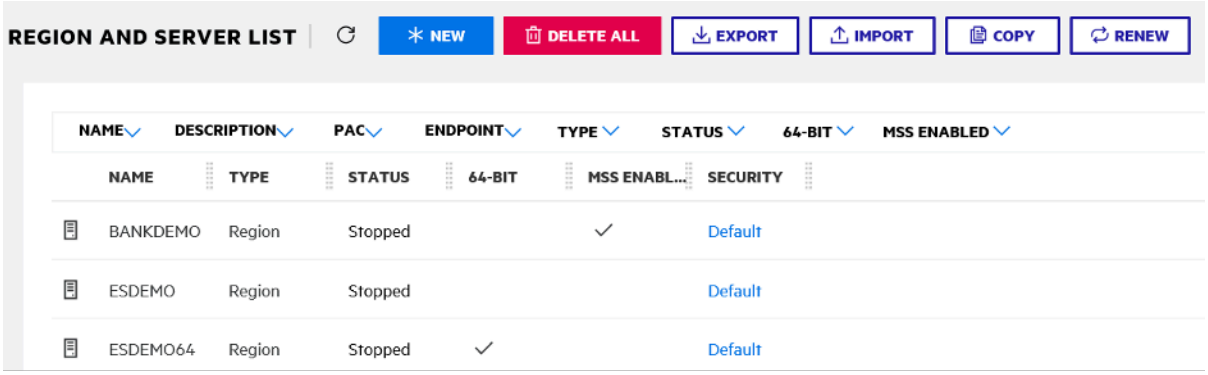



Depois de selecionado, escolha Avançar.

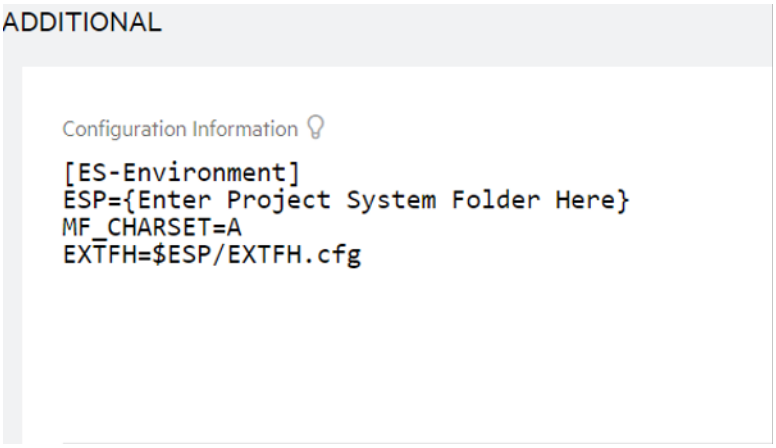



No painel Selecionar regiões, certifique-se de que a opção Limpar portas dos endpoints não esteja selecionada e continue escolhendo Avançar nos painéis até que o painel Executar importação seja exibido. Selecione Import (Importar).

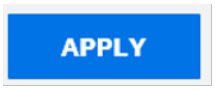

Por fim, clique em Concluir. A região BANKDEMO será então adicionada à lista de servidores.




5. Navegue até as Propriedades Gerais da região BANKDEMO.
6. Role até a seção Configuration (Configuração).
7. A variável de ambiente ESP precisa ser definida na `System` pasta relevante para o Projeto Eclipse criado nas etapas anteriores. O deve ser .

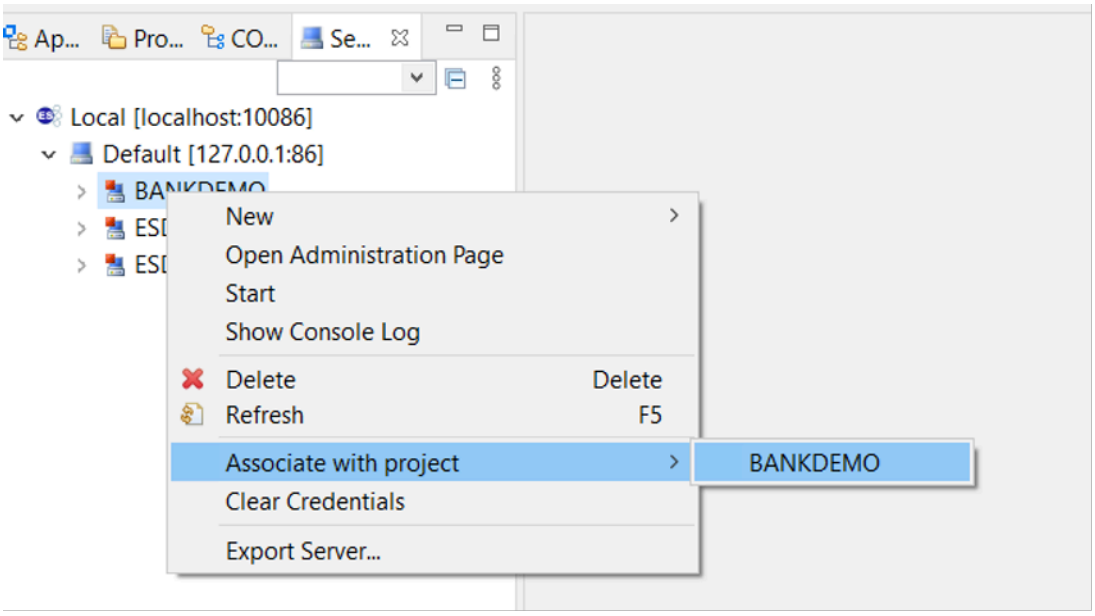



8. Clique em Apply (Aplicar).




A região agora está totalmente configurada para ser executada em conjunto com o projeto Eclipse COBOL.

9. Por fim, de volta ao Enterprise Developer, associe a região importada ao projeto.




O ambiente Enterprise Developer agora está pronto para uso, com uma versão funcional completa do BankDemo. Você pode editar, compilar e depurar o código na região.

> **Important**
>
> Se você usar a versão do Enterprise Developer para Windows, os binários gerados pelo compilador poderão ser executados somente no Enterprise Server fornecido com o Enterprise Developer. Você não pode executá-los no tempo de execução do AWS Mainframe Modernization, que é baseado no Linux.

# Tutorial: Configurando a versão da Micro Focus para o aplicativo de amostra BankDemo

AWS Mainframe Modernization oferece a capacidade de configurar compilações e pipelines de integração contínua/entrega contínua (CI/CD) para seus aplicativos migrados. Essas compilações e pipelines usam AWS CodeBuild, AWS CodeCommit, e AWS CodePipeline para fornecer esses recursos. O CodeBuild é um serviço de compilação totalmente gerenciado que compila seu código-fonte, executa testes de unidade e produz artefatos prontos para implantação. O CodeCommit é um serviço de controle de versão que permite armazenar e gerenciar repositórios Git de forma privada na nuvem do AWS. O CodePipeline é um serviço de entrega contínua que permite modelar, visualizar e automatizar as etapas necessárias para liberar seu software.

Este tutorial demonstra como usar o AWS CodeBuild para compilar o código-fonte do aplicativo de amostra BankDemo do Amazon S3 e, em seguida, exportar o código compilado de volta para o Amazon S3.

AWS CodeBuild é um serviço de integração contínuo e totalmente gerenciado que compila o código-fonte, executa testes e produz pacotes de software prontos para implantação. Com o CodeBuild, você pode usar ambientes de compilação pré-empacotados ou pode criar ambientes de compilação personalizados que usam suas próprias ferramentas de compilação. Esse cenário de demonstração usa a segunda opção. Ele consiste em um ambiente de compilação do CodeBuild que usa uma imagem Docker pré-empacotada.

> **Important**
>
> Antes de iniciar seu projeto de modernização de mainframe, recomendamos que você conheça o Programa de aceleração de migração (MAP) da AWS para mainframe ou entre em contato com os especialistas em mainframe da AWS para saber mais sobre as etapas necessárias para modernizar um aplicativo de mainframe.

Tópicos



## Pré-requisitos

Antes de iniciar este tutorial, conclua os seguintes pré-requisitos.

- Faça o download do aplicativo de amostra BankDemo e descompacte-o em uma pasta. A pasta de origem contém programas e cadernos COBOL e definições do CICS BMS. Ele também contém uma pasta JCL para referência, embora você não precise criar o JCL. A pasta também contém os meta-arquivos necessários para a compilação.
- No console do AWS Mainframe Modernization, selecione Ferramentas. Em Análise, desenvolvimento e criação de ativos, selecione Compartilhar ativos com minha conta da AWS.

## Etapa 1: Criar buckets do Amazon S3

Nesta etapa, você cria dois buckets do Amazon S3. O primeiro é um bucket de entrada para armazenar o código-fonte e o outro é um bucket de saída para armazenar a saída da compilação. Para obter mais informações, consulte Criação, configuração e trabalho com buckets do Amazon S3 no Guia do Usuário do Amazon S3.

1. Para criar o bucket de entrada, faça login no console do Amazon S3 e escolha Criar bucket.
2. Em Configuração geral, forneça um nome para o bucket e especifique Região da AWS onde deseja criar o bucket. Um exemplo de nome é`codebuild-regionId-accountId-input-bucket`, where `regionId` is the Região da AWS do bucket e `accountId` é seu Conta da AWS ID.

   > **Note**
   >
   > Se você estiver criando o bucket em um local diferente Região da AWS do Leste dos EUA (Norte da Virgínia), especifique o parâmetro`LocationConstraint`. Para obter mais informações, consulte Criar Bucket na Referência da API do Amazon Simple Storage Service.

3. Mantenha todas as outras configurações e escolha Criar bucket.
4. Repita as etapas de 1 a 3 para criar o bucket de saída. Um exemplo de nome é `codebuild-regionId-accountId-output-bucket`, where `regionId` is the Região da AWS do bucket e `accountId` é seu Conta da AWS ID.

   Independentemente dos nomes que você escolher para esses baldes, não deixe de usá-los ao longo deste tutorial.

# Etapa 2: Criar o arquivo de especificações de compilação

Nesta etapa, você cria um arquivo de especificações de compilação. Esse arquivo fornece comandos de compilação e configurações relacionadas, no formato YAML, para que o CodeBuild execute a compilação. Para obter mais informações, consulte Referência de especificação de compilação para CodeBuild no Guia do Usuário AWS CodeBuild.

1. Crie um arquivo nomeado `buildspec.yml` no diretório que você descompactou como pré-requisito.
2. Adicione o seguinte conteúdo ao arquivo e salve. Nenhuma alteração é necessária para este arquivo.

```
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/**
```

Aqui `CODEBUILD_BUILD_ID`, `CODEBUILD_BUILD_ARN`, `$CODEBUILD_SRC_DIR/source`, e `$CODEBUILD_SRC_DIR/target` estão as variáveis de ambiente disponíveis no CodeBuild. Para obter mais informações, consulte Variáveis de ambiente em ambientes de compilação.

Nesse ponto, seu diretório deve ter a seguinte aparência.

```
(root directory name)
    |-- build.xml
    |-- buildspec.yml
    |-- LICENSE.txt
    |-- source
         |... etc.
```

3. Compacte o conteúdo da pasta em um arquivo chamado `BankDemo.zip` Para este tutorial, não é possível compactar a pasta. Em vez disso, compacte o conteúdo da pasta no arquivo `BankDemo.zip`.

## Etapa 3: Faça upload dos arquivos de origem

Nesta etapa, você carrega o código-fonte do aplicativo de amostra BankDemo no seu bucket de entrada do Amazon S3.

1. Faça login no console do Amazon S3 e escolha Buckets no painel de navegação esquerdo. Em seguida, escolha o bucket de entrada que você criou anteriormente.
2. Em Objetos, escolha Carregar.
3. Na seção Arquivos e pastas, escolha Adicionar arquivos.
4. Navegue e escolha seu arquivo `BankDemo.zip`.
5. Escolha Carregar.

## Etapa 4: criar políticas do IAM

Nesta etapa, você cria duas políticas do IAM. Uma política concede permissões para a modernização do AWS mainframe acessar e usar a imagem do Docker que contém as ferramentas de criação da Micro Focus. Essa política não é personalizada para clientes. A outra política concede permissões para que o AWS Mainframe Modernization interaja com os buckets de entrada e saída e com os logs do Amazon CloudWatch gerados pelo CodeBuild.

Para saber mais sobre como criar uma política de IAM, consulte Edição de políticas de IAM no Guia do Usuário do IAM.

Para criar uma política para acessar imagens do Docker

1. No console do IAM, copie o seguinte documento de política e cole-o no editor de políticas.

```
{
    "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. Forneça um nome para a política, por exemplo, `m2CodeBuildPolicy`.

Para criar uma política que permita que o AWS Mainframe Modernization interaja com buckets e registros

1. No console do IAM, copie o seguinte documento de política e cole-o no editor de políticas. Certifique-se de atualizar `regionId` para o Região da AWS, e `accountId` para o seu Conta da 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. Forneça um nome para a política, por exemplo, `BankdemoCodeBuildRolePolicy`.

# Etapa 5: Criar uma função de IAM

Nesta etapa, você cria uma nova função do IAM que permite que o CodeBuild interaja AWS com os recursos para você, depois de associar as políticas do IAM que você criou anteriormente a essa nova função do IAM.

Para obter informações sobre como criar uma função de serviço, consulte Criação de uma função para delegar permissões a um serviço do AWS no Guia do usuário do IAM.

1. Faça login no console do IAM e escolha Funções no painel de navegação esquerdo.
2. Selecione Criar perfil.
3. Em Tipo de entidade confiável, escolha Serviço AWS.
4. Em Casos de uso para outros serviços da AWS, escolha CodeBuild e, em seguida, escolha CodeBuild novamente.
5. Escolha Next (Próximo).
6. Na página Adicionar permissões, escolha Próximo. Você atribui uma política à função posteriormente.
7. Em Detalhes da função, forneça um nome para a função, por exemplo,`BankdemoCodeBuildServiceRole`.
8. Em Selecionar entidades confiáveis, verifique se o documento de política tem a seguinte aparência:

```
{
          "Version": "2012-10-17",
          "Statement": [
            {
              "Effect": "Allow",
              "Principal": {
                "Service": "codebuild.amazonaws.com"
              },
              "Action": "sts:AssumeRole"
            }
          ]
        }
```

9. Selecione Criar perfil.

## Etapa 6: Anexe as políticas de IAM à função de IAM

Nesta etapa, você anexa as duas políticas de IAM que criou anteriormente à função do `BankdemoCodeBuildServiceRole` IAM.

1. Faça login no console do IAM e escolha Funções no painel de navegação esquerdo.
2. Em Função do IAM, escolha a função que você criou anteriormente, por exemplo, `BankdemoCodeBuildServiceRole`.
3. Em Políticas de permissões, escolha Adicionar permissões e, em seguida, Anexar políticas.
4. Em Outras políticas de permissões, escolha as políticas que você criou anteriormente, por exemplo, `m2CodeBuildPolicy` e `BankdemoCodeBuildRolePolicy`.
5. Escolha Anexar políticas.

## Etapa 7: Criar o projeto do CodeBuild

Nesta etapa, você cria o projeto do CodeBuild.

1. Faça login no console do CodeBuild e escolha Criar projeto de compilação.
2. Na seção Configuração do projeto, forneça um nome para o projeto, por exemplo,`codebuild-bankdemo-project`.
3. Na seção Fonte, em Provedor de origem, escolha Amazon S3 e, em seguida, escolha o bucket de entrada que você criou anteriormente, por exemplo, `codebuild-regionId-accountId-input-bucket`.
4. No campo Chave do objeto do S3 ou pasta do S3, insira o nome do arquivo zip que você carregou no bucket do S3. Nesse caso, o nome do arquivo é `bankdemo.zip`.
5. Na seção Ambiente, escolha Imagem personalizada.
6. No campo Tipo de ambiente, escolha Linux.
7. Em Registro de imagens, escolha Outro registro.
8. No campo URL do registro externo, digite `673918848628.dkr.ecr.us-west-2.amazonaws.com/m2-enterprise-build-tools:latest`
9. Em Função de serviço, escolha Função de serviço existente e, no campo ARN da função, escolha a função de serviço que você criou anteriormente; por exemplo,. `BankdemoCodeBuildServiceRole`

10. Na seção Buildspec, escolha Usar um arquivo buildspec.
11. Na seção Artefatos, em Tipo, escolha Amazon S3 e, em seguida, escolha seu bucket de saída, por exemplo, `codebuild-regionId-accountId-output-bucket`.
12. No campo Nome, insira o nome de uma pasta no bucket que você deseja que contenha os artefatos de saída da compilação, por exemplo,`bankdemo-output.zip`.
13. Em Embalagem de artefatos, escolha Zip.
14. Selecione Create build project (Criar projeto de compilação).

## Etapa 8: Iniciar a construção

Nesta etapa, você inicia a compilação.

1. Faça login no console do CodeBuild.
2. No painel de navegação esquerdo, selecione Criar projetos.
3. Escolha o projeto de construção que você criou anteriormente, por exemplo,`codebuild-bankdemo-project`.
4. Selecione Start build.

Esse comando inicia a compilação. A compilação é executada de forma assíncrona. A saída do comando é um JSON que inclui o id do atributo. Esse id de atributo é uma referência ao ID de compilação do CodeBuild da compilação que você acabou de iniciar. Você pode visualizar o status da compilação no console do CodeBuild. Você também pode ver registros detalhados sobre a execução da compilação no console. Para obter mais informações, consulte Exibir informações detalhadas da compilação no Guia do usuário AWS CodeBuild.

Quando a fase atual for CONCLUÍDA, significa que sua compilação foi concluída com sucesso e que seus artefatos compilados estão prontos no Amazon S3.

## Etapa 9: Faça o download dos artefatos de saída

Nesta etapa, você baixa os artefatos de saída do Amazon S3. A ferramenta de compilação da Micro Focus pode criar vários tipos diferentes de executáveis. Neste tutorial, ele gera objetos compartilhados.

1. Faça login no console do Amazon S3.

2. Na seção Buckets, escolha o nome do seu bucket de saída, por exemplo, `codebuild-regionId-accountId-output-bucket`.
3. Escolha Baixar.
4. Não descompacte o arquivo obtido por download. Navegue até a pasta de destino para ver os artefatos de construção. Isso inclui os objetos compartilhados do `.so` Linux.

## Limpeza de recursos

Se os recursos criados para este tutorial não forem mais necessários, exclua-os para evitar cobranças adicionais. Para fazer isso, conclua as etapas a seguir:

- Exclua os buckets do S3 que você criou para este tutorial. Para obter mais informações, consulte Excluir um bucket no Guia do usuário do Amazon Simple Storage Service.
- Exclua as políticas de IAM que você criou para este tutorial. Para obter mais informações, consulte Exclusão de políticas do IAM no Guia do Usuário do IAM.
- Exclua a função IAM que você criou para este tutorial. Para obter mais informações sobre como excluir uma função, consulte Excluir funções ou perfis de instância no Manual do usuário do IAM.
- Exclua o projeto do CodeBuild que você criou para este tutorial. Para obter mais informações, consulte Excluir um projeto de compilação no CodeBuild no Guia do Usuário AWS CodeBuild.

# Tutorial: Configurando um pipeline de CI/CD para uso com o Micro Focus Enterprise Developer

Este tutorial mostra como importar, editar, compilar e executar o aplicativo de amostra BankDemo no Micro Focus Enterprise Developer e, em seguida, confirmar suas alterações para acionar um pipeline de CI/CD.

Sumário

## Pré-requisitos

Faça o download dos seguintes arquivos.

- `basic-infra.yaml`
  - Baixe da região Europa (Frankfurt).
  - Baixe da região Leste dos EUA (Norte da Virgínia).
- `pipeline.yaml`
  - Baixe da região Europa (Frankfurt).
  - Baixe da região Leste dos EUA (Norte da Virgínia).
- `m2-code-sync-function.zip`
  - Baixe da região Europa (Frankfurt).

  - Baixe da região Leste dos EUA (Norte da Virgínia).
- `config_git.yml`
  - Baixe da região Europa (Frankfurt).
  - Baixe da região Leste dos EUA (Norte da Virgínia).
- `BANKDEMO-source.zip`
  - Baixe da região Europa (Frankfurt).
  - Baixe da região Leste dos EUA (Norte da Virgínia).
- `BANKDEMO-exercise.zip`
  - Baixe da região Europa (Frankfurt).
  - Baixe da região Leste dos EUA (Norte da Virgínia).

A finalidade de cada arquivo é a seguinte:

`basic-infra.yaml`

Esse modelo AWS CloudFormation cria a infraestrutura básica necessária para o pipeline de CI/CD: VPC, buckets Amazon S3 e assim por diante.

`pipeline.yaml`

Esse modelo AWS CloudFormation é usado por uma função Lambda para iniciar a pilha do pipeline. Certifique-se de que esse modelo esteja localizado em um bucket do Amazon S3 acessível publicamente. Adicione o link para esse bucket como o valor padrão para o `PipelineTemplateURL` parâmetro no modelo `basic-infra.yaml`.

`m2-code-sync-function.zip`

Essa função Lambda cria o repositório CodeCommit, a estrutura de diretórios baseada no, e inicia a pilha de pipeline `config_git.yaml` usando `pipeline.yaml` Certifique-se de que esse arquivo zip esteja disponível em um bucket do Amazon S3 acessível publicamente em todas as Regiões da AWS em que o AWS Mainframe Modernization é suportado. Recomendamos que você armazene o arquivo em um bucket em um Região da AWS e o replique em buckets em todos Regiões da AWS. Use uma convenção de nomenclatura para o intervalo com um sufixo que identifique o específico Região da AWS (por exemplo,`m2-cicd-deployment-source-eu-west-1`) e adicione o prefixo `m2-cicd-deployment-source` como valor padrão para o parâmetro `DeploymentSourceBucket` e forme o intervalo completo usando a função de substituição AWS CloudFormation `!Sub {DeploymentSourceBucket}-`

`${AWS::Region}` ao se referir a esse intervalo no modelo `basic-infra.yaml` de recurso `SourceSyncLambdaFunction`.

`config_git.yml`

Definição da estrutura de diretórios do CodeCommit. Para obter mais informações, consulte Exemplo de arquivo acionador YAML config_git.yml.

`BANKDEMO-source.zip`.

Código-fonte e arquivo de configuração do BankDemo criados a partir do repositório CodeCommit.

`BANKDEMO-exercise.zip`.

Fonte BankDemo para exercícios tutoriais criados a partir do repositório CodeCommit.

## Crie uma infraestrutura básica de pipeline de CI/CD

Use o modelo AWS CloudFormation `basic-infra.yaml` para criar a pilha de infraestrutura básica do pipeline de CI/CD por meio do console AWS CloudFormation. Essa pilha cria buckets do Amazon S3 nos quais você faz upload do código e dos dados do aplicativo e uma função de suporte do AWS Lambda para criar outros recursos necessários, como um repositório AWS CodeCommit e um pipeline AWS CodePipeline.

> **Note**
>
> Para iniciar essa pilha, você precisa de permissões para administrar o IAM, o Amazon S3, o Lambda e o AWS CloudFormation, além de permissões para usar o AWS KMS.

1. Faça login no AWS Management Console e abra o console AWS CloudFormation em https://console.aws.amazon.com/cloudformation.
2. Crie uma nova pilha usando uma das seguintes opções:
   - Selecione Create Stack (Criar pilha). Esta será a única opção se houver uma pilha em execução no momento.
   - Na página Pilhas, selecione Criar pilha. Essa opção aparecerá somente se não houver pilhas em execução.
3. Na página Especificar modelo:

- Em Preparar modelo, selecione O modelo está pronto.
- Em Especificar modelo, escolha o URL do Amazon S3 como fonte do modelo e insira um dos seguintes URLs, dependendo do seu Região da 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`
- Para aceitar suas configurações, selecione Próximo.

A página Criar pilha é aberta.

# 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

Faça as seguintes alterações em:

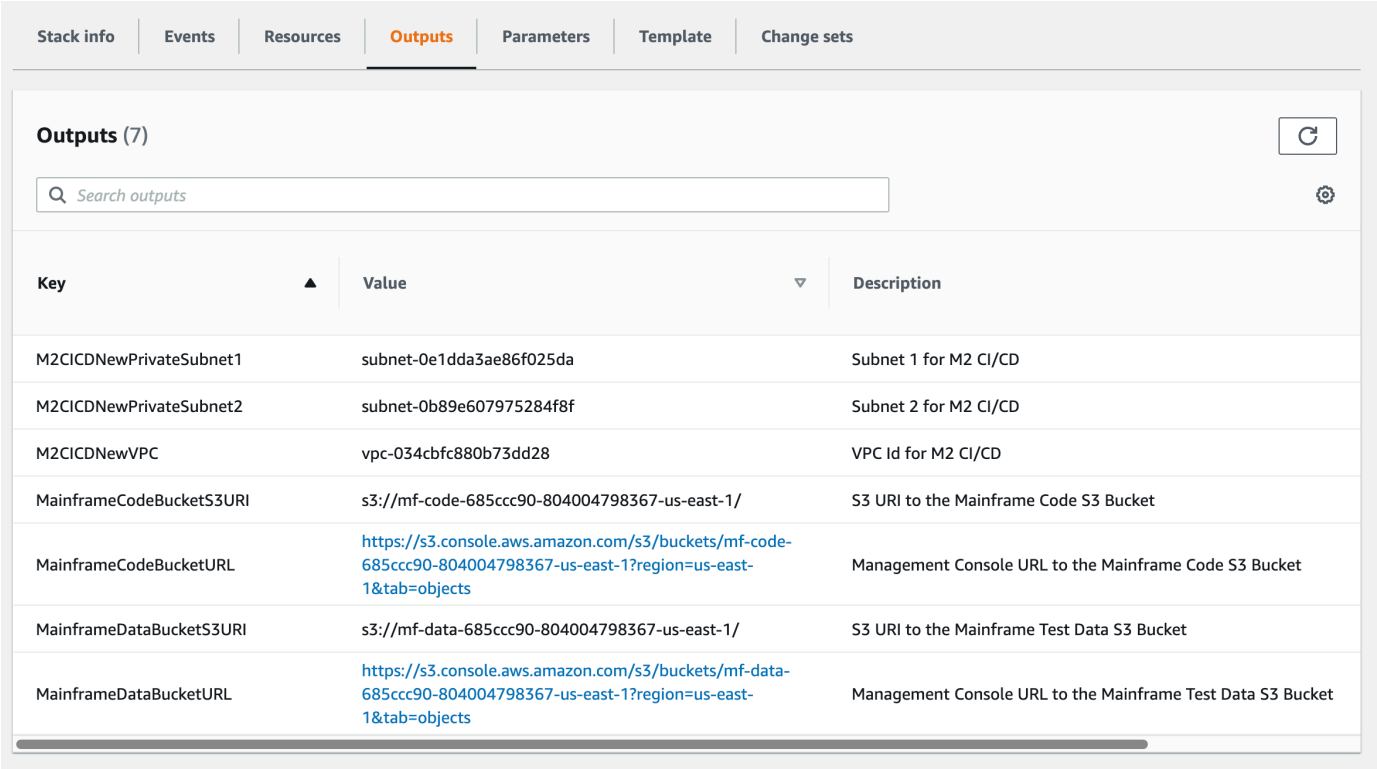
- Forneça os valores apropriados para o Nome da pilha e os parâmetros para a Configuração de rede.
- A maioria dos parâmetros nas Configurações de implantação são pré-preenchidos adequadamente para que você não precise modificá-los. Dependendo da sua Região da AWS, altere o modelo AWS CloudFormation do pipeline para um dos seguintes URLs do Amazon S3.
  - `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`
- Escolha Next (Próximo).

> **Note**
>
> Não altere os valores dos parâmetros padrão, a menos que você mesmo tenha modificado o modelo AWS CloudFormation.

4. Em Configurar opções de pilha, selecione Próximo.
5. Em Recursos, escolha Eu reconheço que a AWS CloudFormation pode criar recursos de IAM para permitir que o AWS CloudFormation crie uma função de IAM em seu nome. Selecione Criar pilha.

> **Note**
>
> Pode levar de 3 a 5 minutos para que essa pilha seja provisionada.

6. Depois que a pilha for criada com sucesso, navegue até a seção Saídas da pilha recém-provisionada. Lá você encontrará o bucket do Amazon S3 onde você precisa fazer o upload do código do mainframe e dos arquivos dependentes.




## Crie repositório AWS CodeCommit e pipeline de CI/CD

Nesta etapa, você cria um repositório do CodeCommit e provisiona uma pilha de pipeline de CI/CD chamando uma função Lambda que chama para criar a pilha de pipeline AWS CloudFormation.

1. Faça o download aplicativo de amostra BankDemo em sua máquina local.
2. Faça o upload `bankdemo.zip` da sua máquina local para o bucket Amazon S3 criado em Crie uma infraestrutura básica de pipeline de CI/CD.
3. Baixar `config_git.yml`.
4. Modifique o `config_git.yml`, se necessário, da seguinte maneira:

   - Adicione seu próprio nome de repositório de destino, ramificação de destino e mensagem de confirmação.

   ```
   repository-config:
     target-repository: bankdemo-repo
     target-branch: main
     commit-message: Initial commit for bankdemo-repo main branch
   ```

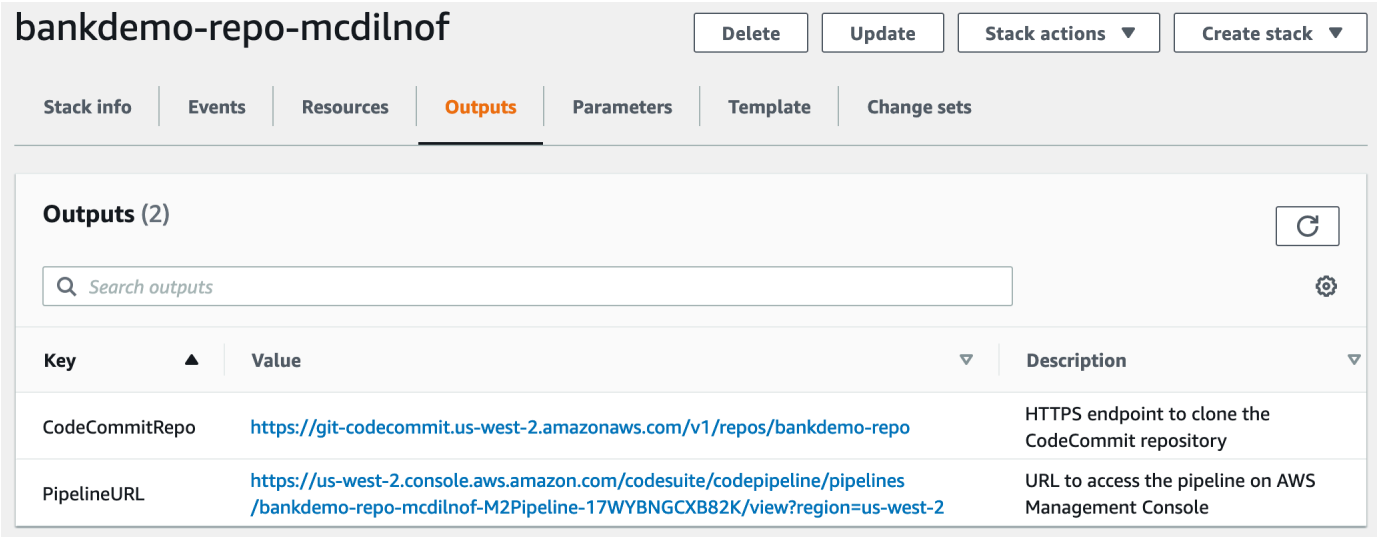
   - Adicione o endereço de e-mail que você deseja receber notificações.

```
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. Faça upload do arquivo `config_git.yml` que contém a definição da estrutura de pastas do repositório do CodeCommit para o bucket do Amazon S3 criado em Crie uma infraestrutura básica de pipeline de CI/CD. Isso invocará a função Lambda que provisionará automaticamente o repositório e o pipeline.

   Isso criará um repositório do CodeCommit com o nome fornecido no `target-repository` definido no arquivo `config_git.yml`; por exemplo, `bankdemo-repo`.

   A função Lambda também criará a pilha de pipeline de CI/CD por meio de AWS CloudFormation. A pilha AWS CloudFormation terá o mesmo prefixo que o nome `target-repository` fornecido, seguido de uma cadeia aleatória (por exemplo, `bankdemo-repo-`*`01234567`*. Você pode encontrar a URL do repositório CodeCommit e a URL para acessar o pipeline criado no AWS Management Console.




6. Se a criação do repositório CodeCommit for concluída, o pipeline de CI/CD será acionado imediatamente para realizar um CI/CD completo.

7. Depois que o arquivo for enviado, ele acionará automaticamente o pipeline, que será construído, implantado em teste, executará alguns testes e aguardará a aprovação manual antes de implantá-lo no ambiente de produção.

## Exemplo de arquivo acionador YAML config_git.yml

```
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
```

## Criação do ApStream 2.0 ApStream 2.0 para desenvolvedores corporativos

Para configurar o Micro Focus Enterprise Developer no AppStream 2.0, consulte Tutorial: Configurar o Micro Focus Enterprise Developer no AppStream 2.0.

Para conectar o repositório do CodeCommit ao Enterprise Developer, use o nome especificado em `target-repository` em Exemplo de arquivo acionador YAML config_git.yml.

## Configuração e teste para desenvolvedores corporativos

Tópicos

- Clone o repositório BankDemo CodeCommit no Enterprise Developer
- Crie um projeto COBOL de mainframe BankDemo e crie um aplicativo
- Crie um ambiente local de CICS e lotes do BankDemo para testes
- Inicie o servidor BANKDEMO do Enterprise Developer
- Inicie o terminal Rumba 3270
- Execute uma transação BankDemo
- Interrompa o servidor BANKDEMO no Enterprise Developer

Conecte-se à instância Enterprise Developer AppStream 2.0 em que você criou Criação do ApStream 2.0 ApStream 2.0 para desenvolvedores corporativos.

1. Inicie o Enterprise Developer a partir do Windows Start. Escolha Micro Focus Enterprise Developer e, em seguida, escolha Enterprise Developer for Eclipse. Se você está começando pela primeira vez, pode levar algum tempo.
2. No Eclipse Launcher, no Workspace: entre e `C:\Users\<username>\workspace` escolha Launch..

   > **Note**
   >
   > Certifique-se de escolher o mesmo local depois de se reconectar à instância do AppStream 2.0. A seleção do espaço de trabalho não é persistente.

3. Em Bem-vindo, escolha Open COBOL Perspective. Isso só será exibido na primeira vez em um novo espaço de trabalho.

## Clone o repositório BankDemo CodeCommit no Enterprise Developer

1. Escolha Janela / Perspectiva / Abrir perspectiva / Outro ... / Git.
2. Escolha Clonar um repositório Git.
3. Em Clonar repositório Git, insira as seguintes informações:

   - Em URI de localização, insira a URL HTTPS do repositório CodeCommit.

     > **Note**
     >
     > Copie o URL do clone HTTPS para o repositório do CodeCommit no AWS Management Console e cole-o aqui. O URI será dividido nos caminhos Host e Repositório.

   - As credenciais do repositório do usuário CodeCommit em Autenticação de usuário e Senha e escolha Armazenamento in Armazenamento seguro.
4. Em Seleção de ramificação, escolha Ramificação principal e, em seguida, escolha Próximo.
5. Em Destino local, em Diretório, insira `C:\Users\<username>\workspace` e escolha Concluir.

O processo de clonagem é concluído quando `BANKDEMO [main]` exibido na visualização Repositórios Git.

## Crie um projeto COBOL de mainframe BankDemo e crie um aplicativo

1. Altere para a perspectiva COBOL.
2. No Projeto, desative Criar automaticamente.
3. Em Arquivo, escolha Novo e, em seguida, Projeto COBOL Mainframe.
4. Em Novo projeto COBOL mainframe, insira as seguintes informações:
   - Em Nome do projeto, digite `BankDemo`.
   - Escolha o modelo Micro Focus [64 bits].
   - Escolha Terminar.
5. No COBOL Explorer, expanda o novo projeto BankDemo.

   > **Note**
   >
   > `[BANKDEMO main]` entre colchetes indica que o projeto está conectado ao repositório local do BankDemo CodeCommit.

6. Se a visualização em árvore não mostrar entradas para programas COBOL, cadernos, código-fonte BMS e arquivos JCL, escolha Atualizar no menu de contexto do projeto BankDemo.
7. No menu de contexto do BankDemo, escolha Propriedades / Micro Focus / Configurações do projeto / COBOL:
   - Escolha Conjunto de caracteres - ASCII.
   - Escolha Aplicar e, em seguida, Fechar.
8. Se a compilação da fonte BMS e COBOL não for iniciada imediatamente, verifique no menu Projeto se a opção Criar Automaticamente está ativada.

   A saída do Build será exibida na visualização do Console e deverá ser concluída após alguns minutos com mensagens `BUILD SUCCESSFUL` e `Build finished with no errors`.

   O aplicativo BankDemo agora deve estar compilado e pronto para execução local.

## Crie um ambiente local de CICS e lotes do BankDemo para testes

1. No COBOL Explorer, expandaBANKDEMO / config.
2. No editor, abra BANKDEMO_ED.json.
3. Localize a string ED_Home= e altere o caminho para apontar para o projeto Enterprise Developer, da seguinte forma: D:\\<username>\\workspace\\BANKDEMO. Observe o uso de barras duplas (\\) na definição do caminho.
4. Salve e feche o arquivo.
5. Escolha Server Explorer.
6. No menu de contexto Padrão, escolha Abrir página de administração. A página de administração do Micro Focus Enterprise Server é aberta no navegador padrão.
7. Somente para sessões do AppStream 2.0, faça as seguintes alterações para que você possa preservar sua região local do Enterprise Server para testes locais:
   - Em Servidor de Diretórios / Padrão, escolha PROPRIEDADES / Configuração.
   - Substitua a localização do repositório porD:\<username>\My Files\Home Folder\MFDS.

   > **Note**
   >
   > Você deve concluir as etapas 5 a 8 após cada nova conexão com uma instância do AppStream 2.0.

8. Em Servidor de Diretórios / Padrão, escolha Importar e conclua as seguintes etapas:
   - Na Etapa 1: Tipo de importação, escolha JSON e escolha Próximo.
   - Na Etapa 2: Carregar, clique para fazer upload do arquivo no quadrado azul.
   - Em Escolher arquivo para carregar, digite:
     - Nome do arquivo: D:\<username>\workspace\BANKDEMO\config \BANKDEMO_ED.json.
     - Escolha Open (Abrir).
   - Escolha Next (Próximo).
   - Na Etapa 3: as regiões limpam as portas limpas dos endpoints.
   - Escolha Next (Próximo).
   - Na Etapa 4: Importar, escolha Importar.

- Escolha Terminar.

A lista agora mostrará um novo nome de servidor `BANKDEMO`.

## Inicie o servidor BANKDEMO do Enterprise Developer

1. Escolha Enterprise Developer.
2. No Explorador do servidor, escolha Padrão e, em seguida, escolha Atualizar no menu de contexto.

   A lista de servidores agora também deve mostrar BANKDEMO.
3. Escolha BANKDEMO.
4. No menu de contexto, escolha Associar ao projeto e escolha BANKDEMO.
5. No menu contextual, escolha Iniciar.

   A visualização do console deve exibir o registro da inicialização do servidor.

   Se a mensagem `BANKDEMO CASSI5030I PLTPI Phase 2 List(PI) Processing Completed` for exibida, o servidor está pronto para testar o aplicativo CICS BANKDEMO.

## Inicie o terminal Rumba 3270

1. No Windows Start, inicie o Micro Focus Rumba+ Desktop /Rumba+ Desktop.
2. Em Bem-vindo, escolha CREATE NEW SESSION/Mainframe Display.
3. No Mainframe Display, escolha Conexão / Configurar.
4. Em Configuração da sessão, escolha Connection/TN3270.
5. Em Nome do host/Endereço, escolha Inserir e insira o endereço IP `127.0.0.1`.
6. Em Porta Telnet, insira porta `6000`.
7. Escolha Aplicar.
8. Selecione Conectar.

   A tela de boas-vindas do CICS exibe a tela com a mensagem da linha 1:`This is the Micro Focus MFE CICS region BANKDEMO`.
9. Pressione Ctrl+Shift+z para limpar a tela.

### Execute uma transação BankDemo

1. Em uma tela vazia, digite BANK.
2. Na tela BANK10, no campo de entrada para ID do usuário... :, digite `guest` e pressione Enter.
3. Na tela BANK20, no campo de entrada antes de Calcular o custo de um empréstimo, digite `/` (barra) e pressione Enter.
4. Na tela BANK70:

   - Em O valor que você gostaria de emprestar...:, digite `10000`.
   - Em A uma taxa de juros de... :, digite `5.0`.
   - Em Por quantos meses...: , digite`10`.
   - Pressione Enter.

   O seguinte resultado deve ser exibido:

   ```
   Resulting monthly payment.............:   $1023.06
   ```

   Isso conclui a configuração do aplicativo BANKDEMO no Enterprise Developer.

### Interrompa o servidor BANKDEMO no Enterprise Developer

1. No Explorador do servidor, escolha Padrão e, em seguida, escolha Atualizar no menu de contexto.
2. Escolha BANKDEMO.
3. No menu de contexto, selecione Interromper.

   A visualização do Console deve exibir o registro da parada do servidor.

   Se a mensagem `Server: BANKDEMO stopped successfully` for exibida, o servidor foi desligado com sucesso.

## Exercício 1: Aprimorar o cálculo do empréstimo no aplicativo BANKDEMO

Tópicos

- Adicionar regra de análise de empréstimos à Enterprise Developer Code Analysis

- Etapa 1: realizar a análise do código para o cálculo do empréstimo
- Etapa 2: Modificar o mapa do CICS BMS e o programa e teste COBOL
- Etapa 3: Adicionar cálculo do valor total no programa COBOL
- Etapa 4: confirmar as alterações e executar o pipeline de CI/CD

Nesse cenário, você percorre o processo de fazer uma alteração de amostra no código, implantá-lo e testá-lo.

O departamento de empréstimos deseja um novo campo na tela de cálculo do empréstimo BANK70 para mostrar o valor total do empréstimo. Isso requer uma alteração na tela BMS MBANK70.CBL, adicionando um novo campo e o programa de tratamento de tela correspondente SBANK70P.CBL com cadernos relacionados. Além disso, a rotina de cálculo do empréstimo no BBANK70P.CBL precisa ser estendida com a fórmula adicional.

Para concluir este exercício, preencha os pré-requisitos a seguir.

- Faça o download de BANKDEMO-exercise.zip para `D:\PhotonUser\My Files\Home Folder`.
- Extraia o arquivo zip para `D:\PhotonUser\My Files\Home Folder\BANKDEMO-exercise`.
- Criar pasta `D:\PhotonUser\My Files\Home Folder\AnalysisRules`
- Copie o arquivo `Loan+Calculation+Update.General-1.xml` de regras da `BANKDEMO-exercise` pasta para `D:\PhotonUser\My Files\Home Folder\AnalysisRules`.

> **Note**
>
> As alterações de código em *.CBL e *.CPY são marcadas com EXER01 nas colunas 1 a 6 deste exercício.

## Adicionar regra de análise de empréstimos à Enterprise Developer Code Analysis

As regras de análise definidas no Micro Focus Enterprise Analyzer podem ser exportadas do Enterprise Analyzer e importadas para o Enterprise Developer para executar as mesmas regras de análise nas fontes do projeto Enterprise Developer.

1. Abra o `Window/Preferences/Micro Focus/COBOL/Code Analysis/Rules`.

2. Escolha Editar... e insira o nome da pasta `D:\PhotonUser\My Files\Home Folder\AnalysisRules` que contém o arquivo de regras `Loan+Calculation +Update.General-1.xml`.
3. Escolha Terminar.
4. Escolha Aplicar e depois Fechar.
5. No menu de contexto do projeto BANKDEMO, escolha Análise de código.

   Você deve ver uma entrada para Atualização do cálculo do empréstimo.

## Etapa 1: realizar a análise do código para o cálculo do empréstimo

Com a nova regra de análise, queremos identificar os programas COBOL e as linhas de código que correspondem aos padrões de pesquisa `*PAYMENT*`, `*LOAN*` e `*RATE*` em expressões, instruções e variáveis. Isso ajudará a navegar pelo código e identificar as alterações de código necessárias.

1. No menu de contexto do projeto BANKDEMO, selecione Code Analysis/Loan Calculation Update.

   Isso executará a regra de pesquisa e listará os resultados em uma nova guia chamada Análise de código. A execução da análise é concluída quando a barra de progresso verde no canto inferior direito desaparece.

   A guia Análise de código deve exibir uma lista expandida de `BBANK20P.CBL`, `BBANK70P.CBL` e `SBANK70P.CBL`, e cada uma listando as declarações, expressões e variáveis que correspondem aos padrões de pesquisa.

   Observando o resultado, `BBANK20P.CBL` há apenas literais movidos que correspondem ao padrão de pesquisa. Portanto, esse programa pode ser ignorado.
2. Na barra de menu da guia, escolha - Ícone para recolher tudo.
3. Expanda `SBANK70P.CBL` e selecione qualquer linha em qualquer ordem com um clique duplo para ver como isso abrirá a fonte e destacará a linha selecionada no código-fonte. Você também reconhecerá que todas as linhas de origem identificadas estão marcadas.

## Etapa 2: Modificar o mapa do CICS BMS e o programa e teste COBOL

Primeiro, alteraremos o mapa `MBANK70.BMS` BMS, o programa de manipulação de tela `SBANK70P.CBL` e o caderno `CBANKDAT.CPY` para exibir o novo campo. Para evitar codificação

desnecessária neste exercício, os módulos de origem modificados estão disponíveis na pasta `D:\PhotonUser\My Files\Home Folder\BANKDEMO-exercise\Exercise01`. Normalmente, um desenvolvedor usaria os resultados da Análise de Código para navegar e modificar as fontes. Se você tiver tempo e quiser fazer as alterações manuais, faça-o com as informações fornecidas em *Alteração manual no MBANK70.BMS e no SBANK70P.CBL (opcional) *.

Para alterações rápidas, copie os seguintes arquivos:

1. `..\BANKDEMO-exercise\Exercis01\screens\MBANK70.BMS` para `D:\PhotonUser\workspace\bankdemo\source\screens`.
2. `.\BANKDEMO-exercise\Exercis01\cobol\SBANK70P.CBL` para `D:\PhotonUser\workspace\bankdemo\source\cobol`.
3. `..\BANKDEMO-exercise\Exercis01\copybook\CBANKDAT.CPY` para `D:\PhotonUser\workspace\bankdemo\source\copybook`.
4. Para garantir que todos os programas afetados pelas alterações sejam compilados, escolha Projetar/Limpar... /Limpe todo o projeto.

Para alterações manuais em `MBANK70.BMS` e `SBANK70P.CBL`, conclua as seguintes etapas:

- Para alteração manual na `MBANK70.BMS` fonte BMS, adicione após o campo `PAYMENT`:
  - TXT09 com os mesmos atributos do TXT08 e valor INICIAL “Valor total do empréstimo”
  - TOTAL com os mesmos atributos de PAGAMENTO

Alterações no teste

Para testar as alterações, repita as etapas nas seguintes seções:

1. Inicie o servidor BANKDEMO do Enterprise Developer
2. Inicie o terminal Rumba 3270
3. Execute uma transação BankDemo

   Além disso, agora você também deve ver o texto `Total Loan Amount.....:`.
4. Interrompa o servidor BANKDEMO no Enterprise Developer

## Etapa 3: Adicionar cálculo do valor total no programa COBOL

Na segunda etapa, alteraremos o `BBANK70P.CBL` e adicionaremos o cálculo do valor total do empréstimo. A fonte preparada com as alterações necessárias está disponível na pasta `D:\PhotonUser\My Files\Home Folder\BANKDEMO-exercise\Exercise01`. Se você tiver tempo e quiser fazer as alterações manuais, faça-as com as informações fornecidas em *Alteração manual no BBANK70P.CBL (opcional)*.

Para uma mudança rápida, copie o seguinte arquivo:

- `..\BANKDEMO-exercise\Exercis01\source\cobol\BBANK70P.CBL` para `D:\PhotonUser\workspace\bankdemo\source\cobol`.

Para fazer uma alteração manual no `BBANK70P.CBL`, conclua as seguintes etapas:

- Use o resultado da Análise de Código para identificar as alterações necessárias.

Alterações no teste

Para testar as alterações, repita as etapas nas seguintes seções:

1. Inicie o servidor BANKDEMO do Enterprise Developer
2. Inicie o terminal Rumba 3270
3. Execute uma transação BankDemo

   Além disso, agora você também deve ver o texto `Total Loan Amount.....................: $10230.60`.
4. Interrompa o servidor BANKDEMO no Enterprise Developer

## Etapa 4: confirmar as alterações e executar o pipeline de CI/CD

Confirme as alterações no repositório central do CodeCommit e acione o pipeline de CI/CD para criar, testar e implantar as alterações.

1. No projeto BANKDEMO, no menu de contexto, selecione Team/Commit.
2. Na guia Git Staging, digite a seguinte mensagem de confirmação: `Added Total Amount Calculation`.

3. Esclha Confirmar e Enviar….
4. Abra o console do CodePipeline e verifique o status da execução do pipeline.

> **Note**
>
> Caso você enfrente algum problema com a função Commit ou Push do Enterprise Developer ou do Teams, use a interface de linha de comando do Git Bash.

## Exercício 2: Extrair o cálculo do empréstimo no aplicativo BankDemo

Tópicos



No próximo exercício, você trabalha com outra solicitação de alteração de amostra. Nesse cenário, o departamento de empréstimos deseja reutilizar a rotina de cálculo do empréstimo como um WebService independente. A rotina deve permanecer em COBOL e também deve ser chamada a partir do programa CICS COBOL existente. `BBANK70P.CBL`

### Etapa 1: Refatorar a rotina de cálculo do empréstimo em uma seção COBOL

Na primeira etapa, extraímos a rotina de cálculo do empréstimo em uma seção COBOL. Essa etapa é necessária para extrair o código em um programa COBOL independente na próxima etapa.

1. Abra o `BBANK70P.CBL` no editor COBOL.
2. No editor, selecione no menu de contexto Code Analysis/Loan Calculation Update. Isso examinará apenas a fonte atual em busca de padrões definidos na regra de análise.
3. No resultado na guia Análise de código, encontre a primeira declaração aritmética `DIVIDE WS-LOAN-INTEREST BY 12`.
4. Clique duas vezes na declaração para navegar até a linha de origem no Editor. Essa é a primeira declaração da rotina de cálculo do empréstimo.
5. Marque o seguinte bloco de código para que a rotina de cálculo do empréstimo seja extraída em uma seção.

```
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. No menu de contexto do editor, escolha Refatorar/Extrair para seção....
7. Insira o nome da nova seção: CÁLCULO DO EMPRÉSTIMO.
8. Escolha OK.

   O bloco de código marcado agora foi extraído para a nova `LOAN-CALCULATION` seção e o bloco de código foi substituído pela instrução `PERFROM LOAN-CALCULATION`.

Alterações no teste

Para testar as alterações, repita as etapas descritas nas seções a seguir.

1. Inicie o servidor BANKDEMO do Enterprise Developer
2. Inicie o terminal Rumba 3270
3. Execute uma transação BankDemo

   Além disso, agora você também deve ver o texto `Total Loan Amount.....................: $10230.60`.
4. Interrompa o servidor BANKDEMO no Enterprise Developer

> **Note**
>
> Se você quiser evitar as etapas acima para extrair o bloco de código para uma seção, você pode copiar a fonte modificada para a Etapa 1 de `..\BANKDEMO-exercise\Exercis02\Step1\cobol\BBANK70P.CBL` para `D:\PhotonUser\workspace\bankdemo\source\cobol`.

## Etapa 2: extrair a rotina de cálculo do empréstimo para um programa COBOL independente

Na Etapa 2, o bloco de código na seção `LOAN-CALCULATION` será extraído para um programa independente e o código original será substituído pelo código para chamar o novo subprograma.

1. Abra `BBANK70P.CBL` no editor e encontre a nova declaração `PERFORM LOAN-CALCULATION` criada na Etapa 1.
2. Coloque o cursor dentro do nome da seção. Estará marcado em cinza.
3. No menu de contexto, selecione Refatorar->Extrair seção/parágrafo para o programa…….
4. Em Extrair seção/parágrafo para o programa, insira o Nome do novo arquivo: LOANCALC.CBL.
5. Escolha OK.

   O novo `LOANCALC.CBL` programa será aberto no editor.
6. Role para baixo e revise o código que está sendo extraído e gerado para a interface de chamada.
7. Selecione o editor com `BBANK70P.CBL` e vá para `LOAN-CALCULATION SECTION`. Revise o código que está sendo gerado para chamar o novo subprograma `LOANCALC.CBL`.

> **Note**
>
> A `CALL` instrução está usando `DFHEIBLK` e `DFHCOMMAREA` para chamar `LOANCALC` com blocos de controle do CICS. Como queremos chamar o novo `LOANCALC.CBL` subprograma de programa não CICS, precisamos remover `DFHEIBLK` e sair `DFHCOMMAREA` da chamada comentando ou excluindo.

Alterações no teste

Para testar as alterações, repita as etapas descritas nas seções a seguir.

1. Inicie o servidor BANKDEMO do Enterprise Developer
2. Inicie o terminal Rumba 3270
3. Execute uma transação BankDemo

   Além disso, agora você também deve ver o texto `Total Loan Amount.....: $10230.60`.

4. Interrompa o servidor BANKDEMO no Enterprise Developer

> **Note**
>
> Se quiser evitar as etapas acima para extrair o bloco de código em uma seção, você pode copiar a fonte modificada para a Etapa 1 de `..\BANKDEMO-exercise\Exercis02\Step2\cobol\BBANK70P.CBL` e `LOANCALC.CBL` para `D:\PhotonUser\workspace\bankdemo\source\cobol`.

### Etapa 3: confirmar as alterações e executar o pipeline de CI/CD

Faça o commit das alterações no repositório central do CodeCommit e acione o pipeline de CI/CD para criar, testar e implementar as alterações.

1. No projeto BANKDEMO, no menu de contexto, selecione Team/Commit.
2. Na guia Git Staging
   - Adicione os estágios não preparados LOANCALC.CBL e LoanCalc.CBL.MFDirSet.
   - Insira uma mensagem de confirmação:`Added Total Amount Calculation`.
3. Esclha Confirmar e Enviar….
4. Abra o console do CodePipeline e verifique o status da execução do pipeline.

> **Note**
>
> Caso você enfrente algum problema com a função Commit ou Push do Enterprise Developer ou do Teams, use a interface de linha de comando do Git Bash.

## Limpeza de recursos

Se os recursos criados durante este tutorial não forem mais necessários, exclua-os para que você não continue sendo cobrado por eles. Execute as etapas a seguir:

- Exclua o pipeline do CodePipeline. Para obter mais informações, consulte Excluir um pipeline no CodePipeline, no Guia do usuário do AWS CodePipeline.

- Para excluir o repositório do CodeCommit Para obter mais informações, consulte Excluir um repositório do CodeCommit no Guia do usuário AWS CodeCommit.
- Exclua o bucket do S3. Para obter mais informações, consulte Excluir um bucket no Guia do usuário do Amazon Simple Storage Service.
- Exclua a pilha AWS CloudFormation. Para obter mais informações, consulte Exclusão de uma pilha no console do AWS CloudFormation no Guia do usuário do AWS CloudFormation.

# Utilitários Batch na modernização do AWS mainframe

Os aplicativos de mainframe geralmente usam programas utilitários em lote para executar funções específicas, como classificar dados, transferir arquivos usando FTP, carregar dados em bancos de dados como DB2, descarregar dados de bancos de dados e assim por diante.

Ao migrar seus aplicativos para a modernização do AWS mainframe, você precisa de utilitários de substituição funcionalmente equivalentes que possam realizar as mesmas tarefas que os usados no mainframe. Alguns desses utilitários podem já estar disponíveis como parte dos mecanismos de tempo de execução da modernização do AWS mainframe, mas estamos fornecendo os seguintes utilitários substitutos:

- M2SFTP - permite a transferência segura de arquivos usando o protocolo SFTP.
- M2WAIT - espera por um período de tempo especificado antes de continuar com a próxima etapa em um trabalho em lotes.
- TXT2PDF: converte arquivos de texto em formato PDF.
- M2DFUTIL: que fornece funções de backup, restauração, exclusão e cópia em conjuntos de dados, semelhantes ao suporte fornecido pelo utilitário ADRDSSU de mainframe.
- M2RUNCMD: permite executar comandos, scripts e chamadas de sistema do Micro Focus diretamente da JCL.

Desenvolvemos esses utilitários em lote com base no feedback dos clientes e os projetamos para fornecer a mesma funcionalidade dos utilitários de mainframe. O objetivo é fazer com que sua transição do mainframe para a modernização do AWS mainframe seja a mais tranquila possível.

Tópicos

- Utilitário em lote M2WAIT
- Utilitário em lote TXT2PDF
- Utilitário em lote M2DFUTIL
- Utilitário em lote M2RUNCMD

## Localização binário

Esses utilitários estão pré-instalados nos produtos Micro Focus Enterprise Developer (ED) e Micro Focus Enterprise Server (ES). Você pode encontrá-los no seguinte local para todas as variantes de ED e ES:

- Linux: `/opt/aws/m2/microfocus/utilities/64bit`
- Windows (32 bits): `C:\AWS\M2\MicroFocus\Utilities\32bit`
- Windows (32 bits): `C:\AWS\M2\MicroFocus\Utilities\64bit`

## Utilitário em lote M2SFTP

O M2SFTP é um programa utilitário JCL projetado para realizar transferências seguras de arquivos entre sistemas usando o Secure File Transfer Protocol (SFTP). O programa usa o cliente Putty SFTP,`psftp`, para realizar as transferências reais de arquivos. O programa funciona de forma semelhante a um programa utilitário de FTP de mainframe e usa autenticação de usuário e senha.

> **Note**
>
> A autenticação de chave pública não é compatível.

Para converter seus JCLs FTP de mainframe para usar SFTP, mude `PGM=FTP` para `PGM=M2SFTP`.

Tópicos

- Plataformas compatíveis
- Instalar as dependências
- Configurar o M2SFTP para AWS a modernização gerenciada do mainframe
- Configurar o M2SFTP para o tempo de execução da modernização do AWS mainframe no Amazon EC2 (incluindo 2.0) AppStream

## Plataformas compatíveis

Você pode usar o M2SFTP em qualquer uma das seguintes plataformas:

- AWS Modernização de mainframe gerenciada pela Micro Focus
- Micro Focus Runtime (no Amazon EC2)
- Todas as variantes dos produtos Micro Focus Enterprise Developer (ED) e Micro Focus Enterprise Server (ES).

## Instalar as dependências

Para instalar o cliente Putty SFTP no Windows

- Faça o download do cliente PuTTY SFTP e instale-o.

Para instalar o cliente Putty SFTP no Linux:

- Execute o seguinte comando para instalar o cliente SFTP do Putty:

```
sudo yum -y install putty
```

## Configurar o M2SFTP para AWS a modernização gerenciada do mainframe

Se seus aplicativos migrados estiverem sendo executados no AWS Mainframe Modernization Managed, você precisará configurar o M2SFTP da seguinte forma.

- Defina as variáveis de ambiente apropriadas do Micro Focus Enterprise Server para MFFTP. Aqui estão alguns exemplos:
  - `MFFTP_TEMP_DIR`
  - `MFFTP_SENDEOL`

- `MFFTP_TIME`
- `MFFTP_ABEND`

Você pode definir quantas ou quantas dessas variáveis quiser. Você pode configurá-los em seu JCL usando a `ENVAR DD` instrução. Para obter mais informações sobre essas variáveis, consulte Variáveis de controle do MFFTP na documentação da Micro Focus.

Para testar sua configuração, consulte Exemplo de JCLs.

## Configurar o M2SFTP para o tempo de execução da modernização do AWS mainframe no Amazon EC2 (incluindo 2.0) AppStream

Se seus aplicativos migrados estiverem sendo executados em tempo de execução de modernização de AWS mainframe no Amazon EC2, configure o M2SFTP da seguinte forma.

1. Altere o caminho do programa Micro Focus JES para incluir a localização binária dos utilitários em lote. Se você precisar especificar vários caminhos, use dois pontos (`:`) para separar caminhos no Linux e ponto e vírgula (`;`) no Windows.

   - Linux: `/opt/aws/m2/microfocus/utilities/64bit`
   - Windows (32 bits): `C:\AWS\M2\MicroFocus\Utilities\32bit`
   - Windows (64 bits): `C:\AWS\M2\MicroFocus\Utilities\64bit`

2. Defina as variáveis de ambiente apropriadas do Micro Focus Enterprise Server para MFFTP. Aqui estão alguns exemplos:

   - `MFFTP_TEMP_DIR`
   - `MFFTP_SENDEOL`
   - `MFFTP_TIME`
   - MFFTP_ABEND

   Você pode definir quantas ou quantas dessas variáveis quiser. Você pode configurá-los em seu JCL usando a `ENVAR DD` instrução. Para obter mais informações sobre essas variáveis, consulte Variáveis de controle do MFFTP na documentação da Micro Focus.

Para testar sua configuração, consulte Exemplo de JCLs.

## Exemplo de JCLs

Para testar a instalação, você pode usar um dos seguintes arquivos de exemplo do JCL.

M2SFTP1.jcl

Este JCL mostra como chamar o M2SFTP para enviar um arquivo para um servidor SFTP remoto. Observe as variáveis de ambiente definidas na `ENVVAR DD` instrução.

```
//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

Este JCL mostra como chamar o M2SFTP para receber um arquivo de um servidor SFTP remoto. Observe as variáveis de ambiente definidas na `ENVVAR DD` declaração.

```
//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**
>
> É altamente recomendável armazenar as credenciais de FTP em um arquivo NETRC e restringir o acesso somente a usuários autorizados.

## Referência de comando do cliente Putty SFTP (PSFTP)

O cliente PSFTP não suporta todos os comandos de FTP. A lista a seguir mostra todos os comandos que o PSFTP suporta.

| Command | Descrição |
| --- | --- |
| ! | Executar um comando local |
| tchau | Conclua sua sessão de SFTP |
| cd ~ | Altere seu diretório de trabalho remoto |
| chmod | Alterar permissões e modos de arquivo |
| feche | Conclua sua sessão de SFTP, mas não saia do PSFTP |
| del | Excluir arquivos no servidor remoto |
| dir | Listar arquivos remotos |
| exit | Conclua sua sessão de SFTP |
| get | Baixe um arquivo do servidor para sua máquina local |
| ajuda | Dê ajuda |
| lcd | Alterar diretório de trabalho local |
| lpwd | Imprimir diretório de trabalho local |
| ls | Listar arquivos remotos |

| Command | Descrição |
| --- | --- |
| ímã | Baixe vários arquivos de uma só vez |
| mkdir | Crie diretórios no servidor remoto |
| entrada | Faça upload de vários arquivos de uma só vez |
| mv | Mover ou renomear arquivo (s) no servidor remoto |
| aberto | Conecte-se a um host |
| put | Upload um arquivo da sua máquina local para o servidor |
| PWD | Imprima seu diretório de trabalho remoto |
| sair | Conclua sua sessão de SFTP |
| arrepender | Continue baixando arquivos |
| arrancar | Mover ou renomear arquivo (s) no servidor remoto |
| reputação | Continuar carregando arquivos |
| /rm | Excluir arquivos no servidor remoto |
| rmdir | Remover diretórios no servidor remoto |

## Próximas etapas

Para carregar e baixar arquivos no Amazon Simple Storage Service usando SFTP, você pode usar o M2SFTP em conjunto com o AWS Transfer Family, conforme descrito nas postagens do blog a seguir.

- Usando diretórios lógicos AWS SFTP para criar um serviço simples de distribuição de dados
- Ativar a autenticação por senha para AWS Transfer for SFTP usar AWS Secrets Manager

# Utilitário em lote M2WAIT

O M2WAIT é um programa utilitário de mainframe que permite introduzir um período de espera em seus scripts JCL especificando uma duração de tempo em segundos, minutos ou horas. Você pode chamar o M2WAIT diretamente do JCL passando o tempo que deseja esperar como parâmetro de entrada. Internamente, o programa M2WAIT chama o módulo fornecido pela Micro Focus `C$SLEEP` para aguardar um tempo especificado.

> **Note**
>
> Você pode usar aliases da Micro Focus para substituir o que você tem em seus scripts JCL. Para obter mais informações, consulte JES Alias na documentação da Micro Focus.

Tópicos



## Plataformas compatíveis

Você pode usar o M2WAIT em qualquer uma das seguintes plataformas:

- AWS Modernização de mainframe gerenciada pela Micro Focus
- Micro Focus Runtime (no Amazon EC2)
- Todas as variantes dos produtos Micro Focus Enterprise Developer (ED) e Micro Focus Enterprise Server (ES).

## Configurar o M2WAIT para a modernização gerenciada do AWS mainframe

Se seus aplicativos migrados estiverem sendo executados no AWS Mainframe Modernization Managed, você precisará configurar o M2WAIT da seguinte forma.

- Use o programa M2WAIT em seu JCL passando o parâmetro de entrada conforme mostrado em. Amostra de ACL

## Configure o M2WAIT para o tempo de AWS execução da modernização do mainframe no Amazon EC2 (incluindo 2.0) AppStream

Se seus aplicativos migrados estiverem sendo executados em tempo de execução de modernização de AWS mainframe no Amazon EC2, configure o M2WAIT da seguinte forma.

1. Altere o caminho do programa Micro Focus JES para incluir a localização binária dos utilitários em lote. Se você precisar especificar vários caminhos, use dois pontos (`:`) para separar caminhos no Linux e ponto e vírgula (`;`) no Windows.
   - Linux: `/opt/aws/m2/microfocus/utilities/64bit`
   - Windows (32 bits): `C:\AWS\M2\MicroFocus\Utilities\32bit`
   - Windows (64 bits): `C:\AWS\M2\MicroFocus\Utilities\64bit`
2. Use o programa M2WAIT em seu JCL passando o parâmetro de entrada conforme mostrado em. Amostra de ACL

## Amostra de ACL

Para testar a instalação, você pode usar o `M2WAIT1.jcl` programa.

Este exemplo de JCL mostra como chamar o M2WAIT e passá-lo por várias durações diferentes.

```
//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=*
//*
//
```

# Utilitário em lote TXT2PDF

O TXT2PDF é um programa utilitário de mainframe comumente usado para converter um arquivo de texto em um arquivo PDF. Esse utilitário usa o mesmo código-fonte do TXT2PDF (z/OS freeware). Nós o modificamos para ser executado no ambiente de execução de modernização de AWS mainframe da Micro Focus.

Tópicos



## Plataformas compatíveis

Você pode usar o TXT2PDF em qualquer uma das seguintes plataformas:

- AWS Modernização de mainframe gerenciada pela Micro Focus
- Micro Focus Runtime (no Amazon EC2)
- Todas as variantes dos produtos Micro Focus Enterprise Developer (ED) e Micro Focus Enterprise Server (ES).

## Configurar o TXT2PDF para a modernização gerenciada do mainframe AWS

Se seus aplicativos migrados estiverem sendo executados no AWS Mainframe Modernization Managed, configure o TXT2PDF da seguinte forma.

- Crie uma biblioteca REXX EXEC chamada `AWS.M2.REXX.EXEC`. Baixe esses módulos REXX e copie-os para a biblioteca.
  - `TXT2PDF.rex`- TXT2PDF z/OS freeware (modificado)
  - `TXT2PDFD.rex`- TXT2PDF z/OS freeware (não modificado)
  - `TXT2PDFX.rex`- TXT2PDF z/OS freeware (modificado)
  - `M2GETOS.rex`- Para verificar o tipo de sistema operacional (Windows ou Linux)

Para testar sua configuração, consulte Amostra de ACL.

## Configurar o TXT2PDF para o tempo de execução da modernização do AWS mainframe no Amazon EC2 (incluindo 2.0) AppStream

Se seus aplicativos migrados estiverem sendo executados em tempo de execução de modernização de AWS mainframe no Amazon EC2, configure o TXT2PDF da seguinte forma.

1. Defina a variável `MFREXX_CHARSET` de ambiente Micro Focus com o valor apropriado, como “`A`" para dados ASCII.

   > **Important**
   >
   > Inserir o valor errado pode causar problemas de conversão de dados (de EBCDIC para ASCII), tornando o PDF resultante ilegível ou inoperável. Recomendamos que `MFREXX_CHARSET` a configuração corresponda`MF_CHARSET`.

2. Altere o caminho do programa Micro Focus JES para incluir a localização binária dos utilitários em lote. Se você precisar especificar vários caminhos, use dois pontos (`:`) para separar caminhos no Linux e ponto e vírgula (`;`) no Windows.
   - Linux: `/opt/aws/m2/microfocus/utilities/64bit`
   - Windows (32 bits): `C:\AWS\M2\MicroFocus\Utilities\32bit`
   - Windows (64 bits): `C:\AWS\M2\MicroFocus\Utilities\64bit`

3. Crie uma biblioteca REXX EXEC chamada `AWS.M2.REXX.EXEC`. Baixe esses módulos REXX e copie-os para a biblioteca.

   - `TXT2PDF.rex`- TXT2PDF z/OS freeware (modificado)
   - `TXT2PDFD.rex`- TXT2PDF z/OS freeware (não modificado)
   - `TXT2PDFX.rex`- TXT2PDF z/OS freeware (modificado)
   - `M2GETOS.rex`- Para verificar o tipo de sistema operacional (Windows ou Linux)

Para testar sua configuração, consulte Amostra de ACL.

## Amostra de ACL

Para testar a instalação, você pode usar um dos seguintes arquivos JCL de exemplo.

Txt2pdf1.jcl

Esse arquivo JCL de amostra usa um nome DD para a conversão 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

Esse exemplo de JCL usa um nome DSN para a conversão 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=*
//*
//
```

## Modificações

Para que o programa TXT2PDF seja executado no ambiente de execução de modernização de AWS mainframe da Micro Focus, fizemos as seguintes alterações:

- Alterações no código-fonte para garantir a compatibilidade com o tempo de execução do Micro Focus REXX
- Alterações para garantir que o programa possa ser executado nos sistemas operacionais Windows e Linux
- Modificações para suportar o tempo de execução EBCDIC e ASCII

## Referências

Referências e código-fonte do TXT2PDF:

- Conversor de texto para PDF
- Ferramentas gratuitas de TCP/IP e e-mail do z/OS

- Guia de referência do usuário do TXT2PDF

# Utilitário em lote M2DFUTIL

O M2DFUTIL é um programa utilitário em JCL que fornece funções de backup, restauração, exclusão e cópia em conjuntos de dados, semelhantes ao suporte fornecido pelo utilitário ADRDSSU de mainframe. Esse programa retém muitos dos parâmetros SYSIN do ADRDSSU, o que simplifica o processo de migração para esse novo utilitário.

Tópicos



## Plataformas compatíveis

Você pode usar o M2DFUTIL em qualquer uma das seguintes plataformas:

- Micro Focus ES no Windows (64 bits e 32 bits)
- Micro Focus ES no Linux (64 bits)

## Requisitos da plataforma

O M2DFUTIL depende da chamada de um script para realizar um teste de expressão regular. No Windows, você deve instalar o Windows Services for Linux (WSL) para que esse script seja executado.

## Suporte futuro planejado

Os recursos que atualmente não estão disponíveis no utilitário ADRDSSU de mainframe, mas que estão no escopo futuro, incluem:

- M2 gerenciado
- VSAM
- Suporte a COPY para a renomeação de nomes de arquivos
- Suporte a RENAME para RESTORE
- INCLUDE e EXCLUDE vários
- Cláusula BY para subseleção por DSORG, CREDT, EXPDT
- Cláusula MWAIT para tentar novamente falhas de enfileiramento
- Suporte ao armazenamento do S3 para DUMP/RESTORE

### Locais dos ativos

O módulo de carregamento desse utilitário é chamado `M2DFUTIL.so` no Linux e `M2DFUTIL.dll` no Windows. Esse módulo de carregamento pode ser encontrado em um dos seguintes locais:

- Linux: `/opt/aws/m2/microfocus/utilities/64bit`
- Windows (32 bits): `C:\AWS\M2\MicroFocus\Utilities\32bit`
- Windows (32 bits): `C:\AWS\M2\MicroFocus\Utilities\64bit`

O script usado para testes de expressão regular é chamado `compare.sh`. O script pode ser encontrado em um dos seguintes locais:

- Linux: `/opt/aws/m2/microfocus/utilities/scripts`
- Windows (32 bits): `C:\AWS\M2\MicroFocus\Utilities\scripts`

## Configure o tempo de execução do M2DFUTIL ou da modernização do AWS mainframe no Amazon EC2 (incluindo 2.0) AppStream

Configure sua região do Enterprise Server com o seguinte:

- Adicione as seguintes variáveis em [ES-Environment]:
  - `M2DFUTILS_BASE_LOC`: o local padrão para a saída DUMP
  - `M2DFUTILS_SCRIPTPATH`: o local do script `compare.sh` documentado em Locais dos ativos
  - `M2DFUTILS_VERBOSE`: [VERBOSO ou NORMAL]. Isso controla o nível de detalhe na saída `SYSPRINT`

- Verificar se o caminho do módulo de carregamento foi adicionado à configuração `JES \Configuration\JES Program Path`
- Verifique se os scripts no diretório de utilitários têm permissões de execução. É possível adicionar uma permissão de execução usando o comando `chmod + x <script name>`, no ambiente Linux

## Sintaxe geral

### DUMP

Fornece a possibilidade de copiar arquivos do local catalogado atual para um local de backup. No momento, esse local deve ser um sistema de arquivos.

### Processar

DUMP executará o seguinte:

1. Crie o diretório do local de destino.
2. Catalogue o diretório do local de destino como membro do PDS.
3. Determine os arquivos a serem incluídos processando o parâmetro INCLUDE.
4. Desmarque os arquivos incluídos processando o parâmetro EXCLUDE.
5. Determine se os arquivos que estão sendo despejados devem ser EXCLUÍDOS.
6. Coloque em fila os arquivos a serem processados.
7. Copie os arquivos.
8. Exporte as informações do DCB catalogadas dos arquivos copiados para um arquivo secundário no local de destino para auxiliar nas futuras operações de RESTORE.

### Sintaxe

```
DUMP
TARGET ( TARGET LOCATION  )    -
INCLUDE ( DSN. )
[ EXCLUDE ( DSN ) ]
[ CANCEL | IGNORE ]
[ DELETE ]
```

### Parâmetros necessários

Veja abaixo os parâmetros necessários para DUMP:

- `SYSPRINT DD NAME`: para conter as informações adicionais de registro
- `TARGET`: local de destino Pode ser um ou outro.
  - Caminho completo do local de despejo
  - Nome do subdiretório criado no local definido na variável M2DFUTILS_BASE_LOC
- `INCLUDE`: um único DSNAME nomeado ou uma string de pesquisa de DSN do mainframe válida
- `EXCLUDE`: um único DSNAME nomeado ou uma string de pesquisa de DSN do mainframe válida

### Parâmetros opcionais

- CANCEL: cancele se ocorrer algum erro. Os arquivos que foram processados serão retidos
- (Padrão) IGNORE: ignore qualquer erro e processo até o final
- DELETE: se nenhum erro ENQ ocorrer, o arquivo será excluído e não será catalogado

## DELETE

Oferece a possibilidade de excluir e não catalogar arquivos em massa. Os arquivos não têm backup.

### Processar

DELETE executará o seguinte:

1. Determine os arquivos a serem incluídos processando o parâmetro INCLUDE.
2. Desmarque os arquivos incluídos processando o parâmetro EXCLUDE.
3. Coloque em fila os arquivos a serem processados. Definindo a disposição como OLD, DELETE, KEEP.

### Sintaxe

```
DELETE
INCLUDE ( DSN )
[ EXCLUDE ( DSN ) ]
[ CANCEL | IGNORE ]
[ DELETE ]
```

Parâmetros necessários

Veja abaixo os parâmetros necessários para DELETE:

- `SYSPRINT DD NAME`: para conter as informações adicionais de registro
- `INCLUDE`: um único DSNAME nomeado ou uma string de pesquisa de DSN do mainframe válida
- `EXCLUDE`: um único DSNAME nomeado ou uma string de pesquisa de DSN do mainframe válida

Parâmetros opcionais

- CANCEL: cancele se ocorrer algum erro. Os arquivos que foram processados serão retidos
- (Padrão) IGNORE: ignore qualquer erro e processo até o final

RESTORE

Fornece a possibilidade de restaurar arquivos que tiveram backup previamente DUMP. Os arquivos são restaurados no local catalogado original, a menos que RENAME seja usado para alterar o DSNAME restaurado.

Processar

RESTORE executará o seguinte:

1. Valide o diretório do local de origem.
2. Determine os arquivos a serem incluídos processando o arquivo de exportação do catálogo.
3. Desmarque os arquivos incluídos processando o parâmetro EXCLUDE.
4. Coloque em fila os arquivos a serem processados.
5. Arquivos de catálogo que não são catalogados com base em suas informações de exportação.
6. Se um arquivo já estiver catalogado e as informações do catálogo de exportação forem as mesmas, RESTORE substituirá o conjunto de dados catalogado se a opção REPLACE estiver definida.

Sintaxe

```
RESTORE
SOURCE ( TARGET LOCATION )
INCLUDE ( DSN )
```

```
[ EXCLUDE ( DSN ) ]
[ CANCEL | IGNORE ]
[ REPLACE]
```

Parâmetros necessários

Veja abaixo os parâmetros necessários para RESTORE:

- `SYSPRINT DD NAME`: para conter as informações adicionais de registro
- `SOURCE`: local de origem. Pode ser um ou outro.
  - Caminho completo do local de despejo
  - Nome do subdiretório criado no local definido na variável M2DFUTILS_BASE_LOC
- `INCLUDE`: um único DSNAME nomeado ou uma string de pesquisa de DSN do mainframe válida
- `EXCLUDE`: um único DSNAME nomeado ou uma string de pesquisa de DSN do mainframe válida

Parâmetros opcionais

- CANCEL: cancele se houver algum erro. Arquivos processados retidos
- (Padrão) IGNORE: ignore qualquer erro e processo até o final
- REPLACE: se o arquivo que está sendo restaurado já estiver catalogado e os registros do catálogo forem os mesmos, substitua o arquivo catalogado

## Exemplo de JCLs

Trabalho de DUMP

Esse trabalho criará um subdiretório chamado `TESTDUMP`. Esse é o local de backup padrão especificado pela variável M2DFUTILS_BASE_LOC. Ele criará uma biblioteca PDS para esse backup chamada `M2DFUTILS.TESTDUMP`. Os dados do catálogo exportado são armazenados em um arquivo sequencial de linhas no diretório de backup chamado `CATDUMP.DAT`. Todos os arquivos selecionados serão copiados para esse diretório de backup.

```
//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
/*
//
```

### Trabalho DELETE

Esse trabalho excluirá todos os arquivos do catálogo que correspondam ao parâmetro 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
/*
 //
```

### Trabalho RESTORE

Esse trabalho restaurará os arquivos que correspondem ao parâmetro INCLUDE do local de backup de TESTDUMP. Os arquivos catalogados serão substituídos se o arquivo catalogado for o mesmo da exportação de CATDUMP e a opção REPLACE for especificada.

```
//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
/*
//
```

# Utilitário em lote M2RUNCMD

Você pode usar o M2RUNCMD, um programa utilitário em lote, para executar comandos, scripts e chamadas de sistema do Micro Focus diretamente da JCL em vez de executá-los em um terminal ou prompt de comando. A saída dos comandos é registrada no log de spool do trabalho em lote.

Tópicos



## Plataformas compatíveis

É possível usar o M2RUNCMD nas seguintes plataformas:

- Micro Focus Runtime (no Amazon EC2)
- Todas as variantes dos produtos Micro Focus Enterprise Developer (ED) e Micro Focus Enterprise Server (ES).

## Configure o M2RUNCMD para o tempo de execução de modernização de AWS mainframe no Amazon EC2 (incluindo 2.0) AppStream

Se seus aplicativos migrados estiverem sendo executados em tempo de execução de modernização de AWS mainframe no Amazon EC2, configure o M2RUNCMD da seguinte forma.

- Altere o caminho do programa Micro Focus JES para incluir a localização binária dos utilitários em lote. Se precisar especificar vários caminhos, use dois-pontos (:) para separar caminhos no Linux e ponto e vírgula (;) no Windows.
  - Linux: `/opt/aws/m2/microfocus/utilities/64bit`
  - Windows (32 bits): `C:\AWS\M2\MicroFocus\Utilities\32bit`
  - Windows (64 bits): `C:\AWS\M2\MicroFocus\Utilities\64bit`

## Exemplo de JCLs

Para testar a instalação, use um dos seguintes arquivos JCL de exemplo:

RUNSCRL1.jcl

Esse exemplo de JCL cria um script e o executa. A primeira etapa cria um script chamado `/tmp/TEST_SCRIPT.sh` e com conteúdo de dados no stream de SYSUT1. A segunda etapa define a permissão de execução e executa o script criado na primeira etapa. Também é possível optar por realizar somente a segunda etapa para executar comandos já existentes do Micro Focus e do sistema.

```
//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

A saída do comando ou script executado é gravada no log de SYS0UT. Para cada comando executado, ele exibe o comando, a saída e o código de retorno.

```
************ 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

Este exemplo de JCL usa RUNCMD para executar vários comandos.

```
//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 Modernização do mainframe e replicação de dados com precisão

AWS A modernização do mainframe oferece uma variedade de Amazon Machine Images (AMIs). Essas AMIs facilitam o provisionamento rápido de instâncias do Amazon EC2, criando um ambiente personalizado para replicação de dados de sistemas de mainframe para o uso do Precisely. AWS Este guia fornece as etapas necessárias para acessar e usar essas AMIs.

## Pré-requisitos

- Certifique-se de ter acesso de administrador a uma AWS conta na qual você possa criar instâncias do Amazon EC2.
- Verifique se o serviço de modernização do AWS mainframe está disponível na região em que você planeja criar as instâncias do Amazon EC2. Veja a lista de serviços da AWS disponíveis por região.
- Identifique a Amazon Virtual Private Cloud (Amazon VPC) na qual as instâncias do Amazon EC2 serão criadas.
- Ao criar instâncias do Amazon EC2 em uma Amazon VPC, certifique-se de que a tabela de rotas associada tenha um gateway da Internet ou um gateway NAT.

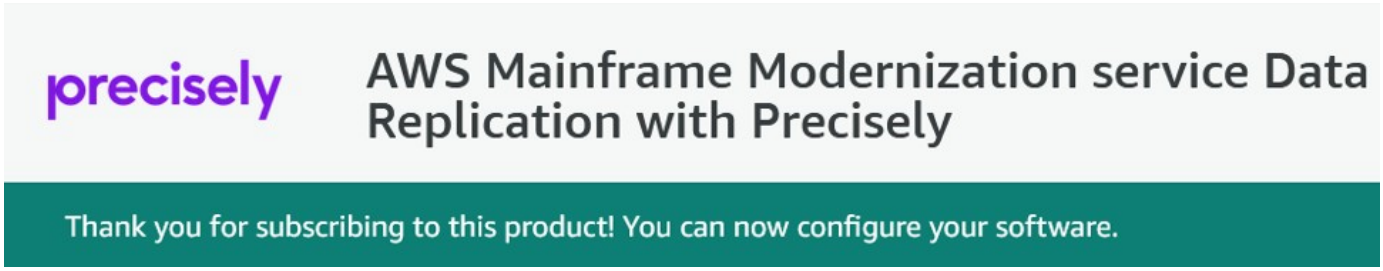
> **Note**
>
> A replicação de dados com êxito exige que a instância do EC2 da AWS tenha acesso de comunicação ao AWS Marketplace. Se houver um problema de conectividade com o AWS Marketplace, haverá falha no processo de replicação.

## Inscrever-se para receber a imagem de máquina da Amazon

Após a inscrição em um produto do AWS Marketplace, você pode executar uma instância usando a AMI do produto.

1. Faça login no AWS Management Console e abra o AWS Marketplace console em https://console.aws.amazon.com/marketplace.

2. Escolha Manage subscriptions (Gerenciar assinaturas).
3. Copie e cole o seguinte link na barra de endereço do navegador: https://aws.amazon.com/marketplace/pp/prodview-en3xrbgzbs3dk
4. Escolha Continue to Subscribe (Continuar para assinar).
5. Se os Termos e Condições forem aceitáveis, escolha Aceitar Termos. A assinatura pode levar alguns minutos para ser processada.
6. Aguarde até que a mensagem de agradecimento seja exibida, conforme mostrado abaixo. Essa mensagem confirma que você se inscreveu com êxito no produto.




7. No painel de navegação esquerdo, escolha Gerenciar assinaturas. Essa visualização mostra todas as assinaturas nas quais você se inscreveu.

## Inicie a replicação de dados AWS da modernização do mainframe com o Precisely

1. Abra o AWS Marketplace console em https://console.aws.amazon.com/marketplace.
2. No painel de navegação esquerdo, escolha Gerenciar assinaturas.
3. Encontre a AMI que você deseja iniciar e escolha Executar nova instância.
4. Em Região, selecione a região listada como permitida.
5. Escolha Continuar para executar por meio do EC2. Essa ação direcionará você para o console do Amazon EC2.
6. Insira um nome para o servidor.
7. Selecione um tipo de instância que corresponda aos requisitos de desempenho e custo do seu projeto. O ponto de partida sugerido para o tamanho da instância é `c5.2xLarge`.
8. Escolha um par de chaves existente ou crie um e salve-o. Para obter mais informações sobre pares de chave, consulte Pares de chaves do Amazon EC2 e instâncias do Linux no Manual do usuário do Amazon EC2 para instâncias do Linux.
9. Edite as configurações de rede e escolha a VPC na lista de permissões e a sub-rede apropriada.

10. Escolha um grupo de segurança existente ou crie um. Além de permitir o acesso SSH (por padrão na porta 22), para a replicação de dados com uma instância EC2 do servidor da Precisely, é comum permitir o tráfego TCP para sua porta padrão 2626.
11. Configure o armazenamento para a instância do Amazon EC2.
12. Revise o resumo e envie a Executar instância. Para que a execução tenha êxito, o tipo de instância deve ser válido. Se houver falha na execução, escolha Editar configuração da instância e escolha um tipo de instância diferente.
13. Depois de ver a mensagem de êxito, escolha Conectar-se à Instância.
14. Abra o console do Amazon EC2 em https://console.aws.amazon.com/ec2/.
15. No painel de navegação esquerdo, no menu Instâncias, escolha Instâncias.
16. No painel principal, verifique o status da sua instância.

## Criar uma política do IAM

Para operar com sucesso as instâncias EC2 de modernização de AWS mainframe implantadas por meio de nossa AWS Marketplace listagem, você deve configurar uma função e uma política do IAM. Essa configuração específica do IAM não é opcional; ela autoriza suas instâncias do Amazon EC2 a interagir com o serviço. AWS Marketplace A função e a política do IAM permitem que a modernização do AWS mainframe registre com precisão os dados de uso, o que é essencial para um faturamento preciso. A não implementação dessa configuração pode levar a falha nas tentativas de replicação de dados e a interrupções operacionais.

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação à esquerda, escolha Policies (Políticas).
3. Se essa for a primeira vez que você escolhe Políticas, a página Bem-vindo às políticas gerenciadas será exibida. Escolha Começar.
4. Na parte superior da página, escolha Criar política.
5. Na seção Editor de políticas, escolha a opção JSON.
6. Insira a seguinte política JSON:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": ["aws-marketplace:MeterUsage"],
```

```
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

## Criar um perfil do IAM

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, escolha Roles e Create role.
3. Na seção Tipo de entidade confiável, escolha Serviço da AWS .
4. Na seção Caso de uso, em Serviço ou caso de uso, escolha Amazon EC2.
5. Escolha Próximo.
6. Na lista de políticas, selecione Gerenciada pelo cliente no menu suspenso Filtrar por tipo e insira o nome da política que você criou. Marque a caixa de seleção ao lado do nome da política.
7. Escolha Próximo.
8. Insira um nome e, opcionalmente, uma descrição para o perfil.
9. Revise a política de confiança e as permissões e escolha Criar perfil.

## Anexar o perfil do IAM à instância do Amazon EC2

1. Abra o console do Amazon EC2 em https://console.aws.amazon.com/ec2/.
2. No painel de navegação, escolha Instances (Instâncias).
3. Selecione sua instância Amazon EC2.
4. No menu Ações, escolha Segurança e, depois, escolha Modificar o perfil do IAM.
5. Selecione o perfil do IAM a ser anexado à instância e escolha Atualizar perfil do IAM.

# Integração do Charon

## Introdução ao Charon-SSP

Em 1987, a Sun Microsystems lançou o processador SPARC V7, um processador RISC de 32 bits. O SPARC V8 foi lançado em 1990, uma revisão do SPARC V7 original, com a inclusão mais notável de instruções de divisão e multiplicação de hardware. Os processadores SPARC V8 formaram a base para vários servidores e estações de trabalho, como o SPARCstation 5, 10 e 20. Em 1993, depois do SPARC V8, foi lançado o processador SPARC V9 de 64 bits. Isso também se tornou a base para vários servidores e estações de trabalho, como o Enterprise 250 e 450.

Devido à obsolescência do hardware e à falta de peças de reposição ou de peças recondicionadas, o software e os sistemas desenvolvidos para essas estações de trabalho e servidores mais antigos baseados no SPARC ficaram mais difíceis de manter. Para preencher a necessidade contínua de determinados sistemas end-of-life baseados em SPARC, a Stromasys S.A. desenvolveu a linha Charon-SSP de produtos emuladores SPARC. Os produtos a seguir são substitutos de máquinas virtuais baseados em software para os sistemas SPARC de hardware nativo especificados. A seguir encontra-se uma visão geral das famílias de hardware emulado.

O Charon-SSP/4M emula o seguinte hardware SPARC:

- Família Sun-4m (representada pelo Sun SPARCstation 20): originalmente, uma variante do multiprocessador Sun-4, baseada no barramento do módulo do processador MBus apresentado na série SPARCServer 600MP. Posteriormente, a arquitetura Sun-4m também incluiu sistemas uniprocessadores que não sejam MBus, como o SPAR Cstation 5, utilizando processadores da arquitetura SPARC V8. Compatível a partir do SunOS 4.1.2 e do Solaris 2.1 até o Solaris 9. A compatibilidade com o SPARCServer 600MP foi descontinuada após o Solaris 2.5.1.

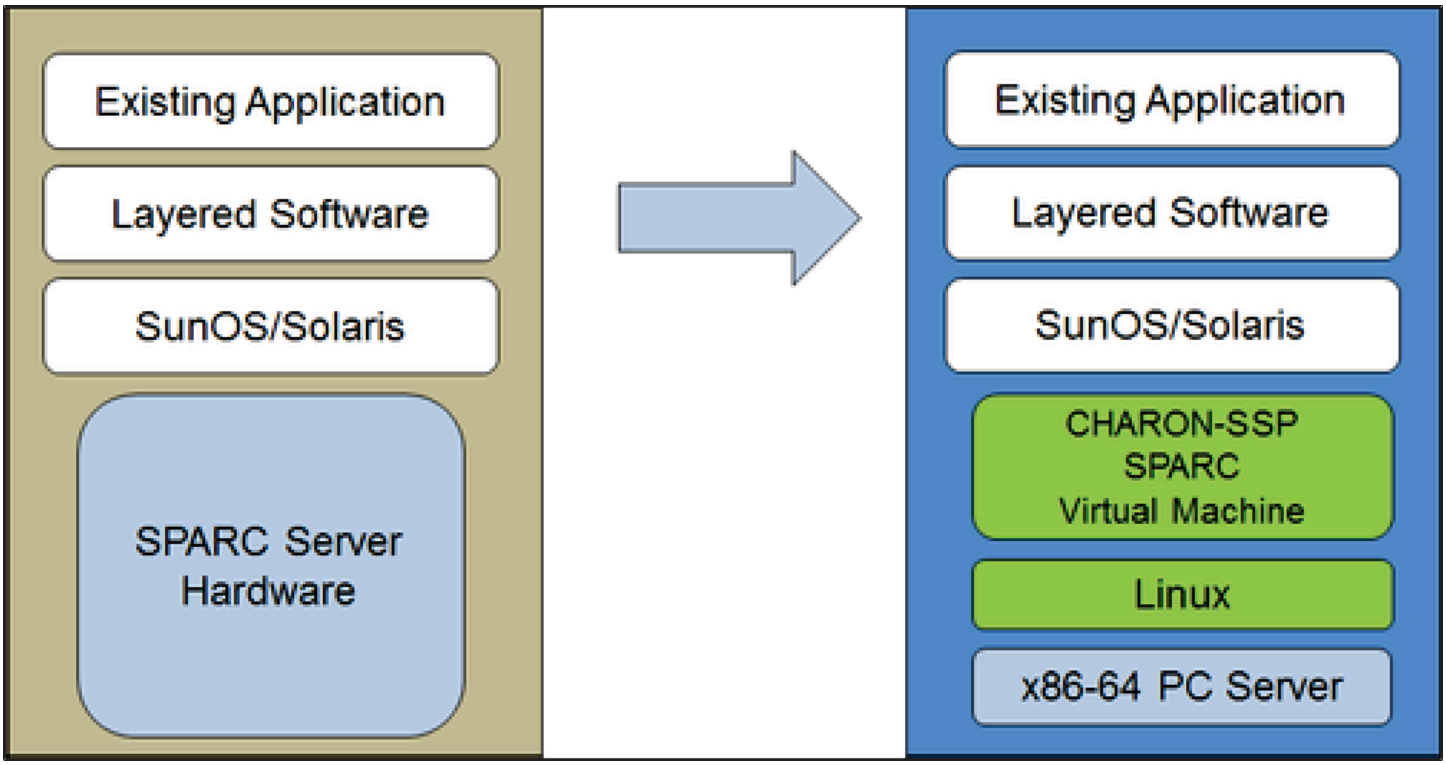
O Charon-SSP/4U(+) emula o seguinte hardware SPARC:

- Família Sun-4u (representada pelo Sun Enterprise 450): (U para UltraSPARC): essa variante apresentou a arquitetura do processador SPARC V9 de 64 bits e a interconexão do processador UPA usada pela primeira vez na série Sun Ultra. Compatível com versões de 32 bits do Solaris a partir da versão 2.5.1. A primeira versão do Solaris de 64 bits para o Sun-4u foi o Solaris 7. A compatibilidade com o UltraSPARC I foi descontinuada após o Solaris 9. O Solaris 10 é compatível com implementações do Sun-4u do UltraSP ARC II ao UltraSPARC IV.

O Charon-SSP/4V(+) emula o seguinte hardware SPARC:

- Família Sun-4v (representada pelos SPARC T2 e T4): essa variação adicionou a virtualização do processador hipervisor ao Sun-4u, apresentada no processador multicore Ultra SPARC T1. O hardware selecionado era compatível com a versão 10 do Solaris a partir da versão 3/05 HW2 (a maioria dos modelos, incluindo o hardware emulado pelo Charon-SSP, exige versões mais recentes do Solaris 10). Várias versões do Solaris 11 também são compatíveis.

A imagem a seguir mostra o conceito básico da migração de hardware físico para um emulador.




As máquinas virtuais Charon-SSP permitem que os usuários de computadores baseados em Sun e Oracle SPARC substituam seu hardware nativo de uma forma que exija pouca ou nenhuma alteração na configuração original do sistema. Isso significa que você pode continuar executando suas aplicações e dados sem a necessidade de alternar ou migrar para outra plataforma. O software Charon-SSP funciona em sistemas Intel de 64 bits básicos, garantindo a proteção contínua do seu investimento.

O Charon-SSP/4U+ é compatível com as mesmas plataformas SPARC virtuais que o Charon-SSP/4U, e o Charon-SSP/4V+ é o mesmo que o Charon-SSP/4V. No entanto, as versões 4U+ e 4V + aproveitam as vantagens da tecnologia de virtualização assistida por hardware VTx/EPT da Intel e AMD-v/NPT da AMD em CPUs modernas para oferecer melhor desempenho da CPU virtual. O

Charon-SSP/4U+ e o Charon-SSP/4V+ exigem CPUs compatíveis com VT-x/EPT ou AMD-v/NPT e devem ser instalados em um sistema host dedicado. A execução dessas variantes de produto em uma VM (por exemplo, no VMware) não é compatível.

> **Note**
>
> Se você planeja executar o Charon-SSP/4U+ ou o 4V+ em um ambiente de nuvem, entre em contato com a Stromasys ou com um VAR da Stromasys para conversar sobre seus requisitos.

## Sistemas operacionais convidados compatíveis

As máquinas virtuais Charon-SSP/4M são compatíveis com as seguintes versões do sistema operacional convidado:

- SunOS 4.1.3 a 4.1.4
- Solaris 2.3 a Solaris 9

As máquinas virtuais Charon-SSP/4U(+) são compatíveis com as seguintes versões do sistema operacional convidado:

- Solaris 2.5.1 a Solaris 10

As máquinas virtuais Charon-SSP/4V(+) são compatíveis com as seguintes versões do sistema operacional convidado:

- Solaris 10 (começando com a atualização 4, 08/07) e Solaris 11.1 a Solaris 11.4

Para Charon-SSP/4V(+), observe o seguinte:

- Para o SPARC T4 emulado, as versões compatíveis do Solaris 10 são: Oracle Solaris 10 1/13, Oracle Solaris 10 8/11 e Solaris 10 9/10, ou Solaris 10 10/09 com o conjunto de patches Oracle Solaris 10 8/11.
- O modelo SPARC T4 emulado é um pré-requisito para executar o Solaris 11.4 no emulador.
- Não há compatibilidade com zonas de kernel do Solaris.

# Pré-requisitos da instância de nuvem do Charon-SSP

Ao selecionar um tipo ou formato de instância, selecione o hardware virtual que será usado para a instância do host do Charon-SSP na nuvem. Portanto, a seleção de um tipo ou formato de instância determina as características do hardware do host virtual do Charon-SSP (por exemplo, quantos núcleos de CPU e quanta memória seu sistema host virtual do Charon terá).

> **Note**
>
> Se você usar uma imagem do marketplace do Charon-SSP para executar a instância, todos os requisitos do sistema operacional do host do Linux serão atendidos.

Os requisitos mínimos de hardware estão descritos abaixo.

Pontos importantes em relação às diretrizes de dimensionamento:

- As diretrizes de dimensionamento abaixo, especialmente em relação ao número de núcleos de CPU e memória do host, mostram os requisitos mínimos. Cada situação de implantação deve ser revisada e o dimensionamento real do host deve ser adaptado conforme necessário. Por exemplo, o número de núcleos de CPU disponíveis para E/S deve ser aumentado se as aplicações convidadas produzirem uma alta carga de E/S. Além disso, um sistema com muitas CPUs emuladas normalmente pode criar uma carga de E/S maior e, portanto, talvez seja necessário aumentar o número de núcleos de CPU disponíveis para E/S. Em um ambiente hyper-threading, para obter o melhor desempenho, o número de núcleos de CPU (ou seja, CPUs real/físicas) deve ser suficiente para atender aos requisitos de CPU dos emuladores ativos, evitando assim que threads de alta workload compartilhem um núcleo físico de CPU.
- A alocação de núcleos de CPU para CPUs emuladas e de núcleos de CPU para processamento de E/S é determinada pela configuração. Consulte Configuração da CPU no Guia geral do usuário do Charon-SSP para obter mais informações sobre isso e a alocação padrão de núcleos de CPU para processamento de E/S.

> **Informações gerais importantes**
>
> - Para facilitar a transferência rápida dos dados do emulador de uma instância de nuvem para outra, é altamente recomendável armazenar todos os dados relevantes do emulador

em um volume de disco separado que possa ser facilmente desanexado da instância antiga e anexado a uma nova instância.

- Certifique-se de dimensionar a instância corretamente desde o início (verifique os requisitos mínimos abaixo). A licença do Charon-SSP para o Charon-SSP AL é criada quando a instância é executada pela primeira vez. Alterar posteriormente para outro tamanho/tipo de instância e, assim, alterar o número de núcleos de CPU invalidará a licença e, assim, impedirá que as instâncias do Charon sejam executadas (será necessária uma nova instância). Se estiver planejando usar a instância do Charon-SSP AL no modo AutoVE, certifique-se de incluir as informações do servidor AutoVE antes da primeira execução, caso contrário, os servidores de licenças públicas serão usados. A licença do Charon-SSP VE é criada com base na impressão digital obtida no servidor de licenças. Se o servidor de licenças for executado diretamente no host do emulador e o host do emulador exigir posteriormente, por exemplo, uma alteração no número de núcleos de CPU, a licença será invalidada (será necessária uma nova licença e, possivelmente, uma nova instância).

## Pré-requisitos da instância

Requisitos gerais de CPU: o Charon-SSP é compatível com processadores modernos de arquitetura x86-64 baseados em instâncias do Amazon EC2.

Requisitos mínimos para Charon-SSP:

- Número mínimo de núcleos de CPU do sistema host:
  - Pelo menos um núcleo de CPU para o sistema operacional host, além de:
  - Para cada sistema SPARC emulado:
    - Um núcleo de CPU para cada CPU emulada da instância, além de:
    - Pelo menos um núcleo de CPU adicional para processamento de E/S (pelo menos dois, se a otimização JIT do servidor for usada). Consulte a seção Configuração da CPU mencionada acima para ver as opções de configuração. Por padrão, o Charon atribuirá 1/3 (mínimo 1; arredondado para baixo) do número de CPUs visíveis para o host do Charon ao processamento de E/S.
- Requisitos mínimos de memória:
  - 4 GB ou mais de RAM para o sistema operacional host Linux. Os requisitos reais podem ser maiores e dependerão dos requisitos dos serviços não emuladores executados no host Linux.

A recomendação anterior de pelo menos 2 GB de RAM para o host Linux ainda será válida para muitos sistemas, mas os crescentes requisitos do sistema operacional e das aplicações do Linux levaram à recomendação atualizada para novas instalações. Além de:

  - Para cada sistema SPARC emulado:
    - A memória configurada da instância emulada, além de:
    - 2 GB de RAM (6 GB de RAM se o servidor JIT for usado) para permitir a otimização de DIT, requisitos do emulador, buffers de tempo de execução, SMP e emulação gráfica.
- Se o hyper-threading estiver habilitado em CPUs x86-64 modernas, dois threads poderão ser executados em um núcleo físico de CPU, fornecendo duas CPUs lógicas para o sistema operacional host. Se possível, desative o hyper-threading no host Charon-SSP. No entanto, isso geralmente não é possível em ambientes VMware e de nuvem, ou não está claro se o hyper-threading é usado ou não. A opção de hyper-threading do Charon-SSP permite que o Charon-SSP se adapte a esses ambientes. Consulte a seção Configuração da CPU no Guia geral do usuário do Charon-SSP mencionado acima para obter informações detalhadas da configuração. Observação: para obter o melhor desempenho, os threads Charon-SSP não devem compartilhar um núcleo físico de CPU. Núcleos físicos suficientes devem estar disponíveis no sistema host para satisfazer os requisitos do(s) emulador(es) configurado(s).
- Uma ou mais interfaces de rede, dependendo dos requisitos do cliente.
- O Charon-SSP/4U+ e o Charon-SSP/4V+ devem ser executados no hardware físico compatível com Intel VT-x/EPT ou AMD-v/NPT (instâncias baremetal) e, portanto, não podem ser executados em todos os ambientes de nuvem. Verifique a documentação do seu provedor de nuvem para saber sobre a disponibilidade desse hardware. Além disso, observe os seguintes pontos:
  - O Charon-SSP/4U+ e o Charon-SSP/4V+ só estão disponíveis ao usar um kernel do Linux compatível com a Stromasys.
  - Se você precisar desse tipo de hardware SPARC emulado, entre em contato com a Stromasys ou com seu VAR da Stromasys para discutir seus requisitos em detalhes.

# Criação e configuração de uma instância de AWS nuvem para Charon (nova GUI)

Esta seção reflete o AWS Management Console na primavera de 2022. Se você ainda usa o console antigo, consulte o Apêndice do guia de introdução do AWS Charon-SSP.

# Pré-requisitos gerais

Essa descrição mostra a configuração básica de uma instância do Linux na AWS. Ela não lista os pré-requisitos específicos. No entanto, dependendo do seu caso de uso, considere os seguintes pré-requisitos:

- Conta e AWS Marketplace assinaturas da Amazon
  - Para configurar uma instância Linux no AWS, você precisa de uma AWS conta com acesso de administrador.
  - Identifique a AWS região na qual você planeja executar sua instância. Certifique-se de que os serviços da AWS que você planeja usar estejam disponíveis nessa região. Consulte Serviços da AWS por região.
  - Identifique a VPC e a sub-rede na qual você planeja executar a instância.
  - Se a instância exigir acesso à Internet, certifique-se de que a tabela de rotas associada à sua VPC tenha um gateway da Internet. Se a instância exigir acesso da VPN à sua rede local, verifique se há um gateway de VPN disponível. A configuração exata da sua VPC e de suas sub-redes dependerá do design da rede e dos requisitos da aplicação.
  - Para assinar um AWS Marketplace serviço específico, escolha Assinaturas do AWS Marketplace no AWS Management Console e, em seguida, escolha Gerenciar assinaturas.
  - Procure o serviço que você planeja usar e inscreva-se nele. Depois de inscrever-se com êxito, você encontrará a assinatura na seção Gerenciar assinaturas. A partir daí, você poderá executar diretamente uma nova instância.
- Os pré-requisitos de hardware e de software da instância serão diferentes dependendo do uso planejado da instância:
  - Opção 1: a instância deve ser usada como um sistema host do emulador Charon:
    - Consulte as seções de pré-requisitos de hardware e de software do Guia do usuário e/ou do Guia de conceitos básicos do seu produto Charon para determinar os pré-requisitos exatos de hardware e de software que devem ser atendidos pela instância do Linux. A imagem que você usa para executar a instância e o tipo de instância escolhido determinam o software e o hardware da sua instância de nuvem.
    - É necessária uma licença de produto Charon para executar sistemas herdados emulados. Consulte as informações de licenciamento na documentação do seu produto Charon ou entre em contato com seu representante da Stromasys ou com o VAR da Stromasys para obter informações adicionais.
  - Opção 2: a instância deve ser usada como um servidor de licenças VE dedicado:

    - Consulte o Guia do Servidor de Licenças VE para obter os pré-requisitos detalhados.
- Alguns sistemas operacionais antigos que podem ser executados nos sistemas emulados fornecidos pelos produtos emuladores Charon exigem uma licença do fornecedor original do sistema operacional. O usuário é responsável por quaisquer obrigações de licenciamento relacionadas ao sistema operacional antigo e deve fornecer as licenças apropriadas.

## Usando o AWS Management Console para iniciar uma nova instância

Como criar uma instância

1. Faça login no AWS Management Console e abra o console do Amazon EC2 em https://console.aws.amazon.com/ec2/.
2. Escolha Iniciar instância.
3. Insira um nome para a instância.
4. Selecione uma AMI. Uma AMI é uma imagem pré-empacotada usada para executar instâncias de nuvem. Ela inclui o sistema operacional e o software aplicável à aplicação. A escolha da AMI depende de como você planeja usar a instância:

    - Se a instância for usada como um sistema host do emulador Charon, várias opções de AMI serão possíveis:
        - Instalar o sistema host Charon a partir de uma imagem pré-empacotada do marketplace do Charon: elas contêm o sistema operacional subjacente e o software Charon pré-instalado.
            - Verifique com seu representante da Stromasys quais opções estão atualmente disponíveis no marketplace de seus provedores de nuvem.
            - Dependendo do provedor de nuvem e dos planos de lançamento do produto da Stromasys, pode haver duas variantes:
                - Licenciamento automático (AL) para uso com um servidor de licenças público operado pela Stromasys ou com um servidor de licenças AutoVE privado operado pelo cliente
                - Ambiente virtual (VE) para uso com um servidor de licenças VE privado operado pelo cliente
        - Instalar o sistema host Charon usando uma instalação convencional do emulador Charon com os pacotes RPM de instalação do emulador Charon para Linux:

    - Escolha uma AMI do Linux de uma distribuição compatível com o produto e a versão selecionados do Charon. Consulte o guia do usuário do seu produto no site de documentação da Stromasys.
  - Se a instância for usada como um servidor de licenças VE dedicado, consulte o Guia do servidor de licenças VE na documentação de licenciamento para ver os requisitos da instância do Linux.

  Depois de decidir qual AMI é necessária, selecione uma AMI de produto Linux ou Charon correspondente. Caso não veja a AMI de que precisa, escolha Pesquisar mais AMIs. Escolha a AMI do Linux que corresponda à forma como você planeja usar a instância. Uma das seguintes opções é possível:

  - Uma imagem pré-embalada do marketplace Charon VE. O nome da AMI incluirá a string “ve”.
  - Uma imagem pré-embalada do marketplace Charon AL para licenciamento automático ou AutoVE.
  - Uma versão do Linux compatível com a instalação de um produto RPM.
  - Uma versão do Linux compatível com o servidor de licenças VE.
5. Selecione um tipo de instância. O Amazon EC2 oferece tipos de instância com combinações variadas de CPU, memória, armazenamento e capacidade de rede. Selecione um tipo de instância que corresponda aos requisitos do produto Charon que você deseja usar. Algumas imagens do marketplace têm uma seleção restrita de tipos de instância.
6. Selecione um par de chaves existente ou crie um e salve-o. Se você selecionar um par de chaves existente, verifique se você tem a chave privada correspondente. Caso contrário, você não poderá se conectar à instância.

   > **Note**
   >
   > Se o seu sistema de gerenciamento for compatível com RHEL 9.x, Rocky Linux 9.x e Oracle Linux 9.x, use o tipo de chave SSH ECDSA ou ED25519. Esses tipos permitem que você se conecte a esses sistemas Linux do host Charon usando um túnel SSH sem precisar alterar as configurações padrão da política de criptografia no host Charon para configurações menos seguras. Por exemplo, isso é importante para o Charon-SSP Manager. Consulte Usando políticas criptográficas em todo o sistema na documentação da Red Hat.

7. Na seção Configurações de rede, escolha Editar. Escolha as configurações que correspondam ao seu ambiente.

   - Especifique uma VPC.
   - Especifique uma sub-rede existente ou crie uma.
   - Habilite ou desabilite a atribuição automática de um endereço IP público à interface principal. A atribuição automática só é possível se a instância tiver uma única interface de rede.
   - Atribua um grupo de segurança personalizado novo ou existente. O grupo de segurança deve permitir que pelo menos o SSH acesse a instância. Todas as portas exigidas pelas aplicações que você planeja executar na instância também devem ser permitidas. É possível modificar o grupo de segurança a qualquer momento depois que a instância for criada.

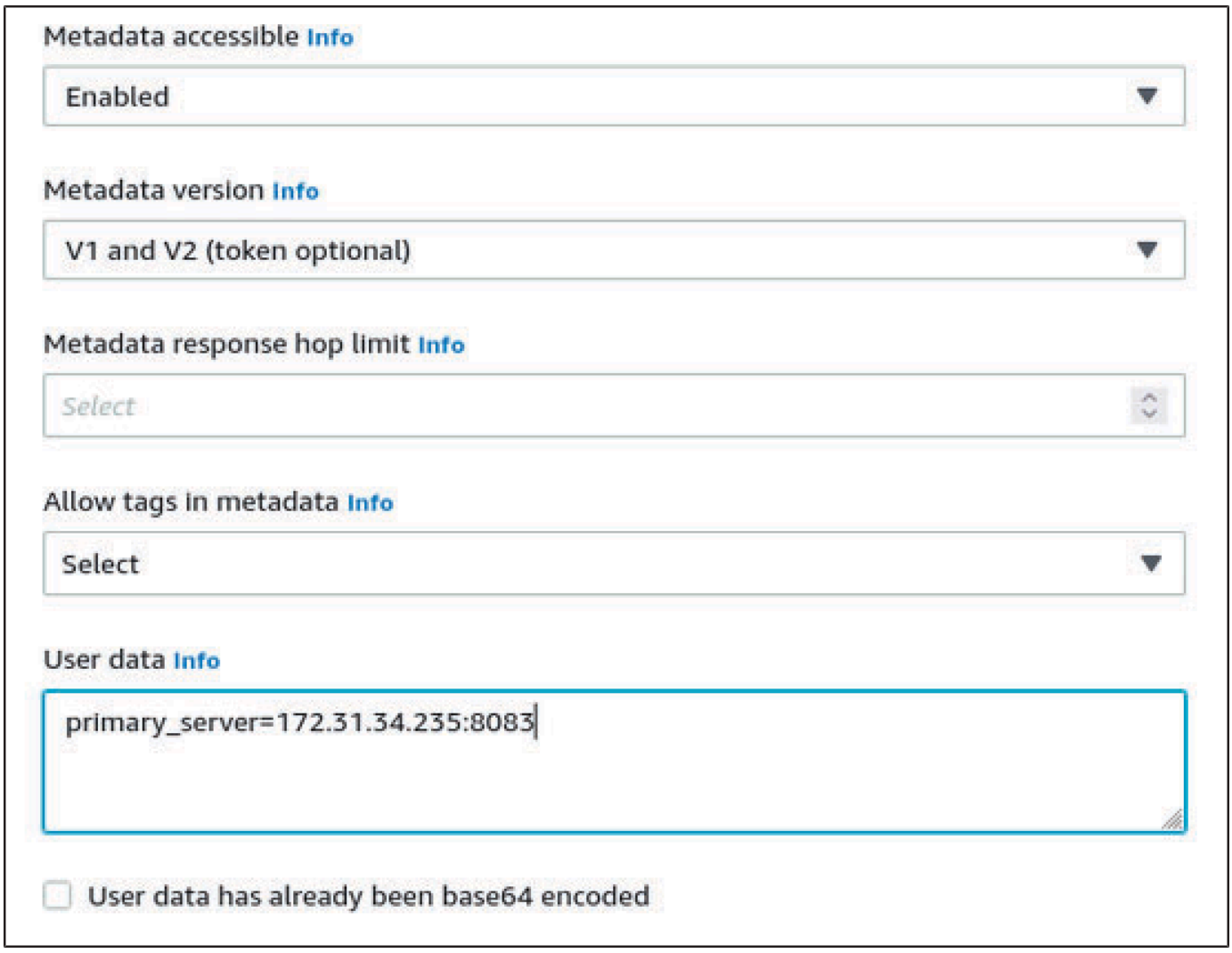
8. Na seção Armazenamento, para o volume raiz (o disco do sistema), escolha um tamanho adequado ao seu ambiente. O tamanho mínimo de disco do sistema recomendado para o sistema Linux é de 30 GiB. Para fornecer espaço para contêineres de disco virtual e outros requisitos de armazenamento, é possível adicionar mais armazenamento agora ou depois de executar a instância. Mas o tamanho do disco do sistema deve cobrir os requisitos do sistema Linux, incluindo todas as aplicações e utilitários que você planeja instalar.

   > **Note**
   >
   > Recomendamos que você crie volumes de armazenamento separados para os dados da aplicação Charon (por exemplo, imagens de disco). Se necessário, você poderá migrar esses volumes para outra instância posteriormente.

9. Expanda Detalhes avançados e marque a caixa de seleção Especificar opções de CPU. Três que têm maior probabilidade de serem úteis para um ambiente de emulador Charon são mostradas na imagem a seguir como exemplos.


10. Para um sistema de servidor de licenças VE com uma versão anterior à 1.1.23, você deverá atribuir um perfil do IAM necessário à instância. Ele deve ser um perfil que permita a ação `ListUsers`. Para atribuir um perfil, na seção expandida Detalhes avançados, selecione um perfil em Perfil da instância do IAM ou escolha Criar um perfil do IAM. Para obter mais informações, consulte Funções do IAM para Amazon EC2.
11. Se sua instância for baseada em uma AWS Marketplace imagem Charon AL e você planeja usar os servidores de licenças públicas operados pela Stromasys, você deve adicionar as informações correspondentes à configuração da instância antes de executar a instância.

    Insira as informações do servidor de licenças AutoVE, conforme mostrado na imagem a seguir.




As seguintes opções de configuração de dados do usuários são válidas:

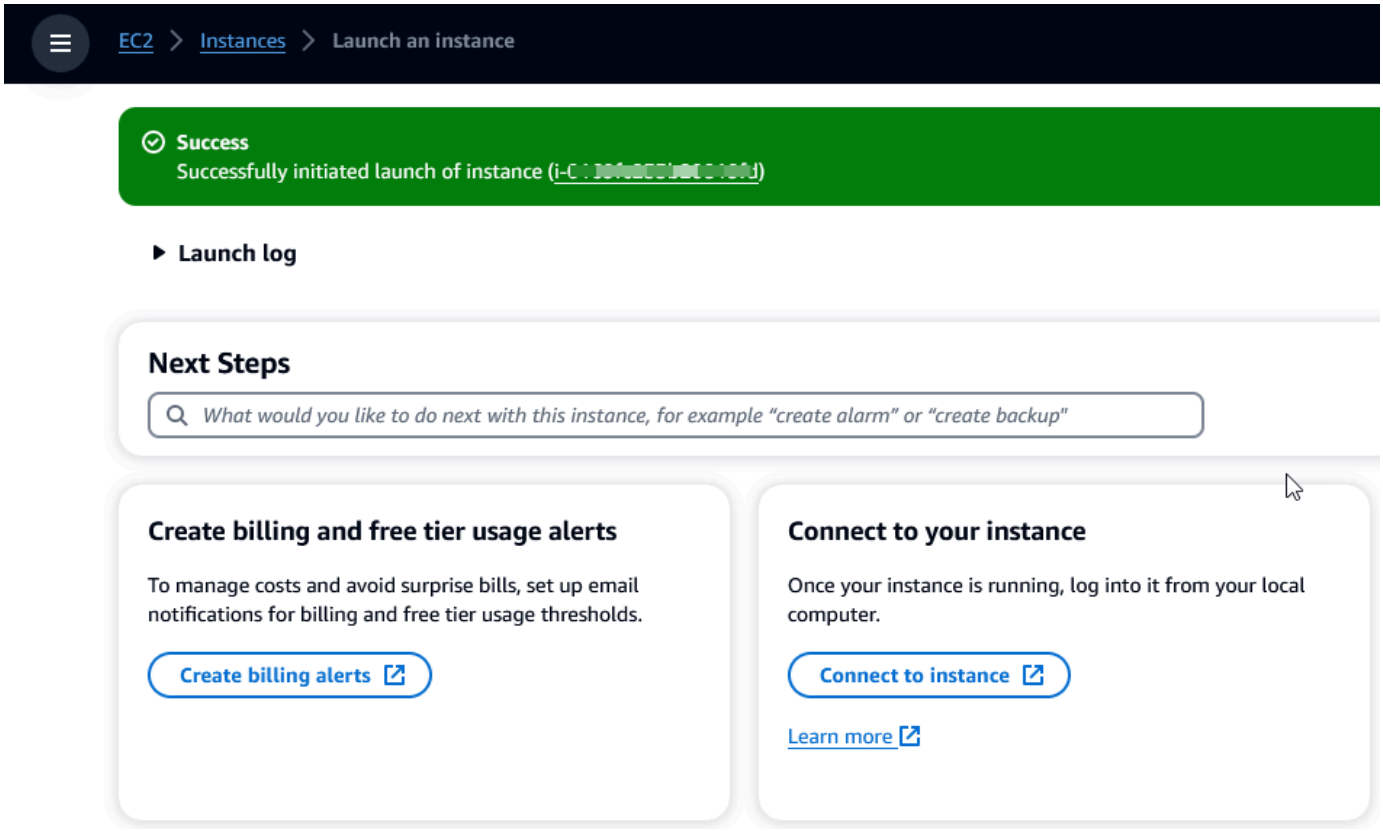
- **`primary_server=`***`<ip-address>`*`[:`*`<port>`*`]`
- **`backup_server=`***`<ip-address>`*`[:`*`<port>`*`]`

Em que

- <ip-address> significa o endereço IP do servidor primário e do servidor de backup, conforme aplicável.
- <port> significa uma porta TCP não padrão usada para se comunicar com o servidor de licenças (padrão: TCP/8083).

> **Note**
>
> Pelo menos um servidor de licenças deve ser configurado na execução inicial para ativar o modo AutoVE. Caso contrário, a instância será vinculada a um dos servidores de licenças públicas operados pela Stromasys.

12. No painel Resumol, escolha Executar instância. Depois de um tempo, você receberá a seguinte mensagem de êxito:




13. No canto inferior direito da tela, escolha Visualizar todas as instâncias.
14. Para ver os detalhes da instância, marque a caixa de seleção à esquerda da linha que representa a instância na tabela Instâncias. Os detalhes da instância serão exibidos na metade inferior da tela. Para obter informações sobre como conectar-se à sua instância, consulte Conectar no Manual do usuário do Amazon EC2 para instâncias do Linux.

# AWS Modernização do mainframe e reformulação da plataforma com a NTT DATA

AWS A modernização do mainframe oferece uma variedade de Amazon Machine Images (AMIs). Essas AMIs facilitam o rápido provisionamento de instâncias do Amazon EC2, criando um ambiente personalizado para rehospedar e reformular aplicativos de mainframe usando o NTT Data. AWS Este guia fornece as etapas necessárias para acessar e usar essas AMIs.

## Pré-requisitos

- Certifique-se de ter acesso de administrador a uma AWS conta na qual você possa criar instâncias do Amazon EC2.
- Verifique se o serviço de modernização do AWS mainframe está disponível na região em que você planeja criar as instâncias do Amazon EC2. Veja a lista de serviços da AWS disponíveis por região.
- Identifique a Amazon VPC em que você deseja criar as instâncias do Amazon EC2.

## Inscrever-se para receber a imagem de máquina da Amazon

Após a inscrição em um produto do AWS Marketplace, você pode executar uma instância usando a AMI do produto.

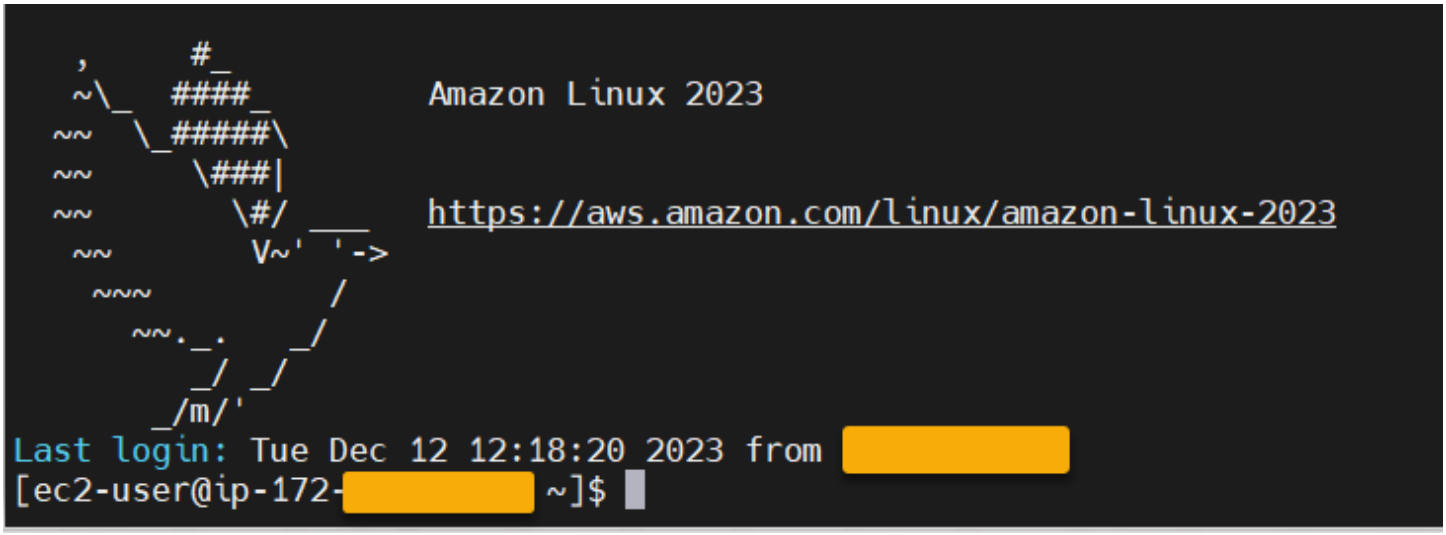
1. Faça login no AWS Management Console e abra o AWS Marketplace console em https://console.aws.amazon.com/marketplace.
2. Escolha Manage subscriptions (Gerenciar assinaturas).
3. Copie e cole o seguinte link na barra de endereço do navegador: https://aws.amazon.com/marketplace/pp/prodview-eg227ymldsnx2
4. Escolha Continue to Subscribe (Continuar para assinar).
5. Se os Termos e Condições forem aceitáveis, escolha Aceitar Termos. A assinatura pode levar alguns minutos para ser processada.
6. Aguarde até que uma mensagem de agradecimento seja exibida. Essa mensagem confirma que você se inscreveu com êxito no produto.
7. No painel de navegação esquerdo, escolha Gerenciar assinaturas. Essa visualização mostra todas as suas assinaturas.

# Inicie a replataforma de modernização de AWS mainframe com a instância NTT DATA

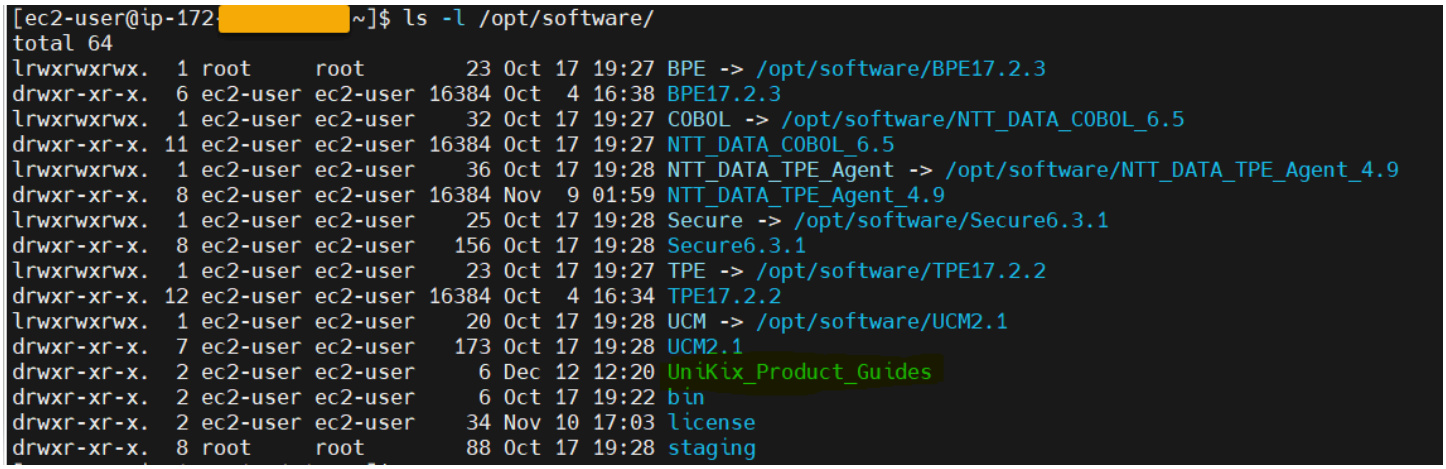
1. Abra o AWS Marketplace console em https://console.aws.amazon.com/marketplace.
2. No painel de navegação esquerdo, escolha Gerenciar assinaturas.
3. Encontre a AMI que você deseja iniciar e escolha Executar nova instância.
4. Em Região, selecione a região listada como permitida.
5. Escolha Continuar para executar por meio do EC2. Essa ação direcionará você para o console do Amazon EC2.
6. Insira um nome para o servidor.
7. Selecione um tipo de instância que corresponda aos requisitos de desempenho e custo do seu projeto. O ponto de partida sugerido para o tamanho da instância é `c5.2xLarge`.
8. Escolha um par de chaves existente ou crie um e salve-o. Para obter mais informações sobre pares de chave, consulte Pares de chaves do Amazon EC2 e instâncias do Linux no Manual do usuário do Amazon EC2 para instâncias do Linux.
9. Edite as configurações de rede e escolha a VPC na lista de permissões e a sub-rede apropriada.
10. Escolha um grupo de segurança existente ou crie um. Se for uma instância do Amazon EC2 do Enterprise Server, é normal permitir o tráfego TCP para as portas 86 e 10086 para administrar a configuração do Micro Focus.
11. Configure o armazenamento para a instância do Amazon EC2.
12. Revise o resumo e envie a Executar instância. Para que a execução tenha êxito, o tipo de instância deve ser válido. Se houver falha na execução, escolha Editar configuração da instância e escolha um tipo de instância diferente.
13. Depois de ver a mensagem de êxito, escolha Conectar-se à Instância.
14. Abra o console do Amazon EC2 em https://console.aws.amazon.com/ec2/.
15. No painel de navegação esquerdo, no menu Instâncias, escolha Instâncias.
16. No painel principal, verifique o status da sua instância.

# Conceitos básicos da NTT Data

Depois de provisionar a instância do Amazon EC2, faça o SSH nela com o nome de usuário `ec2-user`. A tela será exibida como a imagem a seguir.




Abaixo da pasta `/opt/software/`, há uma pasta chamada `UniKix_Product_Guides`, conforme mostrado na imagem a seguir.




A pasta `UniKix_Product_Guides` inclui a documentação dos seguintes componentes que estão instalados nessa instância do Amazon EC2:

- NTT DATA TPE
- NTT DATA BPE
- NTT DATA Enterprise COBOL
- NTT DATA Secure UniKix
- Gerente UniKix central da NTT DATA

A pasta `software` que aparece na imagem anterior tem os binários dos componentes listados acima.

Depois de validar com sucesso a instância do Amazon EC2, comece a AWS usar a Mainframe Modernization Replatform com a NTT DATA seguindo a documentação da NTT Data.

# Aplicativos no AWS Mainframe Modernization

Se você é novo no AWS Mainframe Modernization, consulte os tópicos a seguir para começar:

- O que é modernização AWS do mainframe?
- Configurando a modernização AWS do mainframe
- Tutorial: Tempo de execução gerenciado para AWS Blu Age
- Tutorial: Tempo de execução gerenciado para Micro Focus

Um aplicativo no AWS Mainframe Modernization contém uma carga de trabalho de mainframe migrada. O aplicativo é análogo a uma carga de trabalho no mainframe e está associado a um ambiente de tempo de execução. Você pode adicionar arquivos em lotes e conjuntos de dados aos aplicativos e monitorar os aplicativos à medida que são executados. Você cria aplicativos no AWS Mainframe Modernization para cada carga de trabalho que você migra. Ao criar um aplicativo AWS Mainframe Modernization, você especifica o mecanismo no qual o aplicativo é executado ao criá-lo. Escolha AWS Blu Age se estiver usando o padrão de refatoração automatizado e escolha Micro Focus se estiver usando o padrão de replataforma.

Tópicos

- Criar um aplicativo do AWS Mainframe Modernization
- Implemente um aplicativo AWS Mainframe Modernization
- Atualize um aplicativo do AWS Mainframe Modernization
- Excluir um aplicativo AWS Mainframe Modernization de um ambiente
- Excluir um aplicativo do AWS Mainframe Modernization
- Envie trabalhos em lotes para aplicativos do AWS Mainframe Modernization
- Importe conjuntos de dados para aplicativos do AWS Mainframe Modernization
- Gerencie transações para aplicativos do AWS Mainframe Modernization
- Crie AWS recursos para um aplicativo migrado
- Configurar o aplicativo gerenciado
- AWS Referência de definição de aplicativo de modernização de mainframe
- Referência de definição do conjunto de dados do AWS Mainframe Modernization

# Criar um aplicativo do AWS Mainframe Modernization

Use o console do AWS Mainframe Modernization para criar um aplicativo do AWS Mainframe Modernization.

Estas instruções supõem que você tenha concluído as etapas em Configurando a modernização AWS do mainframe.

## Cria uma aplicação

Para criar um aplicativo.

1. Abra o console do AWS Mainframe Modernization em https://console.aws.amazon.com/m2/.
2. No seletor Região da AWS, escolha a região onde você deseja criar o aplicativo.
3. Na página Aplicativos, escolha Criar aplicativo.
4. Na página Especificar informações básicas, na seção Nome e descrição, insira um nome para o aplicativo.
5. (Opcional) No campo Descrição, digite uma descrição breve para o aplicativo. Essa descrição pode ajudar você e outros usuários a identificar a finalidade do aplicativo.
6. Na seção Tipo de motor, escolha Blu Age para refatoração automatizada ou Micro Focus para reformulação de plataforma.
7. Na seção Chave KMS, escolha Personalizar configurações de criptografia se quiser usar uma AWS KMS chave gerenciada pelo cliente. Para ter mais informações, consulte Criptografia de dados em repouso para o serviço de modernização AWS de mainframe.

   > **Note**
   >
   > Por padrão, o AWS Mainframe Modernization criptografa seus dados com uma chave AWS KMS que o AWS Mainframe Modernization possui e gerencia para você. No entanto, você pode optar por usar uma AWS KMS chave gerenciada pelo cliente.

8. (Opcional) Escolha uma AWS KMS chave por nome ou nome de recurso da Amazon (ARN) ou escolha Criar uma AWS KMS chave para acessar o AWS KMS console e criar uma nova AWS KMS chave.
9. (Opcional) Na seção Tags, escolha Adicionar nova tag para adicionar uma ou mais tags de aplicativo ao seu aplicativo. Uma tag de aplicativo é um rótulo de atributo personalizado que ajuda a organizar e gerenciar AWS os recursos.

10. Escolha Próximo.
11. Na seção Recursos e configurações, use o editor embutido para inserir a definição do aplicativo. Como alternativa, escolha Usar um arquivo JSON de definição de aplicativo em um bucket do Amazon S3 e forneça a localização da definição do aplicativo que você deseja usar. Para obter mais informações, consulte AWS Exemplo de definição do aplicativo Blu Age ou Definição de aplicativo do Micro Focus.
12. Escolha Próximo.
13. Na página Revisar e criar, analise as informações fornecidas e selecione Criar fonte de dados.

# Implemente um aplicativo AWS Mainframe Modernization

Use o console do AWS Mainframe Modernization para implantar uma aplicação do AWS Mainframe Modernization.

Estas instruções supõem que você tenha concluído as etapas em Configurando a modernização AWS do mainframe.

## Implantar uma aplicação

Para executar uma aplicação do AWS Mainframe Modernization, é necessário implantá-la primeiro em um ambiente de tempo de execução. Um aplicativo pode ter mais de uma versão. Cada versão de um aplicativo tem sua própria definição de aplicativo. Para implantar um aplicativo, você deve especificar a versão que deseja implantar.

Você pode implantar somente uma versão de um determinado aplicativo por vez. Se você implantar uma versão de um aplicativo e decidir implantar uma versão diferente, primeiro interrompa o aplicativo se ele estiver em execução.

Para implantar uma aplicação

1. Abra o console do AWS Mainframe Modernization em https://console.aws.amazon.com/m2/.
2. No seletor Região da AWS, escolha a região onde você deseja criar o aplicativo.
3. Na página Aplicativos, escolha o aplicativo que você deseja implementar.
4. Escolha Implantar aplicativo.
5. No Versões disponíveis, escolha a versão que você deseja implantar.
6. Na seção Ambientes, escolha um ambiente de tempo de execução no qual você deseja que seu aplicativo seja executado.

7. Escolha Implantar.

Para implantar uma versão diferente de um aplicativo implantado

1. Abra o console do AWS Mainframe Modernization em https://console.aws.amazon.com/m2/.
2. No seletor Região da AWS, escolha a região onde você deseja criar o aplicativo.
3. Na página Aplicativos, escolha o aplicativo que você deseja implementar.
4. No menu Ações, escolha Parar aplicativo.
5. Depois que o aplicativo for interrompido, escolha Implantar aplicativo.
6. No Versões disponíveis, escolha a versão que você deseja implantar. Na seção Ambientes, o ambiente no qual o aplicativo já está implantado é pré-selecionado.
7. Escolha Implantar.

# Atualize um aplicativo do AWS Mainframe Modernization

Use o console do AWS Mainframe Modernization para atualizar uma aplicativo do AWS Mainframe Modernization.

Estas instruções supõem que você tenha concluído as etapas em Configurando a modernização AWS do mainframe.

## Atualizar uma aplicação

Um aplicativo do AWS Mainframe Modernization pode ter várias versões, cada uma com sua própria definição de aplicativo. Para atualizar um aplicativo, forneça uma nova definição de aplicativo. Isso cria uma nova versão do aplicativo.

Para atualizar uma aplicação

1. Abra o console do AWS Mainframe Modernization em https://console.aws.amazon.com/m2/.
2. No seletor Região da AWS, escolha a região em que o aplicativo que você deseja atualizar foi criado.
3. Na página Aplicativos, escolha o aplicativo que deseja atualizar.
4. Na página de detalhes do aplicativo, na seção Definição atual, escolha Editar para atualizar a definição atual do aplicativo.

5. Na página Atualizar aplicativo, use o editor embutido para atualizar a definição atual do aplicativo.

   Como alternativa, escolha Usar um arquivo JSON de definição de aplicativo em um bucket do Amazon S3 e forneça a localização da definição do aplicativo que você deseja usar. Para obter mais informações, consulte AWS Exemplo de definição do aplicativo Blu Age ou Definição de aplicativo do Micro Focus.
6. Quando terminar de atualizar a definição do aplicativo, escolha Atualizar.

> **Note**
>
> Depois de atualizar o aplicativo, você deve implantá-lo novamente. Para ter mais informações, consulte Implemente um aplicativo AWS Mainframe Modernization.

# Excluir um aplicativo AWS Mainframe Modernization de um ambiente

Você pode excluir um aplicativo do AWS Mainframe Modernization de um ambiente usando o console do AWS Mainframe Modernization.

Estas instruções supõem que você tenha concluído as etapas em Configurando a modernização AWS do mainframe.

## Excluir um aplicativo de um ambiente

Se você precisar excluir um aplicativo AWS Mainframe Modernization e ele estiver em execução, certifique-se de interrompê-lo primeiro. Você pode ver o status do aplicativo na página Aplicativos.

Para excluir um aplicativo de um ambiente

1. Abra o console do AWS Mainframe Modernization em https://console.aws.amazon.com/m2/.
2. No seletor Região da AWS, escolha a região em que o aplicativo que você deseja excluir do ambiente foi criado.
3. Na página Aplicativos, escolha o aplicativo que você deseja excluir do ambiente e, em seguida, escolha Ações.
4. (Opcional) Se o status do aplicativo for `Running`, escolha Interromper aplicativo.

5. Escolha Excluir do ambiente.

   O processo de exclusão iniciará imediatamente.

# Excluir um aplicativo do AWS Mainframe Modernization

Use o console do AWS Mainframe Modernization para excluir um aplicativo do AWS Mainframe Modernization.

Estas instruções supõem que você tenha concluído as etapas em Configurando a modernização AWS do mainframe.

## Deleta o aplicativo

Se você precisar excluir um aplicativo AWS Mainframe Modernization e ele estiver em execução, certifique-se de interrompê-lo primeiro. Você pode ver o status do aplicativo na página Aplicativos.

Como excluir uma aplicação

1. Abra o console do AWS Mainframe Modernization em https://console.aws.amazon.com/m2/.
2. No seletor Região da AWS , escolha a região em que o aplicativo que você deseja excluir foi criado.
3. Na página Aplicativos, escolha o aplicativo que você deseja excluir e, em seguida, escolha Ações.
4. (Opcional) Se o status do aplicativo for `Running`, escolha Interromper aplicativo.
5. Selecione Delete application (Excluir aplicativo).
6. Na janela Excluir aplicativo, insira `delete` para confirmar que você deseja excluir o aplicativo e escolha Excluir.

# Envie trabalhos em lotes para aplicativos do AWS Mainframe Modernization

No AWS Mainframe Modernization, você pode enviar trabalhos em lote para seus aplicativos. Você pode enviar ou cancelar trabalhos em lotes e revisar detalhes sobre execuções de trabalhos em lotes. Cada vez que você envia um trabalho em lotes, o AWS Mainframe Modernization cria uma

execução separada do trabalho em lotes. É possível monitorar a execução do trabalho. Você pode pesquisar trabalhos em lote por nome e fornecer arquivos JCL ou de script para trabalhos em lote.

> **Important**
>
> Se você cancelar um trabalho em lote, isso não excluirá o trabalho. Ele cancela uma execução específica do trabalho em lotes. Os registros do trabalho em lotes permanecem disponíveis para você visualizar os detalhes da execução do trabalho em lotes.

Se o seu trabalho em lote exigir acesso a um ou mais conjuntos de dados, use o console do AWS Mainframe Modernization ou o AWS Command Line Interface (AWS CLI) para importar os conjuntos de dados. Para ter mais informações, consulte Importe conjuntos de dados para aplicativos do AWS Mainframe Modernization.

Estas instruções pressupõem que você tenha concluído as etapas em Configurando a modernização AWS do mainframe e em Criar um aplicativo do AWS Mainframe Modernization.

## Enviar um trabalho em lote

Para enviar um trabalho em lote

1. Abra o console do AWS Mainframe Modernization em https://console.aws.amazon.com/m2/.
2. No seletor Região da AWS, escolha a região onde foi criado o aplicativo para o qual você deseja enviar um trabalho em lote.
3. Na página Aplicativos, escolha o aplicativo para o qual você deseja enviar um trabalho em lotes.

   > **Note**
   >
   > Antes de enviar uma tarefa em lotes para uma aplicação, é necessário implantar a aplicação com sucesso.

4. Na página de detalhes do aplicativo, selecione Trabalhos em lote.
5. Escolha Enviar trabalho.
6. Na seção Selecionar um script, escolha um script. É possível procurar o script que deseja pelo nome do.
7. Escolha Enviar trabalho.

# Importe conjuntos de dados para aplicativos do AWS Mainframe Modernization

Com o AWS Mainframe Modernization, você pode importar conjuntos de dados para usar com seus aplicativos. É possível especificar os conjuntos de dados em um arquivo JSON armazenado em um bucket do Amazon S3, ou os valores de configuração do conjunto de dados separadamente. Depois de importar os conjuntos de dados, você pode revisar os detalhes da tarefa de importação para confirmar se os conjuntos de dados desejados foram importados. Todos os conjuntos de dados catalogados de um aplicativo são listados juntos no console.

Use o console do AWS Mainframe Modernization para atualizar uma aplicativo do AWS Mainframe Modernization.

Estas instruções pressupõem que você tenha concluído as etapas em Configurando a modernização AWS do mainframe e em Criar um aplicativo do AWS Mainframe Modernization.

## Importar um conjunto de dados

Para importar um conjunto de dados

1. Abra o console do AWS Mainframe Modernization em https://console.aws.amazon.com/m2/.
2. No seletor Região da AWS , escolha a região onde foi criado o aplicativo para o qual você deseja importar conjuntos de dados.
3. Na página Aplicativos, escolha o aplicativo para o qual você deseja importar conjuntos de dados.
4. Na página de detalhes do aplicativo, selecione Conjuntos de dados.
5. Escolha Importar.
6. Execute um destes procedimentos:
   - Escolha Usar arquivo JSON de configuração do conjunto de dados em um bucket do Amazon S3 e forneça a localização da configuração do conjunto de dados.
   - Escolha Especificar os valores de configuração do conjunto de dados separadamente com a configuração guiada. Consulte the section called “Referência de definição do conjunto de dados” para obter detalhes específicos da definição.

     Insira o nome, a organização do conjunto de dados (VSAM, GDG, PO, PS), a localização e a localização externa do Amazon S3 e as configurações de parâmetros para cada valor de configuração do conjunto de dados. Na configuração guiada, você também pode escolher Gerar JSON para revisar a configuração JSON a partir de sua entrada.

7. Selecione Enviar.

# Gerencie transações para aplicativos do AWS Mainframe Modernization

Com o AWS Mainframe Modernization, você pode executar um aplicativo, mediante solicitação, ao mesmo tempo que muitos outros usuários que enviam solicitações para executar o mesmo aplicativo usando os mesmos arquivos e programas. Uma única transação consiste em um ou mais programas de aplicativos que realizam o processamento necessário.

Estas instruções pressupõem que você tenha concluído as etapas em Configurando a modernização AWS do mainframe e em Criar um aplicativo do AWS Mainframe Modernization.

## Gerencie transações para aplicativos

Você pode exibir e editar transações para aplicativos.

Para gerenciar transações para aplicativos

1. Abra o console do AWS Mainframe Modernization em https://console.aws.amazon.com/m2/.
2. No seletor Região da AWS, escolha a Região da AWS onde o aplicativo que você deseja executar foi criado.
3. Na página Aplicativos, escolha o aplicativo para o qual você deseja gerenciar transações.
4. Na guia Transações, em Recursos da transação, escolha como você deseja que seus recursos sejam exibidos na lista suspensa. Você pode exibir recursos de acordo com os recursos da transação, grupos, listas ou SITs.

   - Os recursos de transação permitem que você escolha o tipo de recurso de acordo com as definições de arquivo, de transação, de programa ou de fila de dados transitórios.

     > **Note**
     >
     > A transação do AWS Mainframe Modernization Managed oferece suporte a mais tipos de recursos do que esses, mas você não pode editá-los diretamente aqui. Outros recursos devem ser editados externamente e você precisa recriar o aplicativo com entradas de recursos atualizadas.

- Os grupos são uma coleção de recursos de transações. Você também pode escolher grupos que deseja associar ao recurso da transação.
- As listas são coleções ordenadas de grupos. Você pode ver todos os seus recursos e grupos de transações em uma exibição de lista. A lista de inicialização determina quais recursos são carregados quando o servidor é inicializado.
  - Com o mecanismo de refatoração Blu Age, você especifica as listas a serem incluídas na inicialização e não há limite para o número de listas.
  - Com o mecanismo de replataforma da Micro Focus, você pode especificar até quatro listas em um SIT.
- A SIT (Tabela de Inicialização do Sistema) exibe todas as configurações de transação disponíveis. Você pode encontrar SITs de acordo com as propriedades (nome, descrição e listas de inicialização). Você também pode escolher listas para associar ao SIT escolhido.

> **Note**
>
> Os SITs são aplicáveis somente ao mecanismo de replataforma da Micro Focus.

5. Escolha um recurso de transação para exibir todas as informações do recurso. Você também pode visualizar todos os atributos associados ao seu recurso de transação e visualizar ou editar quaisquer atributos adicionais.
6. Se você quiser editar qualquer informação relacionada ao seu recurso de transação atual, escolha Editar.

   a. Na página Editar, você pode adicionar ou alterar a descrição do recurso.

      1. Para recursos de transação exibidos em listas, você também pode adicionar, remover ou reordenar listas conforme desejado.
      2. Para tipos de recursos específicos (definições de arquivos, definições de transações, definições de programas e definições de filas de dados transitórias), você pode editar as propriedades individuais do recurso. Essas propriedades variam de acordo com o mecanismo e o tipo de recurso.

   b. Depois de fazer as alterações desejadas, escolha Salvar alterações.

      Você vê uma mensagem quando o recurso é atualizado com êxito.

# Crie AWS recursos para um aplicativo migrado

Para executar seu aplicativo migrado emAWS, você deve criar alguns AWS recursos com outrosServiços da AWS. Os recursos que você precisa criar incluem o seguinte:

- Um bucket S3 para armazenar o código do aplicativo, a configuração, os arquivos de dados e outros artefatos necessários.
- Um banco de dados Amazon RDS ou Amazon Aurora para armazenar os dados que o aplicativo exige.
- UmAWS KMS key, que é exigido AWS Secrets Manager para criar e armazenar segredos.
- Um segredo do Secrets Manager para manter as credenciais do banco de dados.

> **Note**
>
> Cada aplicativo migrado exige seu próprio conjunto desses recursos. Esse é um conjunto mínimo. Seu aplicativo também pode exigir recursos adicionais, como segredos do Amazon Cognito ou filas do MQ.

## Permissões obrigatórias

Verifique se você tenha as seguintes permissões:

- `s3:CreateBucket`, `s3:PutObject`
- `rds:CreateDBInstance`
- `kms:CreateKey`
- `secretsmanager:CreateSecret`

## Bucket do Amazon S3

Tanto os aplicativos refatorados quanto os reformulados exigem um bucket S3 que você configura da seguinte forma:

```
bucket-name/root-folder-name/application-name
```

nome-do-seu-bucket

Qualquer nome dentro das restrições de nomenclatura do Amazon S3. Recomendamos que você inclua o Região da AWS nome como parte do nome do bucket. Crie o bucket na mesma região em que você planeja implantar o aplicativo migrado.

root-folder-name

Nome necessário para satisfazer as restrições na definição do aplicativo, que você cria como parte do aplicativo AWS Mainframe Modernization. Você pode usar o `root-folder-name` para distinguir entre diferentes versões de um aplicativo, por exemplo, V1 e V2.

application-name

O nome do seu aplicativo migrado, por exemplo, PlanetsDemo ou BankDemo.

## Banco de dados

Tanto os aplicativos refatorados quanto os reformulados podem exigir um banco de dados. Você deve criar, configurar e gerenciar o banco de dados de acordo com os requisitos específicos de cada mecanismo de tempo de execução. AWS O Mainframe Modernization oferece suporte à criptografia em trânsito nesse banco de dados. Se você habilitar o SSL em seu banco de dados, certifique-se de especificar `sslMode` o segredo do banco de dados junto com os detalhes da conexão do banco de dados. Para ter mais informações, consulte Segredo do AWS Secrets Manager.

Se você usa o padrão de refatoração AWS Blu Age e precisa de um BluSam banco de dados, o mecanismo de execução do AWS Blu Age espera um banco de dados Amazon Aurora PostgreSQL, que você deve criar, configurar e gerenciar. O BluSam banco de dados é opcional. Crie esse banco de dados somente se seu aplicativo exigir isso. Para criar o banco de dados, siga as etapas em Criação de um cluster de banco de dados Amazon Aurora no Guia do usuário do Amazon Aurora.

Se você estiver usando o padrão de replataforma da Micro Focus, poderá criar um banco de dados Amazon RDS ou Amazon Aurora PostgreSQL. Para criar o banco de dados, siga as etapas em Criar uma instância de banco de dados Amazon RDS no Guia do usuário do Amazon RDS ou em Criar um cluster de banco de dados Amazon Aurora no Guia do usuário do Amazon Aurora.

Para ambos os mecanismos de tempo de execução, você deve armazenar as credenciais do banco de dados AWS Secrets Manager usando um AWS KMS key para criptografá-las.

## AWS Key Management ServiceChave do

Você deve armazenar as credenciais do banco de dados do aplicativo com segurança em. AWS Secrets Manager Para criar um segredo no Secrets Manager, é necessário criar umAWS KMS key. Para criar uma chave KMS, siga as etapas em Criar chaves no Guia do desenvolvedor do AWS Key Management Service.

Depois de criar a chave, você deve atualizar a política de chaves para conceder permissões de descriptografia ao AWS Mainframe Modernization. Adicione as declarações de política a seguir:

```
{
   "Effect" : "Allow",
   "Principal" : {
   "Service" : "m2.amazonaws.com"
   },
   "Action" : "kms:Decrypt",
   "Resource" : "*"
   }
```

## Segredo do AWS Secrets Manager

Você deve armazenar as credenciais do banco de dados do aplicativo com segurança em. AWS Secrets Manager Para criar um segredo, siga as etapas em Criar um segredo de banco de dados no Guia do Usuário do AWS Secrets Manager.

O AWS Mainframe Modernization oferece suporte à criptografia em trânsito nesse banco de dados. Se você habilitar o SSL em seu banco de dados, certifique-se de especificar `sslMode` o segredo do banco de dados junto com os detalhes da conexão do banco de dados. Você pode especificar um dos seguintes valores para `sslMode`: `verify-full`, `verify-ca`, ou `disable`.

Durante o processo de criação da chave, escolha Permissões de recursos - opcional e, em seguida, escolha Editar permissões. No editor de políticas, adicione uma política baseada em recursos, como a seguinte, para recuperar o conteúdo dos campos criptografados.

```
{
   "Effect" : "Allow",
   "Principal" : {
   "Service" : "m2.amazonaws.com"
   },
   "Action" : "secretsmanager:GetSecretValue",
   "Resource" : "*"
```

```
}
```

# Configurar o aplicativo gerenciado

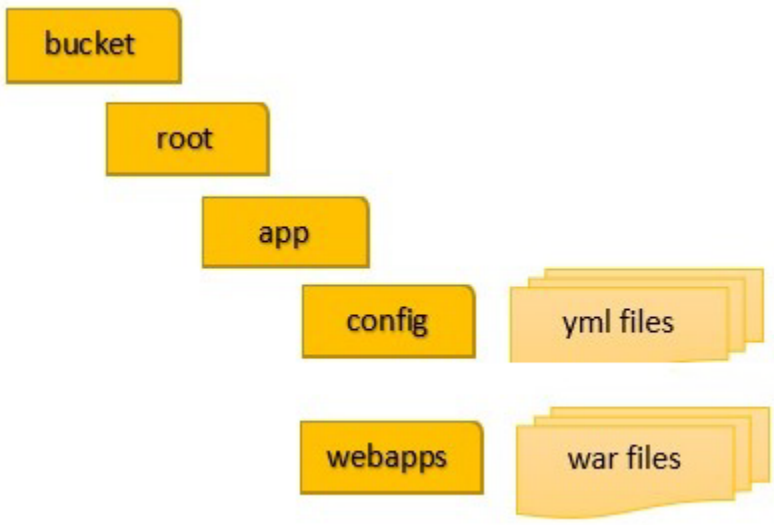
Você pode configurar a aplicação para incluir acesso a utilitários antigos. Você também pode personalizar propriedades adicionais. Para entender o que você pode configurar e onde, é útil entender a estrutura geral de um aplicativo modernizado do AWS Blu Age.

Tópicos



## Estrutura dos aplicativos gerenciados da AWS Blu Age

Se você usa o padrão de refatoração do AWS Blu Age, o mecanismo de tempo de execução do AWS Blu Age espera a seguinte estrutura dentro da `application-name` pasta em seu bucket do S3:




config

Contém os arquivos YAML do seu projeto. Esses são os arquivos YAML específicos do seu aplicativo, normalmente chamados de algo parecido `application-planetsdemo.yaml` e não o `application-main.yaml` arquivo que a Modernização do AWS Mainframe fornece e configura automaticamente para você.

aplicativos da web

Contém os `war` arquivos do seu aplicativo. Esses arquivos são uma saída do processo de modernização.

Um aplicativo também pode ter as seguintes pastas opcionais:

jics/sql

Contém o `initJics.sql` script que inicializa o banco de dados JICS para seu aplicativo.

scripts

Contém scripts de aplicativos, que você também pode fornecer diretamente nos `war` arquivos.

sql

Contém arquivos SQL do aplicativo, que você também pode fornecer diretamente dentro dos `war` arquivos.

.lnk

Contém arquivos LNK do aplicativo, que você também pode fornecer diretamente dentro dos `war` arquivos.

## Gerenciando o consumo de memória Java de um aplicativo

Para gerenciar o consumo de memória Java para o aplicativo, adicione um arquivo de propriedades chamado `tomcat.properties` à `application-name` pasta. Esse arquivo pode ter duas propriedades:`xms`, que especifica o consumo mínimo de memória Java e`xmx`, que especifica o consumo máximo de memória Java. A seguir, um exemplo dos conteúdos em um arquivo `tomcat.properties`:

```
xms=512M
xmx=1G
```

Os valores que você especifica para essas duas propriedades podem estar em qualquer uma das seguintes unidades:

- Bytes: não especifique uma unidade.
- Kilobytes: acrescente um K ao valor.

- Megabytes: acrescente um M ao valor.
- Gigabytes: acrescente um G ao valor.

## Configurando o acesso a utilitários para aplicativos gerenciados

Ao refatorar um aplicativo de mainframe com o AWS Blu Age, talvez seja necessário fornecer suporte para vários programas utilitários de plataforma legados, como IDCAMS, INFUTILB, SORT e assim por diante, se seu aplicativo depender deles. AWS A refatoração do Blu Age fornece esse acesso com um aplicativo web dedicado que é implantado junto com aplicativos modernizados. Esse aplicativo da Web requer um arquivo de configuração `application-utility-pgm.yml`, que você deve fornecer. Se você não fornecer esse arquivo de configuração, o aplicativo Web não poderá ser implantado junto com seu aplicativo e não estará disponível.

Tópicos

- Propriedades de configuração

Este tópico descreve todas as propriedades possíveis que você pode especificar no arquivo `application-utility-pgm.yml` de configuração, junto com seus padrões. O tópico descreve as propriedades obrigatórias e opcionais. O exemplo a seguir é um arquivo de configuração completo. Ele lista as propriedades na ordem que recomendamos. Em seguida, é possível usar esse exemplo como ponto de partida para seu próprio arquivo de configuração.

```
# 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
```

## Propriedades de configuração

Em seu arquivo de configuração, é possível especificar as propriedades a seguir.

spring.jta.enabled

Opcional. Controla se o suporte ao JTA está habilitado. Para utilitários, recomendamos que defina esse valor como `false`.

```
spring.jta.enabled : false
```

logging.config

Obrigatório. Especifica o caminho para o arquivo de configuração do registrador dedicado. Recomendamos que você use o nome `logback-utility.xml` e forneça esse arquivo como parte do aplicativo modernizado. A maneira comum de organizar esses arquivos é colocar todos os arquivos de configuração do registrador no mesmo local, geralmente na subpasta `/config/logback` onde `/config` está a pasta que contém os arquivos de configuração yaml. Para obter mais informações, consulte o Capítulo 3: Configuração do Logback na documentação do Logback.

```
logging.config : classpath:logback-utility.xml
```

encoding

Obrigatório. Especifica o conjunto de caracteres que o programa utilitário usa. Na maioria dos casos, quando você migra das plataformas z/OS, esse conjunto de caracteres é uma variante do EBCDIC e deve corresponder ao conjunto de caracteres configurado para os aplicativos modernizados. Se não estiver definido, o padrão será `ASCII`.

```
encoding : cp1047
```

sysPunchEncoding

Opcional. Especifica o conjunto de caracteres que INFUTILB e DSNUTILB usam para gerar e ler arquivos SYSPUNCH. Se você usar os arquivos SYSPUNCH da plataforma legada como estão, esse valor deve ser uma variante do EBCDIC. Se não estiver definido, o padrão será `ASCII`.

```
sysPunchEncoding : cp1047
```

Configuração da fonte de dados primária

Alguns utilitários relacionados ao banco de dados, como LOAD e UNLOAD, exigem acesso a um banco de dados de destino por meio de uma fonte de dados. Como outras definições de fonte de dados na modernização do AWS mainframe, esse acesso exige que você o use. AWS Secrets Manager As propriedades que apontam para os segredos adequados no Secrets Manager são as seguintes:

spring.aws.client.datasources.primary.secret

Opcional. Especifica o segredo no Secrets Manager que contém as propriedades da fonte de dados.

```
spring.aws.client.datasources.primary.secret: datasource-secret-ARN
```

spring.aws.client.datasources.primary.dbname

Opcional. Especifica o nome do banco de dados de destino se o nome do banco de dados não for fornecido diretamente no segredo do banco de dados, com a `dbname` propriedade.

```
spring.aws.client.datasources.primary.dbname: target-database-name
```

treatLargeNumberAsInteger

Opcional. Relacionado às especificidades do mecanismo de banco de dados Oracle e ao uso dos utilitários DSNTEP2/DSNTEP4. Se você definir esse sinalizador como verdadeiro, números grandes provenientes do banco de dados Oracle (NUMBER (38,0)) serão tratados como números inteiros. Padrão: `false`

```
treatLargeNumberAsInteger : false
```

Modo zoneado

Opcional. Define o modo zoneado para codificar ou decodificar tipos de dados zoneados. Essa configuração influencia a forma como os dígitos do sinal são representados. Os valores a seguir são válidos:

- EBCDIC_STRIC_STRICT: Padrão. Use uma definição estrita para o manuseio de sinais. Dependendo se o conjunto de caracteres é EBCDIC ou ASCII, a representação do dígito de sinal usa os seguintes caracteres:

  - Caracteres EBCDIC que correspondem a bytes (`Cn+Dn`) para representar intervalos de dígitos positivos e negativos (`+0` para `+9` para, `-0` para `-9`). Os caracteres são exibidos como `{`, `A` para `I`, `}`, `J` para `R`
  - Caracteres ASCII que correspondem a bytes (`3n+7n`) para representar intervalos de dígitos positivos e negativos (`+0` para `+9` para, `-0` para `-9`). Os caracteres são exibidos como `0` para `9`, p para y
- EBCDIC_MODIFIED: Use uma definição modificada para o tratamento de sinais. Tanto para EBDIC quanto para ASCII, a mesma lista de caracteres representa os dígitos do sinal, ou seja, mapeados `+0` para `+9` mapeados para `{` + `A` para `I` e `-0` para `-9` mapeados para `}` + `J` para `R`. \
- AS400: Use para ativos legados modernizados provenientes de plataformas iSeries (AS400).

```
zonedMode:EBCDIC_STRICT
```

jcl.type

Opcional. Indica o tipo legado de scripts JCL modernizados. O utilitário IDCAMS usa essa configuração para personalizar o código de retorno se o JCL de invocação for do tipo `vse`. Os valores válidos são os seguintes:

- `mvs` (padrão)
- `vse`

```
jcl.type : mvs
```

Propriedades relacionadas aos utilitários de descarga de banco de dados

Use essas propriedades para configurar utilitários que descarregam tabelas de banco de dados em conjuntos de dados. Todas as propriedades a seguir são opcionais.

Este exemplo mostra todas as propriedades de descarga possíveis.

```
# 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
```

sqlCodePointTurno

Opcional. Especifica um valor inteiro que representa a mudança de ponto do código SQL usada nos dados. O padrão é 0. Isso significa que nenhuma mudança de ponto de código é feita. Alinhe essa configuração com o parâmetro de mudança de ponto do código SQL usado para aplicativos modernizados. Quando a mudança de ponto de código está em uso, o valor mais comum para esse parâmetro é 384.

```
unload.sqlCodePointShift: 0
```

nbi

Opcional. Especifica um byte indicador nulo. Esse é um valor hexadecimal (como uma string) adicionado à direita do valor dos dados. Os dois valores possíveis são os seguintes:

- WhenNull: adicione o valor hexadecimal quando o valor dos dados for nulo. O padrão é6`. Às vezes, o valor alto FF é usado em vez disso.

  ```
  unload.nbi.whenNull: "6F"
  ```

- whenNotNull: adicione o valor hexadecimal quando o valor dos dados não for nulo, mas a coluna for anulável. O padrão é `00` (valor baixo).

  ```
  unload.nbi.whenNotNull: "00"
  ```

useDatabaseConfiguration

Opcional. Especifica as propriedades de formatação de data e hora. Isso é usado para lidar com objetos de data/hora em consultas UNLOAD. O padrão é`false`.

- Se definido como `true`, usa as `pgmDateFormat`, `pgmTimeFormat`, e `pgmTimestampFormat` propriedades do arquivo de configuração principal (`application-main.yml`).
- Se definido como `false`, usa as seguintes propriedades de formatação de data e hora:
  - `unload.format.date`: especifica um padrão de formatação de data. O padrão é`MM/dd/yyyy`.
  - `unload.format.time`: especifica um padrão de formatação de hora. O padrão é`HH.mm.ss`.
  - `unload.format.timestamp`: especifica um padrão de formatação de carimbo de data/hora. O padrão é`yyyy-MM-dd-HH.mm.ss.SSSSSS`.

Tamanho do pedaço

Opcional. Especifica o tamanho dos blocos de dados usados para criar conjuntos de dados SYSREC. Esses conjuntos de dados são o alvo da operação de descarregamento do conjunto de dados, com operações paralelas. O padrão é `0` (sem pedaços).

```
unload.chunkSize:0
```

Tamanho da busca

Opcional. Especifica o tamanho da busca de dados. O valor é o número de registros a serem buscados ao mesmo tempo quando uma estratégia de fragmentos de dados é usada. Padrão: `0`.

```
unload.fetchSize:0
```

varCharIsNulo

Opcional. Especifica como lidar com uma coluna varchar não anulável com conteúdo em branco. O padrão é`false`.

Se você definir esse valor como `true`, o conteúdo da coluna será tratado como uma string vazia para fins de descarga, em vez de uma única string de espaço. Defina esse sinalizador somente `true` para o caso do mecanismo de banco de dados Oracle.

```
unload.varCharIsNull: false
```

Preenchedor de colunas

Opcional. Especifica o valor a ser usado para preencher colunas descarregadas em colunas varchar. Os valores possíveis são espaço ou valores baixos. O padrão é espaço.

```
unload.columnFiller: space
```

Propriedades relacionadas ao carregamento do banco de dados

Use essas propriedades para configurar utilitários que carregam registros do conjunto de dados em um banco de dados de destino, por exemplo, DSNUTILB. Todas as propriedades a seguir são opcionais.

Este exemplo mostra todas as propriedades de carga possíveis.

```
# 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
```

sqlCodePointTurno

Opcional. Especifica um valor inteiro que representa a mudança de ponto do código SQL usada nos dados. O padrão é 0, o que significa que os aplicativos não alteram o ponto de código. Alinhe essa configuração com o parâmetro de mudança de ponto do código SQL usado para aplicativos modernizados. Quando você usa mudanças de ponto de código, o valor mais comum para esse parâmetro é 384.

```
load.sqlCodePointShift : 384
```

batchSize

Opcional. Especifica um valor inteiro que representa o número de registros a serem tratados antes de você enviar uma declaração de lote real para o banco de dados. O padrão é 0.

```
load.batchSize: 500
```

format

Opcional. Especifica os padrões de formatação de data e hora a serem usados para conversões de data/hora durante as operações de carregamento do banco de dados.

- `load.format.localDate`: Padrão de formatação de data local. Isso é padronizado como `dd.MM.yyyy|dd/MM/yyyy|yyyy-MM-dd`.
- `load.format.dbDate`: Padrão de formatação de data do banco de dados. Isso é padronizado como `yyyy-MM-dd`.
- `load.format.localTime`: Padrão de formatação da hora local. Isso é padronizado como `HH:mm:ss|HH.mm.ss`.
- `load.format.dbTime`: Padrão de formatação de hora do banco de dados. Isso é padronizado como `HH:mm:ss`.

mapeamentos de tabela

Opcional. Especifica uma coleção de mapeamentos fornecidos pelo cliente entre nomes de tabelas antigas e modernas. O programa utilitário DSNUTILB consome esses mapeamentos.

Especifique os valores no seguinte formato: MODERN_TABLE_NAME: LEGACY_TABLE_NAME

Exemplo:

```
table-mappings:
 TABLE_1_NAME : LEGACY_TABLE_1_NAME
 TABLE_2_NAME : LEGACY_TABLE_2_NAME
 ...
 TABLE_*N*_NAME : LEGACY_TABLE_*N*_NAME
```

> **Note**
>
> Quando o aplicativo utilitário é iniciado, ele registra explicitamente todos os mapeamentos fornecidos.

# Adicionar propriedades de configuração para o mecanismo AWS Blu Age

Você pode adicionar um arquivo na `config` pasta do seu aplicativo refatorado que lhe dará acesso aos novos recursos no mecanismo de tempo de execução do AWS Blu Age. Você deve nomear esse arquivo `user-properties.yml`. Esse arquivo não substitui a definição do aplicativo, mas a estende. Este tópico descreve as propriedades que você pode incluir no `user-properties.yml` arquivo.

> **Note**
>
> Você não pode alterar alguns parâmetros porque eles são controlados pela modernização do AWS mainframe ou pela definição do aplicativo. Todos os parâmetros definidos na definição do aplicativo têm prioridade sobre os parâmetros especificados em `user-properties.yml`.

Para obter mais informações sobre a estrutura de aplicativos refatorados, consulte. Estrutura dos aplicativos gerenciados da AWS Blu Age

O diagrama a seguir mostra onde localizar o `user-properties.yml` arquivo dentro da estrutura do aplicativo de amostra AWS Blu Age, PlanetsDemo.

```
PlanetsDemo-v1/
   ## config/
   #  ## application-PlanetsDemo.yml
   #  ## user-properties.yml
   ## jics/
   ## webapps/
```

## Referência de propriedades de configuração

Esta é a lista de propriedades disponíveis. Todos os parâmetros são opcionais.

Tópicos

- Propriedades do aplicativo Gapwalk
- Propriedades do batchscript do Gapwalk
- Propriedades Gapwalk Blugen
- Propriedades do comando Gapwalk CL

- Propriedades do corredor Gapwalk CL
- Propriedades Gapwalk JHDB
- Propriedades do Gapwalk JICS
- Propriedades de runtime do Gapwalk
- Propriedades do programa utilitário Gapwalk
- Outras propriedades

Propriedades do aplicativo Gapwalk

bluesam.fileloading.CommitInterval

Opcional. O intervalo de confirmação do bluesam.

Tipo: número

Padrão: 100000

codificação do cartão

Opcional. Codificação do cartão: para ser usado com `useControlMVariable`.

Tipo: string

Padrão: CP1145

verifique o tamanho do arquivo de entrada

Opcional. Especifica se um cheque deve ser liberado se o tamanho do arquivo for múltiplo do tamanho do registro.

Tipo: booliano

Padrão: False

database.cursor.overflow.allowed

Opcional. Especifica se é permitido que o cursor transborde. Defina como `true` para realizar uma próxima chamada no cursor, seja qual for sua posição. Defina como `false` para verificar se o cursor está na última posição antes de realizar uma próxima chamada no cursor. Ative somente se o cursor for ROLÁVEL (SENSÍVEL ou INSENSÍVEL)

Tipo: booliano

Padrão: True

Simplificador de dados. onInvalidNumericDados

Opcional. Como reagir ao decodificar dados numéricos inválidos. Os valores permitidos são: `reject`, `toleratespaces`, `toleratespaceslowvalues` e `toleratemost`.

Tipo: string

Padrão: rejeitar

defaultKeepExistingArquivos

Opcional. Especifica se o valor anterior padrão do conjunto de dados deve ser definido.

Tipo: booliano

Padrão: False

disposição.verifique a existência

Opcional. Especifica se deve liberar uma verificação da existência do arquivo para o conjunto de dados com DISP SHR ou OLD.

Tipo: booliano

Padrão: False

ExternalSort.Threshold

Opcional. O limite de classificação: quando alternar para a classificação externa (mesclagem).

Tipo: string

Padrão: nulo

```
externalSort.threshold: 12MB
```

Forçar o RH

Opcional. Especifica se o SYSPRINT legível por humanos deve ser usado no console ou na saída do arquivo.

Tipo: booliano

Padrão: False

Data forçada

Opcional. Força uma data e hora específicas no banco de dados. Use somente durante o desenvolvimento e o teste.

Padrão: nulo

```
forcedDate: 2022-08-26T12:59:58.123456+01:57
```

Data congelada

Opcional. Congela a data e a hora no banco de dados. Use somente durante o desenvolvimento e o teste.

Padrão: False

```
frozenDate: false
```

IMS.Messages.ExtendedSize

Opcional. Especifica se o extendedSize deve ser definido nas mensagens ims.

Tipo: booliano

Padrão: False

Tempo limite de bloqueio

Opcional. O tempo limite em milissegundos de uma transação quando não é possível adquirir um bloqueio dentro de um período de tempo especificado.

Tipo: número

Padrão: 500

MapTransfo.prefixos

Opcional. Lista de prefixos a serem usados ao transformar variáveis ControlM. Cada um separado por vírgula.

Tipo: string

Padrão: &,@,%%

consulta. useConcatCondition

Opcional. Especifica se a condição da chave é criada por concatenação de chaves ou não.

Tipo: booliano

Padrão: False

Reverter o RTE

Opcional. Especifica se a transação de unidade de execução implícita deve ser revertida em exceções de tempo de execução.

Tipo: booliano

Padrão: False

sctThreadLimit

Opcional. O limite de threads para acionar scripts.

Tipo: número

Padrão: 5

sqlCodePointTurno

Opcional. A mudança de ponto do código sql. Muda o ponto de código dos caracteres de controle que podemos encontrar ao migrar dados rdbms legados para um rdbms moderno. Por exemplo, você pode especificar `384` para corresponder ao `\u0180` caractere Unicode.

Tipo: número

Padrão: 0

sqlIntegerOverflowPermitido

Opcional. Especifica se é permitido o estouro de números inteiros SQL, ou seja, se é permitido colocar valores maiores na variável do host.

Tipo: booliano

Padrão: False

stepFailWhenAbend

Opcional. Especifica se a suspensão deve ser levantada se uma etapa falhar ou concluir a execução.

Tipo: booliano

Padrão: True

stopExecutionWhenProgNotFound

Opcional. Especifica se a execução deve ser interrompida se um programa não for encontrado. Se definido como `true`, interrompe a execução se um programa não for encontrado.

Tipo: booliano

Padrão: True

uppercaseUserInput

Opcional. Especifica se a entrada do usuário deve estar em maiúsculas.

Tipo: booliano

Padrão: True

Use a variável Control M

Opcional. Especifica se a especificação Control-m deve ser usada para substituição de variáveis.

Tipo: booliano

Padrão: False

Propriedades do batchscript do Gapwalk

encoding

Opcional. A codificação usada em projetos de batchscript (não com groovy). Espera uma codificação válida `CP1047`, `IBM930`, `ASCII`, `UTF-8`...

Tipo: string

Padrão: ASCII

Propriedades Gapwalk Blugen

managers.trancode

Opcional. O gerenciador de diálogos trancode mapeamento. Permite mapear um código de transação JICS para um gerenciador de diálogos. O formato esperado é `trancode1:dialogManager1;trancode2:dialogManager2;`.

Tipo: string

Padrão: nulo

```
managers.trancode: OR12:MYDIALOG1
```

Propriedades do comando Gapwalk CL

comandos desativar

Opcional. Lista de comandos a serem desativados, separados por vírgula. Os valores permitidos são: `PGM_BASIC`, `RCVMSG`, `SNDRCVF`, `CHGVAR`, `QCLRDTAQ`, `RTVJOBA`, `ADDLFM`, `ADDPFM`, `RCVF`, `OVRDBF`, `DLTOVR`, `CPYF` e `SNDDTAQ`. Útil quando você deseja desativar ou substituir um programa existente. `PGM_BASIC` é um programa de velocidade específico projetado para fins de depuração.

Tipo: string

Padrão: nulo

spring.datasource.primary.jndi-name

Opcional. A principal fonte de dados Java Naming And Directory Interface (jndi).

Tipo: string

Padrão: jdbc/primary

Modo zoneado

Opcional. O modo para codificar ou decodificar tipos de dados zoneados. Os valores permitidos são `EBCDIC_STRICT` / `EBCDIC_MODIFIED` / `AS400`.

Tipo: string

Padrão: EBCDIC_STRICT

Propriedades do corredor Gapwalk CL

cl.configuration.context.encoding

Opcional. A codificação dos arquivos CL. Espera uma codificação válida `CP1047`, `IBM930`, `ASCII`, `UTF-8`...

Tipo: string

Padrão: CP297

CL. Modo zoneado

Opcional. O modo para codificar ou decodificar comandos da linguagem de controle (CL). Os valores permitidos são `EBCDIC_STRICT` / `EBCDIC_MODIFIED` / `AS400`.

Tipo: string

Padrão: EBCDIC_STRICT

Propriedades Gapwalk JHDB

programas ims

Opcional. Lista de programas IMS a serem usados. Separe cada parâmetro com um ponto e vírgula (`;`) e cada transação com uma vírgula (`,`). Por exemplo: `ims.programs: PCP008,PCT008;PCP054,PCT054;PCP066,PCT066;PCP068,PCT068;`

Tipo: string

Padrão: nulo

jhdb.Caminho do ponto de verificação

Opcional. Caso `jhdb.checkpointPersistence` contrário `none`, esse parâmetro permite que você configure o caminho de persistência do ponto de verificação (local de armazenamento do arquivo checkpoint.dat). Todos os dados dos pontos de verificação contidos no registro são serializados e copiados em um arquivo (checkpoint.dat) localizado na pasta fornecida. Observe que somente os dados do ponto de verificação (ScriptID, StepID, posição do banco de dados e área do ponto de verificação) são afetados por esse backup.

Tipo: string

Padrão: arquivo:. /configurar/

jhdb.Checkpoint Persistence

Opcional. O modo de persistência do ponto de verificação. Os valores permitidos são `none` / `add` / `end`. Use `add` para manter os pontos de verificação quando um novo for criado e adicionado ao registro. Use `end` para manter o ponto de verificação no desligamento do servidor. Quaisquer outros valores desativam a persistência. Observe que sempre que um novo ponto de verificação for adicionado ao registro, todos os pontos de verificação existentes serão serializados e o arquivo será apagado. Não é um acréscimo aos dados existentes no arquivo. Portanto, dependendo do número de pontos de verificação, isso pode ter algum efeito no desempenho.

Tipo: string

Padrão: nenhum

jhdb.configuration.context.encoding

Opcional. A codificação JHDB (Java Hierarchical Database). Espera uma string de codificação válida `CP1047`, `IBM930`, `ASCII`, `UTF-8`...

Tipo: string

Padrão: CP297

jhdb. identificationCardData

Opcional. Usado para codificar alguns “dados do cartão de identificação do operador” no campo MID designado pelo parâmetro CARD.

Tipo: string

Padrão: ""

jhdb.lterm

Opcional. Permite que você force um ID de terminal lógico comum no caso de uma emulação de IMS. Se não for definido, o sessionID será usado.

Tipo: string

Padrão: nulo

jhdb.metadata.extrapath

Um parâmetro de configuração que especifica uma pasta raiz extra específica do tempo de execução para as pastas psbs e dbds.

Tipo: string

Padrão: arquivo:. /configurar/

> **Note**
>
> No momento, devido a restrições de implantação, você deve copiar seus diretórios dbds e psbs no diretório config da sua aplicação ou em um subdiretório do diretório config: por exemplo, config/setup
>
> ```
> config
> |- setup
>    |- dbds
>    |- psbs
> ```
>
> e definir em application-jhdb.yml
> `jhdb.metadata.extrapath: file: ./config/setup/`

jhdb.navigation.cachenexts

Opcional. A duração do cache (em milissegundos) usada na navegação hierárquica para um RDBMS.

Tipo: número

Padrão: 5000

jhdb.query. limitJoinUsage

Opcional. Especifica se o parâmetro limite de uso de junção deve ser usado em gráficos RDBMS.

Tipo: booliano

Padrão: True

jhdb. use-db-prefix

Opcional. Especifica se um prefixo de banco de dados deve ser ativado na navegação hierárquica para um RDBMS.

Tipo: booliano

Padrão: True

Propriedades do Gapwalk JICS

jics.data. dataJsonInitLocalização

Opcional. Localização do arquivo json preparado pelo Analyzer a partir da análise do CSD e usado para inicializar o banco de dados jics,

Tipo: string

Padrão: ""

jics.db. dataScriptLocation

Opcional. Localização do script initJics.sql, preparado pelo Analyzer a partir da análise das exportações de CSD do mainframe.

Tipo: string

Padrão: ""

jics.db. dataTestQueryLocalização

Opcional. Localização de um script sql contendo uma única consulta sql que deve retornar uma contagem de objetos (por exemplo: contagem do número de registros na tabela do programa jics). Se a contagem for igual a 0, o banco de dados será carregado usando o script `jics.db.dataScriptLocation`, caso contrário, o carregamento do banco de dados será ignorado.

Tipo: string

Padrão: ""

jics.db. ddlScriptLocation

Opcional. A localização do script Jics ddl. Permite que você inicie o esquema do banco de dados jics usando um script.sql.

Tipo: string

Padrão: ""

```
jics.db.ddlScriptLocation: ./jics/sql/jics.sql
```

jics.db. schemaTestQueryLocalização

Opcional. Localização do arquivo sql que deve conter uma consulta exclusiva que retorna o número de objetos no esquema jics (se houver).

Tipo: string

Padrão: ""

sucos. runUnitLauncherPool. Habilitar

Opcional. Especifica se o pool do lançador de unidades de execução deve ser ativado no JICS.

Tipo: booliano

Padrão: False

sucos. runUnitLauncherTamanho da piscina

Opcional. O tamanho do pool do lançador da unidade de execução no JICS.

Tipo: número

Padrão: 20

sucos. runUnitLauncherPool. Intervalo de validação

Opcional: O intervalo de validação do pool do lançador de unidades de execução no JICS, expresso em milissegundos.

Tipo: número

Padrão: 1000

jics.queues.sqs.region

Opcional. O Região da AWS para Amazon SQS, usado no JICS. É recomendável definir a mesma região do aplicativo implantado para fins de desempenho, mas isso não é obrigatório.

Tipo: string

Padrão: eu-west-1

jics.xa.agent.timeout

Opcional. Define a duração máxima para que o agente xa responsável pelo gerenciamento de transações distribuídas conclua suas operações.

Tipo: número

Padrão: nulo

mq.queues.sqs.region

Opcional. O Região da AWS para o serviço Amazon SQS MQ.

Tipo: string

Padrão: eu-west-3

Executor de tarefas. allowCoreThreadTimeOut

Opcional. Especifica se os encadeamentos principais devem atingir o tempo limite no JCIS. Isso permite o crescimento e a redução dinâmicos, mesmo em combinação com uma fila diferente de zero (já que o tamanho máximo do pool só aumentará quando a fila estiver cheia).

Tipo: booliano

Padrão: False

Executor de tarefas. corePoolSize

Opcional. Quando uma transação em um terminal é iniciada por meio de um script groovy, um novo thread é criado. Use esse parâmetro para configurar o tamanho do pool principal.

Tipo: número

Padrão: 5

Executor de tarefas. maxPoolSize

Opcional. Quando uma transação em um terminal é iniciada por meio de um script groovy, um novo tópico é criado. Use esse parâmetro para configurar o tamanho máximo do pool (número máximo de threads paralelos).

Tipo: número

Padrão: 10

TaskExecutor.queueCapacity

Opcional. Quando uma transação em um terminal é iniciada por meio de um script groovy, um novo tópico é criado. Use esse parâmetro para configurar o tamanho da fila. (= número máximo de transações pendentes quando `taskExecutor.maxPoolSize` atingido)

Tipo: número

Padrão: 50

Propriedades de runtime do Gapwalk

Metadados em cache

Opcional. Especifica se os metadados do banco de dados devem ser armazenados em cache.

Tipo: booliano

Padrão: True

check-groovy-file

Opcional. Especifica se o conteúdo dos arquivos groovy deve ser verificado antes do registro.

Tipo: booliano

Padrão: True

Estatísticas do banco de dados

Opcional. Especifica se devem permitir que os construtores de SQL coletem e exibam informações estatísticas.

Tipo: booliano

Padrão: False

dateTimeFormat

Opcional. dateTimeFormat Descreve como inserir o tipo de data e hora do banco de dados em entidades simplificadoras de dados. Os valores permitidos são `ISO` / `EUR` / `USA` / `LOCAL`

Tipo: string

Padrão: ISO

dbDateFormat

Opcional. O formato da data alvo do banco de dados.

Tipo: string

Padrão: YYYY-MM-dd

dbTimeFormat

Opcional. O formato da hora alvo do banco de dados.

Tipo: string

Padrão: HH:mm:ss

dbTimestampFormat

Opcional. O formato do timestamp de destino do banco de dados.

Tipo: string

Padrão: aaaa-mm-dd hh:mm:ss.ssssss

Tamanho da busca

Opcional. O valor fetchSize para cursores. Use ao buscar dados usando fragmentos por meio de utilitários de carregamento/descarregamento.

Tipo: número

Padrão: 10

Forçar a desativação do SQL TrimStringType

Opcional. Especifica se o corte de todos os parâmetros da string sql deve ser desativado.

Tipo: booliano

Padrão: False

localDateFormat

Opcional. Lista de formatos de data locais. Separe cada formato com |.

Tipo: string

localTimeFormat

Opcional. Lista de formatos de horário local. Separe cada formato com |.

Tipo: string

localTimestampFormat

Opcional. Lista de formatos de carimbo de data/hora locais. Separe cada formato com |.

Tipo: string

Padrão:

pgmDateFormat

Opcional. O formato de data e hora usado nos programas.

Tipo: string

Padrão: YYYY-MM-dd

pgmTimeFormat

Opcional. O formato de hora usado para execução de pgm (programas).

Tipo: string

Padrão: HH.MM.ss

pgmTimestampFormat

Opcional. O formato do registro de data e hora.

Tipo: string

Padrão: aaaa-mm-dd-hh.mm.ss.ssssss

Propriedades do programa utilitário Gapwalk

jcl.type

Opcional. `.jcl` tipo de arquivo. Os valores permitidos são `jcl` / `vse`. Os comandos PRINT/REPRO do utilitário IDCAMS retornam 4 se o arquivo estiver vazio para um jcl que não seja vse.

Tipo: string

Padrão: mvs

listcat.variablelengthpreprocessor.enabled

Opcional. Especifica se deseja ativar o pré-processador de comprimento variável para o comando LISTCAT.

Tipo: booliano

Padrão: False

listcat.variablelengthpreprocessor.type

Opcional. O tipo de objetos contidos no arquivo listcat, se você habilitar `listcat.variablelengthpreprocessor.enabled`. Os valores permitidos são `rdw` / `bdw`.

Tipo: string

Padrão: rdw

Carregar. Tamanho do lote

Opcional. O tamanho do lote do utilitário de carga.

Tipo: número

Padrão: 0

Load.format.dbDdate

Opcional. O formato do banco de dados do utilitário de carga a ser usado.

Tipo: string

Padrão: YYYY-MM-dd

Load.format.dbTime

Opcional. O tempo de uso do banco de dados do utilitário de carregamento.

Tipo: string

Padrão: HH:mm:ss

Load.format.localDate

Opcional. O formato de data local do utilitário de carregamento a ser usado.

Tipo: string

Padrão: dd.mm.YYYY|dd/mm/aaa|AAAA-MM-dd

load.format.localTime

Opcional. O formato de hora local do utilitário de carregamento a ser usado.

Tipo: string

Padrão: HH:MM:SS|HH.MM.SS

carregar. sqlCodePointTurno

Opcional. A mudança de pontos do código SQL para o utilitário de carregamento. Executa o processo de mudança de caracteres. Obrigatório quando seu banco de dados de destino do DB2 é o Postgresql.

Tipo: número

Padrão: 0

sysPunchEncoding

Opcional. O conjunto de caracteres de codificação syspunch. Os valores suportados são `Cp1047` / `ASCII`.

Tipo: string

Padrão: ASCII

treatLargeNumberAsInteger

Opcional. Especifica se os números grandes devem ser tratados como `Integer`. Eles são tratados como `BigDecimal` padrão.

Tipo: booliano

Padrão: False

Unload.chunkSize

Opcional. Tamanho do pedaço usado para o utilitário de descarga.

Tipo: número

Padrão: 0

Descarregue.columnFiller

Opcional. O preenchedor de colunas do utilitário de descarga.

Tipo: string

Padrão: espaço

Unload.fetchSize

Opcional. Permite ajustar o tamanho da busca ao manipular cursores no utilitário de descarga.

Tipo: número

Padrão: 0

descarregar.format.date

Opcional. Se `unload.useDatabaseConfiguration` estiver ativado, o formato de data a ser usado no utilitário de descarga. Para obter mais informações, consulte unload.format.date.

Tipo: string

Padrão: MM/dd/aaaa

descarregar.format.time

Opcional. Se `unload.useDatabaseConfiguration` estiver ativado, o formato de hora a ser usado no utilitário de descarga.

Tipo: string

Padrão: HH.MM.ss

unload.format.timestamp

Opcional. Se `unload.useDatabaseConfiguration` estiver ativado, o formato de carimbo de data/hora a ser usado no utilitário de descarga.

Tipo: string

Padrão: aaaa-mm-dd-hh.mm.ss.ssssss

descarregue.nbi. whenNotNull

Opcional. O valor do Indicador de Byte Nulo (nbi) a ser adicionado quando o valor do banco de dados não for nulo.

Tipo: hexadecimal

Padrão: 00

Descarregue.nbi.whenNull

Opcional. O valor do Indicador de Byte Nulo (nbi) a ser adicionado quando o valor do banco de dados for nulo.

Tipo: hexadecimal

Padrão: 6F

descarregue.nbi. writeNullIndicator

Opcional. Especifica se o indicador nulo deve ser gravado no arquivo de saída de descarga.

Tipo: booliano

Padrão: False

descarregar. sqlCodePointTurno

Opcional. O utilitário de mudança de pontos do código SQL para descarga. Executa o processo de mudança de caracteres. Obrigatório quando seu banco de dados de destino do DB2 é o Postgresql.

Tipo: número

Padrão: 0

descarregar. useDatabaseConfiguration

Opcional. Especifica se a configuração de data ou hora do application-main.yml deve ser usada no utilitário de descarregamento.

Tipo: booliano

Padrão: False

descarregar. varCharIsNulo

Opcional. Use esse parâmetro no programa INFTILB, se definido como `true`, todos os campos não anuláveis com valores em branco (espaço) retornarão uma string vazia.

Tipo: booliano

Padrão: False

Outras propriedades

qtemp.cleanup.threshold.hours

Opcional. Para especificar quando `qtemp.dblog` está ativado. A vida útil da partição db (em horas).

Tipo: número

Padrão: 0

qtemp.dblog

Opcional. Se deve habilitar o registro do banco de dados QTEMP.

Tipo: booliano

Padrão: False

qtemp.uuid.length

Opcional. O comprimento de identificação exclusivo do QTEMP.

Tipo: número

Padrão: 9

quartz.scheduler. stand-by-if-error

Opcional. Especifica se a execução do trabalho deve ser acionado se o agendador de trabalhos estiver no modo de espera. Se verdadeiro, quando ativada, a execução do trabalho não é acionada.

Tipo: booliano

Padrão: False

warmUpCache

Opcional. Especifica se todos os dados da tabela de comunicação de dados devem ser carregados em um cache de aquecimento na inicialização do servidor.

Tipo: booliano

Padrão: False

# AWS Referência de definição de aplicativo de modernização de mainframe

Na Modernização do AWS Mainframe, você configura os aplicativos de mainframe migrados em um arquivo JSON de definição de aplicativo, que é específico para o mecanismo de tempo de execução escolhido. Uma definição de aplicativo contém informações gerais e informações específicas do

mecanismo. Este tópico descreve as definições dos aplicativos AWS Blu Age e Micro Focus e identifica todos os elementos obrigatórios e opcionais.

Tópicos



## Seção de cabeçalho geral

Cada definição de aplicativo começa com informações gerais sobre a versão do modelo e os locais de origem. A versão atual da definição do aplicativo é 2.0. Embora a versão 1 ainda funcione, ela está prestes a ser desativada. É recomendável usar a versão 2 ao criar ou atualizar aplicativos.

Use a estrutura a seguir para especificar a versão do modelo e os locais de origem.

```
"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**
>
> É possível usar a seguinte sintaxe se você quiser inserir o ARN do S3 como 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"
            }
        }
    ]
```

Versão do modelo

Obrigatório. Especifica a versão do arquivo de definição da aplicação. Não mude esse valor. Atualmente, o único valor permitido é 2,0. Especifique `template-version` com uma string.

locais de origem

Especifica os locais dos arquivos e outros recursos que o aplicativo exige durante o tempo de execução.

propriedades

Fornece os detalhes do local de origem. Cada propriedade é especificada com uma string.

- `s3-bucket` - Obrigatório. Especifica o nome do bucket do Amazon S3 onde os arquivos estão armazenados.
- `s3-key-prefix` - Obrigatório. Especifica o nome da pasta no bucket do Amazon S3 em que os arquivos são armazenados.

> **Note**
>
> Certifique-se de especificar o nome do bucket do Amazon S3, não o ARN do bucket. Não especifique um caminho absoluto para os recursos no bucket.

## Visão geral da seção de definição

Especifica as definições de recursos dos serviços, configurações, dados e outros recursos típicos que o aplicativo precisa para ser executado. Quando você atualiza uma definição de aplicativo, a modernização do AWS mainframe detecta alterações comparando `definition` as listas `source-locations` e das versões anteriores e atuais do arquivo JSON de definição do aplicativo.

A seção de definição é específica do motor e está sujeita a alterações. As seções a seguir mostram exemplos de definições de aplicações específicas do motor para ambos os motores.

## AWS Exemplo de definição do aplicativo 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 Detalhes da definição de Blu Age

## Receptores: obrigatório

Especifique a porta que você usará para acessar o aplicativo por meio do Elastic Load Balancing criado pela AWS Mainframe Modernization. Use a seguinte estrutura:

```
"listeners": [{
            "port": 8194,
            "type": "http"
        }],
```

porta

Obrigatório. Você pode usar qualquer porta disponível, exceto as conhecidas portas de 0 a 1023. Recomendamos usar o intervalo de 8192 a 8199. Verifique se não há outros ouvintes ou aplicativos operando nessa porta.

type

Obrigatório. No momento, só há compatibilidade com `http`.

## AWS Aplicativo Blu Age - obrigatório

Especifique o local em que o mecanismo coleta o arquivo de imagem do aplicativo usando a estrutura a seguir.

```
"ba-application": {
            "app-location": "${s3-source}/murachs-v6/",
            "files-directory": "/m2/mount/myfolder",
            "enable-jics": <true|false>,
            "shared-app-location": "${s3-source}/shared/"
        },
```

localização da aplicação

O local específico no Amazon S3 em que o arquivo de imagem da aplicação é armazenado.

diretório de arquivos

Opcional. A localização dos arquivos de entrada/saída para lotes. Deve ser uma subpasta da configuração do ponto de montagem do Amazon EFS ou do Amazon FSx no nível do ambiente.

habilitar jics

Opcional. Especifica se o JICS deve ser ativado. O valor padrão é verdadeiro. Definir isso como false impede que o banco de dados JICS seja gerado.

shared-app-location

Opcional. Localização adicional no Amazon S3 em que os elementos de aplicativo compartilhados são armazenados. Ele pode conter o mesmo tipo de estrutura de aplicativo que a localização do aplicativo.

## BlusAM - opcional

Especifique o banco de dados BlusAM e o cache Redis usando a estrutura a seguir.

```
"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

Especifica as propriedades do banco de dados usado com o aplicativo. O banco de dados deve ser um banco de dados do Aurora PostgreSQL. Você pode especificar as seguintes propriedades:

- `nb-threads`: opcional. Especifica quantos encadeamentos dedicados são usados para o mecanismo de gravação no qual o mecanismo blusam depende. O padrão é 8.
- `batch-size`: opcional. Especifica o limite que o mecanismo de gravação posterior usa para iniciar as operações de armazenamento em lote. O limite representa o número de registros modificados que iniciarão uma operação de armazenamento em lote para garantir que os

registros modificados persistam. O gatilho em si é baseado em uma combinação do tamanho do lote e do tempo decorrido de um segundo, o que for atingido primeiro. O padrão é 10000.

- `name`: opcional. Especifica o nome do banco de dados.
- `secret-manager-arn` - Especifica o nome de recurso da Amazon (ARN) do segredo do que contém as credenciais do banco de dados. Para ter mais informações, consulte Etapa 4: criar e configurar um segredo de AWS Secrets Manager banco de dados.

redis

Especifica as propriedades do cache Redis que o aplicativo usa para armazenar os dados temporários necessários em um local central para melhorar o desempenho. Recomendamos que você criptografe e proteja com senha o cache do Redis.

- `hostname` - Especifica a localização do cache do Redis.
- `port` - Especifica a porta, normalmente 6379, pela qual o cache do Redis envia e recebe comunicação.
- `useSsl` - Especifica se o cache do Redis está criptografado. Se o cache não estiver criptografado, defina `useSsl` como false.
- `secret-manager-arn` - Especifica o nome de recurso da Amazon (ARN) do segredo do que contém as credenciais do banco de dados. Se o cache do Redis não estiver protegido por senha, não especifique. `secret-manager-arn` Para ter mais informações, consulte Etapa 4: criar e configurar um segredo de AWS Secrets Manager banco de dados.

## AWS Filas de mensagens do Blu Age - opcionais

Especifique os detalhes da conexão JMS-MQ para AWS o aplicativo 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"
  }
]
```

tipo de produto

Obrigatório. Especifica o tipo de produto. Atualmente, isso só pode ser “JMS-MQ” para AWS aplicativos Blu Age.

gerenciador de filas

Obrigatório. Especifica o nome do gerenciador de filas.

channel

Obrigatório. Especifica o nome do canal de conexão do servidor.

hostname

Obrigatório. Especifica o nome do host do servidor da fila de mensagens.

porta

Obrigatório. Especifica o número da porta do ouvinte em que o servidor está escutando.

ID do usuário

Opcional. Especifica a ID da conta de usuário permitida para realizar operações de fila de mensagens no canal especificado.

secret-manager-arn

Opcional. Especifica o nome de recurso da Amazon (ARN) do Secrets Manager que fornece a senha do usuário especificado.

## Definição de aplicativo do Micro Focus

A seção de definição de exemplo a seguir é para o mecanismo de tempo de execução da Micro Focus e contém elementos obrigatórios e opcionais.

```
{
 "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"
        }]
    }
}
```

## Detalhes da definição do Micro Focus

O conteúdo na seção de definição do arquivo de definição do aplicativo Micro Focus varia, dependendo dos recursos que seu aplicativo de mainframe migrado exige em tempo de execução.

### Ouvinte (s) - obrigatório

Especifique um ouvinte usando a seguinte estrutura:

```
"listeners": [{
    "port": 5101,
    "type": "tn3270"
}],
```

porta

Para tn3270, o padrão é 5101. Para outros tipos de ouvintes de serviço, a porta varia. Você pode usar qualquer porta disponível, exceto as conhecidas portas de 0 a 1023. Cada ouvinte deve ter uma porta distinta. Os ouvintes não devem compartilhar portas. Para obter mais informações, consulte Listener Control na documentação do Micro Focus Enterprise Server.

type

Especifica o tipo de ouvinte de serviço. Para obter mais informações, consulte Listeners na documentação do Micro Focus Enterprise Server.

### Localizações do conjunto de dados - obrigatório

Especifique a localização do conjunto de dados usando a estrutura a seguir.

```
"dataset-location": {
            "db-locations": [{
                "name": "Database1",
```

```
                "secret-manager-arn": "arn:aws:secrets:1234:us-east-1:secret:123456"
            }],
 }
```

localizações de banco de dados

Especifica a localização dos conjuntos de dados que o aplicativo migrado cria. Atualmente, a modernização do AWS mainframe suporta somente conjuntos de dados de um único banco de dados VSAM.

- `name` - Especifica o nome da instância do banco de dados que contém os conjuntos de dados criados pelo aplicativo migrado.
- `secret-manager-arn` - Especifica o nome de recurso da Amazon (ARN) do segredo do que contém as credenciais do banco de dados.

## Manipulador de autenticação e autorização do Amazon Cognito — opcional

AWS A modernização do mainframe usa o Amazon Cognito para autenticação e autorização de aplicativos migrados. Especifique o manipulador de autenticação do Amazon Cognito usando a estrutura a seguir.

```
"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

Especifica o grupo de usuários do Amazon Cognito AWS que o Mainframe Modernization usa para autenticar os usuários do aplicativo migrado. O Região da AWS para o grupo de usuários deve corresponder ao do Região da AWS aplicativo de modernização do AWS mainframe.

ID do cliente

Especifica o aplicativo migrado que o usuário autenticado pode acessar.

identity-pool-id

Especifica o pool de identidade do Amazon Cognito em que o usuário autenticado troca um token do grupo de usuários por credenciais que permitem que o usuário acesse a modernização do

mainframe. AWS O Região da AWS for do pool de identidades deve corresponder ao do Região da AWS aplicativo de modernização do AWS mainframe.

## Manipulador do LDAP e do Active Directory: opcional

É possível integrar sua aplicação ao Active Directory (AD) ou a qualquer tipo de servidor LDAP para possibilitar que os usuários da aplicação usem suas credenciais do LDAP/AD para autorização e autenticação.

Como integrar a aplicação ao AD

1. Siga as etapas descritas em Configurar o Active Directory para a segurança do Enterprise Server na documentação do Micro Focus Enterprise Server.
2. Crie um AWS Secrets Manager segredo com os detalhes do AD/LDAP para cada servidor AD/LDAP que você deseja usar com seu aplicativo. Para obter informações sobre como criar um segredo, consulte Criar um segredo do AWS Secrets Manager no Guia AWS Secrets Manager do usuário. Para o tipo de segredo, escolha Outro tipo de segredo e inclua os seguintes pares de chave/valor.

```
{
    "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>"
}
```

⚠ Recomendações de segurança

- Pois`connectionPath`, a modernização do AWS mainframe oferece suporte aos protocolos LDAP e LDAP sobre SSL (LDAPS). Recomendamos usar o LDAPS porque é mais seguro e evita que as credenciais apareçam nas transmissões da rede.
- Para `authorizedId` e `password`, recomendamos que você especifique as credenciais de um usuário sem mais permissões do que as permissões mais

restritivas de somente leitura e verificação necessárias para que a aplicação seja executada.

- Recomendamos fazer a alternância das credenciais do AD/LDAP regularmente.
- Não crie usuários do AD com o nome de usuário `awsuser` ou `mfuser`. Esses dois nomes de usuário são reservados para uso da AWS .

Veja um exemplo a seguir.

```
{
    "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"
}
```

Crie o segredo com uma chave do KMS gerenciada pelo cliente. Você deve conceder à Modernização do AWS Mainframe `DescribeSecret` as permissões `GetSecretValue` e sobre o segredo `Decrypt` e `DescribeKey` as permissões sobre a chave KMS. Para obter mais informações, consulte Permissões para a chave KMS no Guia do AWS Secrets Manager usuário.

3. Adicione o seguinte à definição da aplicação:

```
"ldap-ad-auth-handler": {
    "ldap-ad-connection-secrets": [LIST OF AD/LDAP SECRETS]
}
```

Veja um exemplo a seguir.

```
"ldap-ad-auth-handler": {
    "ldap-ad-connection-secrets": ["arn:aws:secrets:1234:us-east-1:secret:123456"]
}
```

O manipulador de autenticação do LDAP/AD está disponível para o Micro Focus 8.0.11 e versões posteriores.

## Configurações de lote - obrigatórias

Especifique os detalhes exigidos pelos trabalhos em lotes que são executados como parte do aplicativo usando a estrutura a seguir.

```
"batch-settings": {
            "initiators": [{
                "classes": ["A","B"],
                "description": "initiator...."
            }],
            "jcl-file-location": "${s3-source}/batch/jcls"
 }
```

iniciadores

Especifica um iniciador de lote que é iniciado quando o aplicativo migrado é iniciado com êxito e continua em execução até que o aplicativo seja interrompido. Você pode definir uma ou várias classes por iniciador. Você também pode definir vários iniciadores. Por exemplo: .

```
"batch-settings": {
            "initiators": [
                {
                  "classes": ["A", "B"],
                  "description": "initiator...."
                },
                {
                  "classes": ["C", "D"],
                  "description": "initiator...."
                }
            ],
            "jcl-file-location": "${s3-source}/batch/jcls"
 }
```

Para obter mais informações, consulte Para definir um iniciador de lote ou uma impressora SEP na documentação do Micro Focus Enterprise Server.

- `classes` - Especifica as classes de trabalho que o iniciador pode executar. Você pode usar até 36 caracteres. Você pode usar os seguintes caracteres: A-Z ou 0-9.

- `description` - Descreve para que serve o iniciador.
- `jcl-file-location` - Especifica a localização dos arquivos JCL exigidos pelos trabalhos em lotes que o aplicativo migrado executa.

## Configurações do CICS - obrigatórias

Especifique os detalhes necessários para as transações do CICS que são executadas como parte do aplicativo usando a estrutura a seguir.

```
"cics-settings": {
            "binary-file-location": "${s3-source}/cics/binaries",
            "csd-file-location": "${s3-source}/cics/def",
            "system-initialization-table": "BNKCICV"
        }
```

binary-file-location

Especifica o local dos arquivos do programa de transação do CICS.

csd-file-location

Especifica a localização do arquivo de definição de recursos (CSD) do CICS para esse aplicativo. Para obter mais informações, consulte Definições de recursos do CICS na documentação do Micro Focus Enterprise Server.

system-initialization-table

Especifica a tabela de inicialização do sistema (SIT) que o aplicativo migrado usa. O nome da tabela SIT pode ter até 8 caracteres. Você pode usar A-Z, 0-9, $, @ e #. Para obter mais informações, consulte Definições de recursos do CICS na documentação do Micro Focus Enterprise Server.

## Recursos XA - necessários

Especifique os detalhes necessários para os recursos XA que o aplicativo exige usando a estrutura a seguir.

```
"xa-resources" : [{
            "name": "XASQL",
```

```
            "secret-manager-arn": "arn:aws:secrets:1234:us-east-1:secret:123456",
            "module": "${s3-source}/xa/ESPGSQLXA64.so"
        }]
```

name

Obrigatório. Especifica o nome do recurso XA.

secret-manager-arn

Especifica o nome de recurso da Amazon (ARN) para o segredo do que contém as credenciais para se conectar ao banco de dados.

módulo

Especifica a localização do arquivo executável do módulo de switch RM. Para obter mais informações, consulte Planejando e projetando XARs na documentação do Micro Focus Enterprise Server.

# Referência de definição do conjunto de dados do AWS Mainframe Modernization

Se seu aplicativo exigir mais do que alguns conjuntos de dados para processamento, inseri-los um por um no console do AWS Mainframe Modernization é ineficiente. Em vez disso, recomendamos que crie um arquivo JSON para especificar cada conjunto de dados. Diferentes tipos de conjuntos de dados são especificados de forma diferente no JSON, embora muitos parâmetros sejam comuns. Este documento descreve os detalhes do JSON necessários para importar diferentes tipos de conjuntos de dados.

> **Note**
>
> Antes de importar qualquer conjunto de dados, você deve transferir os conjuntos de dados do mainframe para o. AWS Em seguida, você deve garantir que os conjuntos de dados sejam convertidos do formato de mainframe para um formato que AWS possa ser usado. Se necessário, transforme os dados conforme necessário e armazene os conjuntos de dados transformados no Amazon S3. Especifique o nome do bucket e da pasta no arquivo JSON de definição do conjunto de dados.
> Se você estiver usando o mecanismo de tempo de execução da Micro Focus, poderá usar o utilitário `DFCONV` para converter os conjuntos de dados. Incluímos esse utilitário em

nossas imagens do Micro Focus Enterprise Developer e Enterprise Server. Para obter mais informações, consulte DFCONV Batch File Conversion na documentação do Micro Focus Enterprise Developer.

Tópicos



## Propriedades gerais

Vários parâmetros são comuns a todos os conjuntos de dados. Esses parâmetros abrangem as seguintes áreas:

- Informações sobre o conjunto de dados (`datasetName`,`datasetOrg`,`recordLength`,`encoding`)
- Informações sobre o local de onde você está importando, ou seja, o local de origem do conjunto de dados. Esse não é o local no mainframe. É o caminho para o local do Amazon S3 no qual você fez o upload do conjunto de dados (`externalLocation`).
- Informações sobre o local para o qual você está importando, ou seja, o local de destino do conjunto de dados. Esse local é um banco de dados ou um sistema de arquivos, dependendo do seu mecanismo de tempo de execução. (`storageType` e `relativePath`).
- Informações sobre o tipo de conjunto de dados (tipo específico de conjunto de dados, formato, codificação etc.).

Cada definição de conjunto de dados tem a mesma estrutura JSON. O exemplo de JSON a seguir mostra todos esses parâmetros comuns.

```
{
    "dataSet": {
        "storageType": "Database",
        "datasetName": "MFI01V.MFIDEMO.BNKACC",
        "relativePath": "DATA",
```

```
        "datasetOrg": {
            "type": {
                type-specific properties
                    ...
            },
        },
    },
}
```

As propriedades a seguir são comuns a todos os conjuntos de dados.

StorageType

Obrigatório. Aplica-se ao local de destino. Especifica se o conjunto de dados é armazenado em um banco de dados ou em um sistema de arquivos. Os valores possíveis são `Database` ou `FileSystem`.

- Mecanismo de tempo de execução AWS Blu Age: sistemas de arquivos não são suportados. Você deve usar um banco de dados.
- Mecanismo de tempo de execução da Micro Focus: bancos de dados e sistemas de arquivos são suportados. Você pode usar o Amazon Relational Database Service ou o Amazon Aurora para bancos de dados, e o Amazon Elastic File System ou o Amazon FSx for Lustre para sistemas de arquivos.

datasetName

Obrigatório. Especifica o nome totalmente qualificado do conjunto de dados conforme ele aparece no mainframe.

Caminho relativo

Obrigatório. Aplica-se ao local de destino. Especifica a localização relativa do conjunto de dados no banco de dados ou no sistema de arquivos.

Conjunto de dados Org

Obrigatório. Especifica o tipo de conjunto de dados. Os valores possíveis são `vsam`, `gdg`, `ps`, `po` ou `unknown`.

- Mecanismo de tempo de execução AWS Blu Age: somente conjuntos de dados do tipo VSAM são suportados.
- Mecanismo de tempo de execução da Micro Focus: conjuntos de dados do tipo VSAM, GDG, PS, PO ou Unknown são compatíveis.

> **Note**
>
> Se seu aplicativo exigir arquivos que não sejam arquivos de dados COBOL, mas sejam PDF ou outros arquivos binários, você poderá especificá-los da seguinte forma:

```
"datasetOrg": {
            "type": PS {
                "format": U
            },
```

## Formato de solicitação de conjunto de dados de amostra para VSAM

- Mecanismo de tempo de execução AWS Blu Age: suportado.
- Mecanismo de tempo de execução Micro Focus: compatível.

Se você estiver importando conjuntos de dados do VSAM, especifique `vsam` como o `datasetOrg`. Seu JSON deve ser semelhante ao exemplo a seguir:

```
{
    "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"
}
```

As propriedades a seguir são compatíveis com conjuntos de dados VSAM.

encoding

Obrigatório. Especifica a codificação do conjunto de caracteres do conjunto de dados. Os valores possíveis são ASCII (`A`), EBCDIC (`E`) e Unknown (`?`).

format

Obrigatório. Especifica o tipo do conjunto de dados VSAM e o formato do registro.

- AWSMecanismo de tempo de execução Blu Age: os valores possíveis são ESDS (`ES`), KSDS (`KS`) e RRDS (`RR`). O formato do registro pode ser fixo ou variável.
- Mecanismo de tempo de execução da Micro Focus: os valores possíveis são ESDS (`ES`), KSDS (`KS`) e RRDS (`RR`). A definição do VSAM inclui o formato do registro, portanto, você não precisa especificá-lo separadamente.

primaryKey

Aplica-se somente aos conjuntos de dados VSAM KSDS. Especifica a chave primária. Consiste no nome da chave primária, no deslocamento da chave e no comprimento da chave. O `name` é opcional; `offset` e `length` são obrigatórios.

recordLength

Obrigatório. Especifica o tamanho de um registro. Para formatos de registro de tamanho fixo, esses valores devem corresponder.

- O mecanismo de tempo de execução AWS Blu Age: para VSAM ESDS, KSDS e RRDS, `min` é opcional e obrigatório `max`.
- Mecanismo de tempo de execução da Micro Focus: `min` e `max` são obrigatórios.

Localização externa

Obrigatório. Especifica o local de origem: ou seja, o bucket do Amazon S3 em que você fez o upload do conjunto de dados.

### Propriedades específicas do motor Blu Age

O mecanismo de tempo de execução do AWS Blu Age suporta compactação para conjuntos de dados VSAM. O exemplo a seguir mostra como você pode especificar essa propriedade no JSON.

```
{
    common properties
        ...
        "datasetOrg": {
            "vsam": {
                common properties
                ...
                "compressed": boolean,
                common properties
                ...
            }
        }
}
```

Especifique a propriedade de compactação da seguinte forma:

compression

Opcional. Especifica se os índices desse conjunto de dados são armazenados como valores compactados. Se você tiver um grande conjunto de dados (normalmente > 100 Mb), considere definir esse sinalizador como`true`.

## Formato de solicitação de conjunto de dados de amostra para o GDG Base

- Mecanismo de tempo de execução AWS Blu Age: não suportado.
- Mecanismo de tempo de execução Micro Focus: compatível.

Se você estiver importando conjuntos de dados de base do GDG, especifique `gdg` como o `datasetOrg`. Seu JSON deve ser semelhante ao exemplo a seguir:

```
{
    "storageType": "Database",
    "datasetName": "AWS.M2.GDG",
    "relativePath": "DATA",
    "datasetOrg": {
```

```
        "gdg": {
            "limit": "3",
            "rollDisposition": "Scratch and No Empty"
        }
    }
}
```

As propriedades a seguir são compatíveis com conjuntos de dados básicos do GDG.

limite

Obrigatório. Especifica o número de gerações ativas ou vieses. Para um cluster base do GDG, o máximo é 255.

Disposição do rolo

Opcional. Especifica como lidar com conjuntos de dados de geração quando o máximo é atingido ou excedido. Os valores possíveis são `No Scratch and No Empty`, `Scratch and No Empty`, `Scratch and Empty`, ou `No Scratch and Empty`. O padrão é `Scratch and No Empty`.

## Formato de solicitação de conjunto de dados de amostra para as gerações PS ou GDG

- Mecanismo de tempo de execução AWS Blu Age: não suportado.
- Mecanismo de tempo de execução Micro Focus: compatível.

Se você estiver importando conjuntos de dados das gerações PS ou GDG, especifique `ps` como o `datasetOrg`. Seu JSON deve ser semelhante ao exemplo a seguir:

```
{
    "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"
}
}
```

As propriedades a seguir são compatíveis com conjuntos de dados das gerações PS ou GDG.

format

Obrigatório. Especifica o formato dos registros do conjunto de dados. Os valores possíveis são `F`, `FA`, `FB`, `FBA`, `FBM`, `FBS`, `FM`, `FS`, `LSEQ`, `U`, `V`, `VA`, `VB`, `VBA`, `VBM`, `VBS`, `VM` e `VS`.

encoding

Obrigatório. Especifica a codificação do conjunto de caracteres do conjunto de dados. Os valores possíveis são ASCII (`A`), EBCDIC (`E`) e Unknown (`?`)

Duração do registro

Obrigatório. Especifica o tamanho de um registro. Você deve especificar o tamanho mínimo (`min`) e máximo (`max`) do registro. Para formatos de registro de tamanho fixo, esses valores devem corresponder.

Localização externa

Obrigatório. Especifica o local de origem: ou seja, o bucket do Amazon S3 em que você fez o upload do conjunto de dados.

## Formato de solicitação de conjunto de dados de amostra para PO

Se você estiver importando conjuntos de dados PO, especifique `po` como o `datasetOrg` Seu JSON deve ser semelhante ao exemplo a seguir:

```
{
    "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/"
}
}
```

As propriedades a seguir são suportadas para conjuntos de dados PO.

format

Obrigatório. Especifica o formato dos registros do conjunto de dados. Os valores possíveis são `F`, `FA`, `FB`, `FBA`, `FBM`, `FBS`, `FM`, `FS`, `LSEQ`, `U`, `V`, `VA`, `VB`, `VBA`, `VBM`, `VBS`, `VM` e `VS`.

encoding

Obrigatório. Especifica a codificação do conjunto de caracteres do conjunto de dados. Os valores possíveis são ASCII (`A`), EBCDIC (`E`) e Unknown (`?`).

memberFileExtensions

Obrigatório. Especifica uma matriz contendo uma ou mais extensões de nome de arquivo, permitindo que você especifique quais arquivos serão incluídos como membro do PDS.

Duração do registro

Opcional. Especifica o tamanho de um registro. Tanto o tamanho mínimo (`min`) quanto o máximo (`max`) do registro são opcionais. Para formatos de registro de tamanho fixo, esses valores devem corresponder.

Localização externa

Obrigatório. Especifica o local de origem: ou seja, o bucket do Amazon S3 em que você fez o upload do conjunto de dados.

> **Note**
>
> A implementação atual do mecanismo de tempo de execução da Micro Focus adiciona entradas PDS como conjuntos de dados dinâmicos.

# Ambientes de tempo de execução gerenciados no AWS Mainframe Modernization

Se você é novo no AWS Mainframe Modernization, consulte os tópicos a seguir para começar:

- O que é modernização AWS do mainframe?
- Configurando a modernização AWS do mainframe
- Introdução à modernização AWS do mainframe
- Tutorial: Tempo de execução gerenciado para AWS Blu Age
- Tutorial: Tempo de execução gerenciado para Micro Focus

Um ambiente de tempo de execução no AWS Mainframe Modernization é uma combinação nomeada de recursos computacionais AWS, um mecanismo de tempo de execução e os detalhes de configuração que você especifica. O ambiente de execução hospeda um ou mais aplicativos. Um aplicativo no AWS Mainframe Modernization contém uma carga de trabalho de mainframe migrada. Você pode escolher o mecanismo de tempo de execução para os ambientes que você cria. Escolha AWS Blu Age se estiver usando o padrão de refatoração automatizado e Micro Focus se estiver usando o padrão de replataforma. Você também pode escolher a quantidade de recursos computacionais adequados para seu aplicativo e, opcionalmente, associar armazenamento a ambientes de tempo de execução AWS A modernização do mainframe permite que você monitore seu ambiente de tempo de execução com CloudWatch métricas e registros da Amazon.

Tópicos

- Crie um ambiente de tempo de execução do AWS Mainframe Modernization
- Atualizar um ambiente de tempo de execução do AWS Mainframe Modernization
- Interrompa um ambiente de execução do AWS Mainframe Modernization
- Reinicie um ambiente de execução de modernização de AWS mainframe
- Excluir um ambiente de execução do AWS Mainframe Modernization

# Crie um ambiente de tempo de execução do AWS Mainframe Modernization

Use o console do AWS Mainframe Modernization para criar um ambiente do AWS Mainframe Modernization.

Estas instruções pressupõem que você tenha concluído as etapas em Configurando a modernização AWS do mainframe.

## Criar um ambiente de tempo de execução

Para criar um ambiente de tempo de execução

1. Abra o console do AWS Mainframe Modernization em https://console.aws.amazon.com/m2/.
2. No seletor Região da AWS, escolha a região onde você deseja criar o ambiente.
3. Na guia Ambientes escolha Criar ambiente.
4. Na página Especificações da inferência, forneça as seguintes informações:
    a. Na seção Nome e descrição, insira um nome para o ambiente.
    b. (Opcional) No campo Descrição do ambiente, insira uma descrição para o ambiente. Essa descrição pode ajudar você e outros usuários a identificar a finalidade do ambiente de tempo de execução.
    c. Na seção Opções do motor, escolha Blu Age para refatoração automatizada ou Micro Focus para reformulação de plataforma.
    d. Escolha uma versão para o mecanismo que você selecionou.
    e. (Opcional) Na seção Tags, escolha Adicionar nova tag para adicionar uma ou mais tags de ambiente ao seu ambiente. Uma tag de ambiente é um rótulo de atributo personalizado que ajuda a organizar e gerenciar seus AWS recursos.
    f. Escolha Próximo.
5. Na página Especificar configurações, forneça as seguintes informações:
    a. Na seção Disponibilidade, escolha Ambiente de tempo de execução autônomo ou Cluster de alta disponibilidade.

O padrão de disponibilidade determina o quão disponível seu aplicativo estará quando for executado. Autônomo é bom para fins de desenvolvimento. A alta disponibilidade é para aplicativos que devem estar sempre disponíveis.

b. Em Recursos, escolha um tipo de instância e a capacidade desejada.

Esses recursos são as instâncias do Amazon EC2 gerenciadas pelo AWS Mainframe Modernization que hospedarão seu ambiente de execução. Ambientes de tempo de execução autônomos oferecem duas opções para o tipo de instância e permitem apenas uma instância. Ambientes de tempo de execução de alta disponibilidade oferecem duas opções para o tipo de instância e permitem até duas instâncias.

Para obter mais informações, consulte Tipos de instância do Amazon EC2 e entre em contato com um especialista em mainframe AWS para obter orientação.

6. Na seção Configurações da rede, faça o seguinte:

a. Se você quiser que os aplicativos sejam acessíveis ao público, escolha Permitir que os aplicativos implantados nesse ambiente sejam acessíveis ao público.

b. Escolha uma nuvem privada virtual (VPC).

c. Se você estiver usando o padrão de alta disponibilidade, escolha duas ou mais sub-redes. Se você estiver usando o padrão autônomo com o mecanismo AWS Blu Age, escolha duas ou mais sub-redes. Se você estiver usando o padrão autônomo com o mecanismo da Micro Focus, pode especificar uma sub-rede.

d. Escolha um grupo de segurança para a VPC que você selecionou.

> **Note**
>
> AWS Mainframe Modernization cria um Network Load Balancer para você distribuir conexões ao seu ambiente de tempo de execução. Verifique se as regras de entrada do grupo de segurança permitem o acesso de um endereço IP à porta especificada na `listener` propriedade da definição do aplicativo. Para obter mais informações, consulte Alvos de registro no Guia do usuário para balanceadores de carga de rede.

e. No campo Chave KMS, escolha Personalizar configurações de criptografia se quiser usar um cliente gerenciado AWS KMS key. Para ter mais informações, consulte Criptografia de dados em repouso para o serviço de modernização AWS de mainframe.

> **Note**
>
> Por padrão, o AWS Mainframe Modernization criptografa seus dados com dados AWS KMS key que o AWS Mainframe Modernization possui e gerencia para você. No entanto, é possível optar por usar um sistema gerenciado pelo cliente AWS KMS key.

f. (Opcional) Escolha um AWS KMS key por nome ou nome de recurso da Amazon (ARN) da Amazon. Como alternativa, escolha Criar um AWS KMS key para acessar o console AWS KMS e criar um novo AWS KMS key.

g. Escolha Próximo.

7. (Opcional) Na página Anexar armazenamento, escolha um ou mais sistemas de arquivos Amazon EFS ou Amazon FSx e, em seguida, escolha Próximo.

8. Na seção Janela de manutenção, escolha quando você deseja aplicar as alterações pendentes no ambiente.

   - Se você escolher Sem preferência, o AWS Mainframe Modernization escolherá uma janela de manutenção otimizada para você.
   - Se você quiser especificar uma janela de manutenção específica, escolha Selecionar nova janela de manutenção. Em seguida, escolha um dia da semana, uma hora de início e uma duração para a janela de manutenção.

   Para obter mais informações sobre a janela de manutenção, consulte Janela de manutenção do AWS Mainframe Modernization.

   Escolha Próximo.

9. Na página Revisar e criar, analise as informações fornecidas e selecione Criar fonte de dados.

# Atualizar um ambiente de tempo de execução do AWS Mainframe Modernization

Use o console do AWS Mainframe Modernization para atualizar um ambiente de tempo de execução do AWS Mainframe Modernization. Você pode atualizar a versão secundária do mecanismo de tempo de execução ou o tipo de instância que hospeda o ambiente de tempo de execução. Você

pode escolher se deseja aplicar as atualizações imediatamente ou durante a janela de manutenção preferida.

Estas instruções supõem que você tenha concluído as etapas em Configurando a modernização AWS do mainframe.

## Atualizar um ambiente de tempo de execução

Para atualizar um ambiente de execução

1. Abra o console do AWS Mainframe Modernization em https://console.aws.amazon.com/m2/.
2. No Região da AWS seletor, escolha a região em que o ambiente que você deseja atualizar foi criado.
3. Na página Ambientes, escolha o ambiente a ser atualizado.
4. Na página de visão geral do ambiente, escolha Ações e depois Trocar URLs do ambiente.
5. Faça uma ou todas as alterações a seguir:
   - Na seção Opções do mecanismo, escolha a versão do mecanismo que você deseja.
   - Na seção Recursos, escolha o tipo de instância desejado.
   - Na seção Janela de manutenção, escolha o dia, a hora e a duração desejados.

   > **Note**
   >
   > As únicas alterações que você pode optar por aplicar durante a janela de manutenção são as alterações na versão do motor. Você deve aplicar todas as outras alterações imediatamente.

6. Escolha Próximo.
7. se desejar aplicar as alterações imediatamente ou durante a próxima janela de manutenção. Escolha Atualizar ambiente.

Se você escolher Imediatamente, verá uma mensagem quando a atualização do ambiente for concluída.

# Janela de manutenção do AWS Mainframe Modernization

Cada ambiente de execução tem uma janela de manutenção semanal de uma hora. Todas as alterações do sistema são aplicadas durante esse período. A janela de manutenção é a sua chance de controlar quando ocorrem as modificações e a aplicação de patches de software. Se um evento de manutenção estiver programado para uma determinada semana, ele começará durante essa janela de manutenção de uma hora. A maioria dos eventos de manutenção também é concluída durante a janela de manutenção de uma hora, embora eventos de manutenção maiores possam levar mais de uma hora para serem concluídos.

A janela de manutenção de uma hora é selecionada aleatoriamente em um bloco de 8 horas por região. Se você não especificar uma janela de manutenção ao criar ou modificar a execução de um ambiente de execução, o AWS Mainframe Modernization atribuirá uma janela de manutenção de 1 hora em um dia da semana selecionado aleatoriamente.

AWS Mainframe Modernization consome alguns dos recursos da instância de ambiente enquanto a manutenção é aplicada. Você poderá observar um impacto mínimo no desempenho ou algumas interrupções nos aplicativos durante a manutenção.

A tabela a seguir lista os blocos de tempo de cada região dos quais as janelas de manutenção padrão são atribuídas.

| Nome da região | Região | Bloco de hora |
|---|---|---|
| Leste dos EUA (Norte da Virgínia) | us-east-1 | 3h às 11h (UTC) |
| Oeste dos EUA (Oregon) | us-west-2 | 6h às 14h (UTC) |
| Asia Pacific (Mumbai) | ap-south-1 | 6h às 14h (UTC) |
| Ásia-Pacífico (Singapura) | ap-southeast-1 | De 14:00 a 22:00 UTC |
| Asia Pacific (Sydney) | ap-southeast-2 | De 12:00 a 20:00 UTC |
| Asia Pacific (Tokyo) | ap-northeast-1 | De 13:00 a 21:00 UTC |
| Canada (Central) | ca-central-1 | 3h às 11h (UTC) |

| Nome da região | Região | Bloco de hora |
|---|---|---|
| Europa (Frankfurt) | eu-central-1 | De 21:00 a 05:00 UTC |
| Europe (Ireland) | eu-west-1 | De 22:00 a 06:00 UTC |
| Europe (London) | eu-west-2 | De 22:00 a 06:00 UTC |
| Europe (Paris) | eu-west-3 | De 23:59 a 07:29 UTC |
| América do Sul (São Paulo) | sa-east-1 | De 00:00 a 08:00 UTC |

# Interrompa um ambiente de execução do AWS Mainframe Modernization

Use o console do AWS Mainframe Modernization para interromper um ambiente de tempo de execução do AWS Mainframe Modernization Quando você interrompe um ambiente, as implantações atuais do aplicativo são mantidas e você não será cobrado pelo ambiente até que o ambiente seja reiniciado.

Estas instruções supõem que você tenha concluído as etapas em Configurando a modernização AWS do mainframe.

## Interromper um ambiente de execução

Se você precisar interromper um ambiente de execução do AWS Mainframe Modernization, siga etapas semelhantes às da seção do ambiente de atualização.

Use o console do AWS Mainframe Modernization para interromper um ambiente de tempo de execução do AWS Mainframe Modernization Quando você interrompe um ambiente, as implantações atuais do aplicativo são mantidas e você não será cobrado pelo ambiente até que o ambiente seja reiniciado.

### Interromper um ambiente de execução

Para interromper um ambiente de execução do AWS Mainframe Modernization, siga etapas semelhantes às da seção de atualização do ambiente.

> **Note**
>
> Você deve interromper todos os aplicativos antes de interromper o ambiente.

Para interromper um ambiente de execução

1. Abra o console do AWS Mainframe Modernization em https://console.aws.amazon.com/m2/.
2. No Região da AWS seletor, escolha a região em que o ambiente que você deseja interromper foi criado.
3. Na página Ambientes, escolha o ambiente a ser interrompido.
4. Na página de visão geral do ambiente, escolha Ações e depois Trocar URLs do ambiente.
5. Na página Editar ambiente, encontre a seção Recursos e atualize a capacidade desejada para zero.

   > **Note**
   >
   > Para interromper um ambiente, você só pode optar por parar imediatamente.

6. Escolha Próximo.
7. Em Quando aplicar essas alterações, escolha Imediatamente. Escolha Atualizar ambiente.

Você vê uma mensagem quando a capacidade do ambiente é atualizada.

# Reinicie um ambiente de execução de modernização de AWS mainframe

Use o console do AWS Mainframe Modernization para reiniciar um ambiente de tempo de execução do AWS Mainframe Modernization Quando você reinicia um ambiente de tempo de execução, a cobrança do ambiente é retomada.

## Reiniciar um ambiente de execução

Para reiniciar um ambiente de execução do AWS Mainframe Modernization, siga etapas semelhantes às da seção de interrupção do ambiente.

Para reiniciar um ambiente de execução

1. Abra o console do AWS Mainframe Modernization em https://console.aws.amazon.com/m2/.
2. No Região da AWS seletor, escolha a região em que o ambiente que você deseja reiniciar foi criado.
3. Na página Ambientes, escolha o ambiente a ser reiniciado.
4. Na página de visão geral do ambiente, escolha Ações e depois Trocar URLs do ambiente.

> **Note**
>
> A capacidade desejada para um ambiente autônomo só pode ser atualizada para 1. Para reiniciar um ambiente de execução, você só pode optar por reiniciar imediatamente.

5. Na página Editar ambiente, encontre a seção Recursos e atualize a capacidade desejada de zero para a capacidade necessária.
6. Escolha Próximo.
7. Em Quando aplicar essas alterações, escolha Imediatamente. Escolha Atualizar ambiente.

Você vê uma mensagem quando a capacidade do ambiente é atualizada e o ambiente é reiniciado.

# Excluir um ambiente de execução do AWS Mainframe Modernization

Use o console do AWS Mainframe Modernization para excluir um ambiente de tempo de execução do AWS Mainframe Modernization

Estas instruções supõem que você tenha concluído as etapas em Configurando a modernização AWS do mainframe.

## Excluir um ambiente de execução

Se você precisar excluir um ambiente de tempo de execução do AWS Mainframe Modernizatione, exclua primeiro todos os aplicativos implantados do ambiente. Você não pode excluir um ambiente de tempo de execução em que os aplicativos são implantados.

Para excluir um ambiente

1. Abra o console do AWS Mainframe Modernization em https://console.aws.amazon.com/m2/.
2. No Região da AWS seletor, escolha a região em que o ambiente que você deseja excluir foi criado.
3. Na página Ambientes, escolha o ambiente que você deseja excluir e, em seguida, escolha Ações e Excluir ambiente.
4. Na janela Excluir ambiente, insira `delete` para confirmar que você deseja excluir o ambiente de tempo de execução e escolha Excluir.

# Teste de aplicativos na modernização AWS do mainframe

AWS O teste de aplicativos está na versão prévia da modernização do AWS mainframe e está sujeito a alterações. Recomendamos o uso desse recurso somente com dados e aplicações de teste, e não em ambientes de produção.

AWS O Mainframe Modernization Application Testing fornece testes automatizados de equivalência funcional para seus projetos de migração.

Tópicos



## O que é o Application Testing do AWS Mainframe Modernization Application?

O AWS Application Testing está em versão de pré-visualização para o AWS Mainframe Modernization e está sujeito a alterações. Recomendamos o uso desse recurso somente com dados e aplicações de teste, e não em ambientes de produção.

Os testes afetam significativamente os projetos de migração. Ele pode consumir até 70% do tempo e do esforço do seu projeto de migração, modernização ou aumento. O AWS O Application Testing, um recurso do AWS Mainframe Modernization, fornece testes automatizados de equivalência funcional para suas aplicações migradas. O teste de equivalência funcional ajuda você a validar se as suas aplicações na Nuvem AWS são equivalentes às aplicações em seu mainframe. O AWS Application Testing compara automaticamente as alterações em conjuntos de dados, registros

de banco de dados e telas 3270 on-line entre seu mainframe e a AWS. Além disso, o Application Testing permite testes repetíveis, para que você possa executar seus cenários de teste várias vezes à medida que atualiza a arquitetura de destino, resolve problemas e avança em direção a uma aplicação totalmente migrada. Após a migração, você pode continuar usando o Application Testing para testes de regressão com o objetivo de garantir que as atualizações nos mecanismos de runtime ou em outros componentes não causem regressões. O Application Testing é econômico: os ambientes de teste de destino são criados usando os modelos do CloudFormation fornecidos pelo usuário, aproveitando os conceitos de infraestrutura como código (IaC). O Application Testing acelera os projetos de migração usando a elasticidade da nuvem. É possível executar cenários de teste independentes em quantos ambientes paralelos forem necessários, reduzindo as linha do tempo de teste.

Tópicos



## Você é um usuário iniciante do Application Testing?

Se estiver usando o Application Testing pela primeira vez, recomendamos que você leia as seguintes seções para começar:

- Conceitos do Application Testing
- Tutorial: Configurar o CardDemo

## Benefícios do Application Testing

O Application Testing oferece vários benefícios para ajudar você no processo de migração:

- Testes de aceleração, agilidade e flexibilidade
- Conceitos de teste “Grave uma vez no mainframe, reproduza várias vezes na AWS”

- Criação IaC de ambientes de destino por meio de modelos do CloudFormation fornecidos pelo usuário
- Altos graus de repetibilidade de testes
- Criado para a nuvem, com escalabilidade e elasticidade em mente
- Testes em grande escala com alto grau de automação
- Eficiência de custos

## Integração com AWS CloudFormation

O Application Testing usa a infraestrutura como código com o AWS CloudFormation. Essa opção de design simplifica e melhora sua experiência de teste. O AWS CloudFormation oferece autonomia e independência para definir a melhor infraestrutura para suas necessidades. Você pode selecionar ou definir vários parâmetros (tamanho da instância, instância do RDS, grupo de segurança ideal) de forma independente. É possível adicionar recursos, como uma fila do Amazon SQS necessária para que sua aplicação funcione adequadamente em condições de teste.

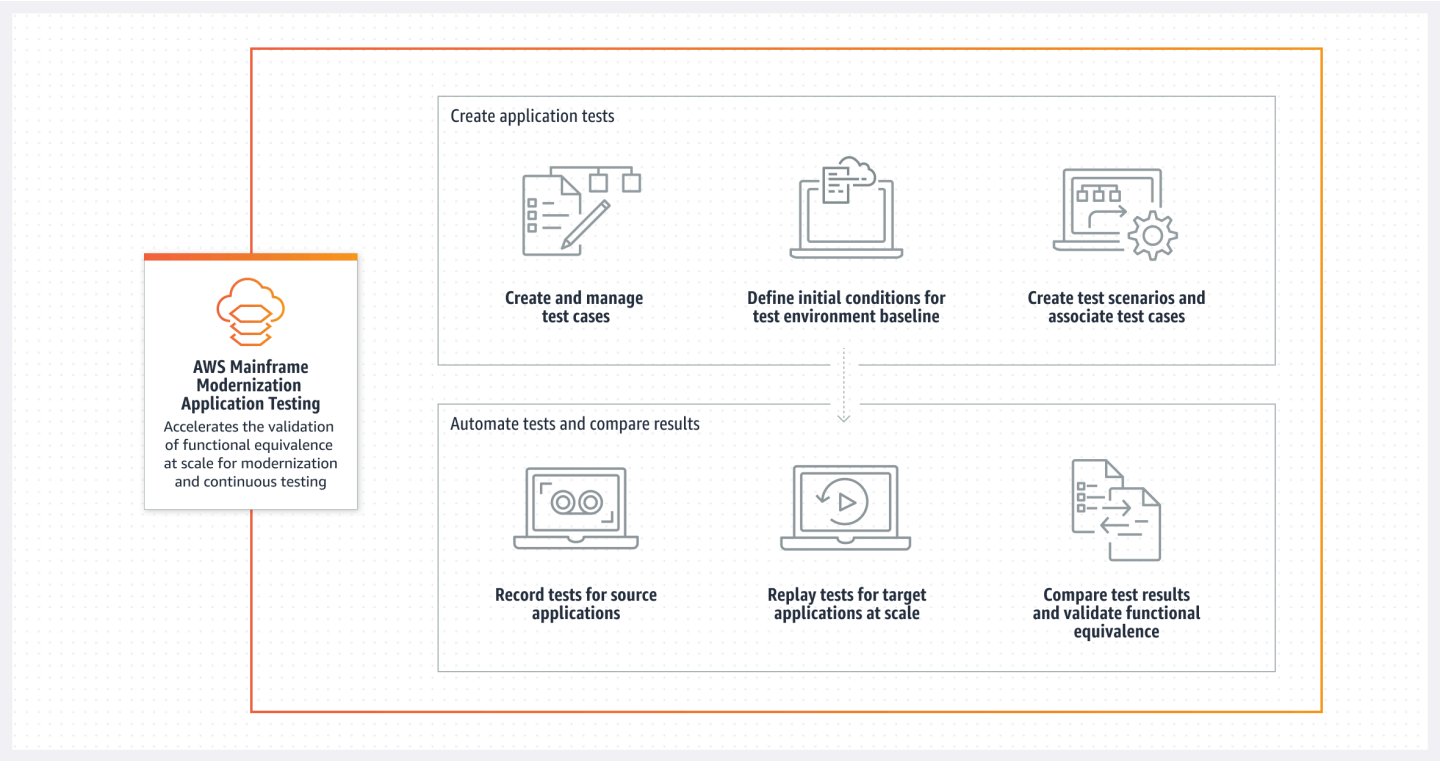
Nos modelos do AWS CloudFormation fornecidos para download, você notará alguns recursos comuns:

- O Application Testing cria uma pilha totalmente isolada, incluindo um ambiente de runtime e uma aplicação do AWS Mainframe Modernization, com suas próprias definições de rede e de segurança. Essa pilha isolada fornece resiliência, porque outros atores da mesma Conta da AWS não podem interferir na atividade de teste. Ela também evita situações em que os operadores do sistema modifiquem a VPC ou o grupo de segurança padrão, o que pode causar falhas nas atividades de teste.
- O grupo de segurança também permite que você controle o acesso externo aos recursos usados nos testes. Por exemplo, um banco de dados pode conter dados confidenciais.
- O isolamento total impede que outros atores que compartilham a VPC espionem o tráfego.
- Ele aprimora o desempenho. Por exemplo, a comunicação entre a aplicação do AWS Mainframe Modernization que o modelo cria e seu banco de dados do Amazon RDS ocorre em uma rede separada (uma VPC privada), o que evita que outros atores diminuam a velocidade do tráfego.

Recomendamos que você implemente esses recursos também nos modelos do AWS CloudFormation que criar.

# Como o Application Testing funciona

A figura a seguir é uma visão geral de como o teste de equivalência funcional no Application Testing funciona.




- É possível transferir dados de entrada da origem para a AWS usar o Transferência de Arquivos do AWS Mainframe Modernization ou suas ferramentas preferidas para transferência de dados do mainframe.
- Você executa a mesma lógica de negócios na origem e no destino.
- O Application Testing compara automaticamente os dados de saída (conjuntos de dados, alterações no banco de dados relacional, telas 3270 e interações do usuário) da origem e do destino. Depois de executar seu cenário de teste no mainframe, você captura os dados de saída e os transfere para a AWS. Depois, reproduz o cenário de teste no destino. O Application Testing compara automaticamente os dados de saída do teste executado na AWS com os dados de saída da origem. É possível ver rapidamente quais registros são idênticos, equivalentes, diferentes ou estão ausentes. Além disso, opcionalmente, é possível definir regras de equivalência, para que os registros que não sejam idênticos continuem tendo o mesmo significado comercial e sejam marcados como equivalentes.

O fluxo de trabalho a ser seguido no Application Testing consiste nas seguintes etapas:

1. Criar casos de teste. Os casos de teste são a menor unidade de ações de teste. Ao criar um caso de teste, você também identifica os tipos de dados a serem comparados que melhor representam a equivalência funcional entre a origem e o destino.
2. Criar cenários de teste. Os cenários de teste agrupam os casos de teste relacionados em uma sequência específica para execução.
3. Criar condições iniciais. As condições iniciais explicam como gravar um teste no mainframe e usar o AWS CloudFormation para criar automaticamente o mesmo estado na AWS.
4. Grave na origem e reproduza no destino. Capture os conjuntos de dados de entrada e saída no mainframe e faça upload deles para a AWS. Depois, reproduza o cenário de teste na AWS.
5. Compare os conjuntos de dados de origem e de destino. O Application Testing compara automaticamente os conjuntos de dados de saída da origem e do destino, para que você possa ver rapidamente o que está correto e o que não está.

Tanto a ação final de um cenário de teste quanto a meta de todo o processo é identificar discrepâncias entre as execuções de teste da origem e do destino. O Application Testing compara a versão de origem e a versão de destino dos dados capturados em todos os canais de interação durante a execução do teste. Ele também compara os estados finais dos dados relevantes (conforme definido nos casos de teste).

## Serviços relacionados

O Application Testing é um recurso do AWS Mainframe Modernization. Ele também usa a infraestrutura como código com o AWS CloudFormation para garantir a repetibilidade, a automação e uma boa relação custo-benefício dos testes. Para obter mais informações, consulte:

- AWS Mainframe Modernization
- AWS CloudFormation

## Acessar o Application Testing

É possível acessar o Application Testing no console do AWS Mainframe Modernization escolhendo Application Testing na navegação esquerda.

## Definição de preços do Application Testing

A definição de preços para o Application Testing pode ser encontrada em AWS Mainframe Modernization Pricing.

# AWS Conceitos de teste de aplicativos de modernização de mainframe

> AWS O teste de aplicativos está na versão prévia da modernização do AWS mainframe e está sujeito a alterações. Recomendamos o uso desse recurso somente com dados e aplicações de teste, e não em ambientes de produção.

AWS O teste de aplicativos usa termos que outros serviços de teste ou pacotes de software podem usar com um significado ligeiramente diferente. As seções a seguir explicam como o AWS Mainframe Modernization Application Testing usa essa terminologia.

Tópicos

- Gravação
- Reproduzir dados
- Dados de referência
- Gravar, reproduzir e comparar
- Diferenças
- Equivalências
- Aplicação de origem
- Aplicação de destino

## Caso de teste

Um caso de teste é a unidade de ação individual mais atômica em seu fluxo de trabalho de teste. Normalmente, um caso de teste é usado para representar uma unidade independente da lógica de negócios que modifica os dados. As comparações serão feitas para cada caso de teste. Os casos de teste são adicionados a um cenário de teste. Os casos de teste contêm metadados sobre os artefatos de dados (conjuntos de dados, bancos de dados) que o caso de teste modifica e sobre as funções de negócios que são acionadas durante a execução do caso de teste: trabalhos em lote, diálogos interativos 3270 e outros. Por exemplo, os nomes e as páginas de código dos conjuntos de dados.

Dados de entrada → Caso de teste → Dados de saída

Os casos de teste podem ser on-line ou do tipo em lote:

- Casos de teste on-line são casos de teste em que o usuário executa diálogos de tela interativos (3270) para ler, modificar ou produzir novos dados comerciais (registros de banco de dados e/ou conjuntos de dados).
- Casos de este em lote são casos de teste que exigem o envio de um lote para ler, processar e modificar ou produzir novos dados comerciais (conjuntos de dados e/ou registros de banco de dados).

## Cenário de teste

Os cenários de teste são uma série de casos de teste que são executados em ordem sequencial, um por um. A reprodução é feita em um nível de cenário de teste. Todos os casos de teste no cenário

de teste são executados no ambiente de teste de destino quando um cenário de teste é reproduzido. Se houver diferenças após comparar artefatos de teste de referência e de reprodução, as diferenças serão mostradas no nível do caso de teste.

Por exemplo, Cenário de teste A:

Caso de teste 1, caso de teste 2, caso de teste 3 e assim por diante.

## Projetos de teste

Os projetos de teste representam uma coleção de cenários de teste para alcançar o marco de teste desejado. Por exemplo, a migração de uma aplicação específica pode ser considerada como um único projeto de teste. O agrupamento de cenários de teste em projetos de teste permite que os gerentes de teste acompanhem o status do projeto de teste, incluindo os testes aprovados/reprovados.

## Condição inicial

Uma condição inicial contém um conjunto de recursos (computação, datastore e outros) que você deve criar e dados de aplicações que devem ser restaurados nesses recursos criados antes de executar cenários de teste. Isso cria a linha de base do ambiente de teste de destino. Ele permite que você forneça um AWS CloudFormation modelo. Você usa o modelo para criar o ambiente de teste de destino e, opcionalmente, a extração DDL do banco de dados de origem se o cenário de teste modificar os registros do banco de dados. Cada cenário de teste será associado a uma condição inicial. Uma condição inicial pode ser associada a vários cenários de teste. Para garantir a repetibilidade com consistência dos resultados e evitar falsos positivos devido aos dados já alterados, é necessário restaurar as condições iniciais antes da execução de cada cenário de teste.

Para cenários de teste que contenham casos de teste que modificam registros do banco de dados, a condição inicial também faz referência a uma exportação DDL dos esquemas e tabelas do banco de dados de origem.

## Gravar (capturar)

As gravações são feitas em um nível de cenário de teste. Durante o registro, você deve fornecer um local do Amazon S3 que contenha os artefatos, conjuntos de dados e diários CDC do banco de dados relacional do mainframe de origem com os quais serão comparados. Eles serão considerados como dados de referência do mainframe de origem. Durante a reprodução, os dados da reprodução

gerados serão comparados com os dados de referência gravados para garantir a equivalência da aplicação.

## Reproduzir

As reproduções são feitas em um nível de cenário de teste. Durante a repetição, o AWS Mainframe Modernization Application Testing usa o CloudFormation script referenciado na condição inicial associada para criar o ambiente de teste de destino e executar o aplicativo. Os conjuntos de dados e as gravações do banco de dados que são modificados durante a reprodução são capturados e comparados com os dados de referência do mainframe. Normalmente, você gravará no mainframe uma vez e depois reproduzirá várias vezes, até que a equivalência funcional seja alcançada.

## Compare

As comparações são feitas automaticamente após o término com êxito de uma reprodução. Durante as comparações, os dados referenciados dos quais você fez upload e capturou durante a fase de gravação são comparados com os dados da reprodução gerados durante a fase de reprodução. As comparações acontecem em um nível de caso de teste individual para conjuntos de dados, registros de banco de dados e telas on-line separadamente.

## Comparações de bancos de dados

O Application Testing emprega uma funcionalidade de correspondência de progresso de estado ao comparar alterações nas gravações do banco de dados entre as aplicações de origem e de destino. A correspondência do progresso do estado compara as diferenças em cada instrução INSERT, UPDATE e DELETE de execução individual, diferentemente da comparação das linhas da tabela no final do processo. A correspondência do progresso do estado é mais eficiente do que as alternativas, fornecendo comparações mais rápidas e precisas ao comparar somente os dados alterados e ao detectar erros de autocorreção no fluxo da transação. Usando a tecnologia captura de dados de alteração (CDC), o Application Testing pode detectar alterações individuais no banco de dados relacional e compará-las entre a origem e o destino.

As alterações do banco de dados relacional são geradas na origem e no destino pelo código da aplicação testada usando instruções de linguagem de modificação de dados (DML), como SQL INSERT, UPDATE ou DELETE, mas também indiretamente quando a aplicação está usando procedimentos armazenados, quando os acionadores do banco de dados são definidos em algumas tabelas ou quando CASCADE DELETE é usada para garantir a integridade referencial, acionando automaticamente exclusões adicionais.

# Comparações de conjuntos de dados

O Application Testing compara automaticamente os conjuntos de dados de referência e de reprodução produzidos nos sistemas de origem (gravação) e de destino (reprodução).

Como comparar conjuntos de dados:

1. Comece com os mesmos dados de entrada (conjuntos de dados, banco de dados) na origem e no destino.
2. Execute seus casos de teste no sistema de origem (mainframe).
3. Capture os conjuntos de dados produzidos e faça upload deles para um bucket do Amazon S3. Você pode transferir conjuntos de dados de entrada da fonte para AWS usar diários, telas e conjuntos de dados do CDC.
4. Especifique a localização do bucket do Amazon S3 onde foi feito upload dos conjuntos de dados do mainframe quando você gravou o caso de teste.

Após a conclusão da reprodução, o Application Testing comparará automaticamente os conjuntos de dados de referência e de destino de saída, mostrando se os registros são idênticos, equivalentes, diferentes ou se estão ausentes. Por exemplo, campos de data relativos ao momento da execução da workload (dia + 1, final do mês atual etc.) são automaticamente considerados equivalentes. Além disso, opcionalmente, é possível definir regras de equivalência, para que os registros que não sejam idênticos continuem tendo o mesmo significado comercial e sejam marcados como equivalentes.

# Status da comparação

O Application Testing usa os seguintes status de comparação: IDÊNTICO, EQUIVALENTE e DIFERENTE.

IDÊNTICO

Os dados de origem e de destino são exatamente os mesmos.

EQUIVALENTE

Os dados de origem e de destino contêm falsas diferenças consideradas equivalências, como datas ou timestamps que não afetam a equivalência funcional quando são relativos ao momento da execução da workload. É possível definir regras de equivalência para identificar quais são essas diferenças. Quando todos os cenários de teste reproduzidos em comparação com seus

cenários de teste de referência mostram o status de IDÊNTICO ou EQUIVALENTE, seu cenário de teste prova equivalência funcional.

DIFERENTE

Os dados de origem e de destino contêm diferenças, como um número diferente de registros em um conjunto de dados ou valores diferentes no mesmo registro.

## Regras de equivalência

Um conjunto de regras para identificar falsas diferenças que podem ser consideradas resultados equivalentes. O teste de equivalência funcional offline (OFET) inevitavelmente causa diferenças em alguns resultados entre os sistemas de origem e de destino. Por exemplo, os timestamps de atualização são diferentes por design. As regras de equivalência explicam como se ajustar a essas diferenças e evitar falsos positivos no momento da comparação. Por exemplo, se uma data for runtime + 2 dias em uma coluna de dados específica, a regra de equivalência a descreve e aceita uma hora no sistema de destino que seja runtime no destino + 2 dias, em vez de um valor que seja estritamente igual à mesma coluna na gravação de referência.

## Comparação do conjunto de dados do estado final

O estado final dos conjuntos de dados que foram criados ou modificados, incluindo todas as alterações ou atualizações feitas nos conjuntos de dados a partir do estado inicial. Para os conjuntos de dados, o Application Testing analisa os registros nesses conjuntos de dados no final da execução de um caso de teste e compara os resultados.

## Comparações de bancos de dados de progresso de estado

Comparações das alterações feitas nos registros do banco de dados como uma sequência de instruções DML individuais (Delete, Update, Insert). O Application Testing compara alterações individuais (inserir, atualizar ou excluir a linha de uma tabela) do banco de dados de origem com o banco de dados de destino e identifica as diferenças para cada alteração individual. Por exemplo, uma instrução INSERT individual pode ser usada para inserir em uma tabela uma linha com valores diferentes no banco de dados de origem em comparação com o banco de dados de destino.

## Equivalência funcional (FE)

Dois sistemas são considerados funcionalmente equivalentes se produzirem os mesmos resultados em todas as operações observáveis, dados os mesmos dados de entrada. Por exemplo, duas

aplicações são consideradas funcionalmente equivalentes se os mesmos dados de entrada produzirem dados de saída idênticos (por meio de telas, alterações no conjunto de dados ou alterações no banco de dados).

## Comparações online de telas 3270

Compara a saída das telas do mainframe 3270 com a saída das telas web de aplicativos modernizados quando o sistema de destino está sendo executado sob o tempo de execução do AWS Blu Age no. Nuvem AWS E compara a saída das telas 3270 do mainframe com as telas 3270 da aplicação com hospedagem redefinida quando o sistema de destino está sendo executado no runtime do Micro Focus na Nuvem AWS.

## Gravação

A ação de restaurar um estado de dados conhecido e, depois, capturar ou gravar dados de referência de um cenário de teste de referência (para um ou vários casos de teste sequencialmente) em um sistema de origem.

## Reproduzir dados

Os dados da reprodução são usados para descrever os dados gerados pela reprodução de um cenário de teste no ambiente de teste de destino. Por exemplo, os dados de repetição são gerados quando um cenário de teste está sendo executado em um aplicativo de serviço de modernização de AWS mainframe. Os dados da reprodução são então comparados com os dados de referência capturados da origem. Toda vez que uma workload é reproduzida no ambiente de destino, uma nova geração de dados de reprodução é feita.

## Dados de referência

Os dados de referência são usados para descrever os dados capturados no mainframe de origem. É a referência com a qual os dados gerados pela reprodução (destino) serão comparados. Normalmente, para cada gravação no mainframe que cria dados de referência, haverá muitas reproduções. Isso ocorre porque os usuários normalmente capturam o estado correto da aplicação no mainframe e reproduzem os casos de teste na aplicação modernizada de destino para validar a equivalência. Se forem encontrados bugs, eles serão corrigidos e os casos de teste serão reproduzidos novamente. Frequentemente, vários ciclos de reprodução, correção de bugs e nova reprodução para validar a ocorrência. Isso é chamado de paradigma de teste de captura única e várias reproduções.

## Gravar, reproduzir e comparar

O Application Testing opera em três etapas:

- Gravar: captura os dados referidos que foram criados no mainframe para cada caso de teste de um cenário de teste. Isso pode incluir telas 3270 on-line, conjuntos de dados e registros de banco de dados.
  - Para telas 3270 on-line, você deve usar o emulador de terminal do Blu Insights para capturar sua workload de origem. Para obter mais informações, consulte a documentação do Blu Insights.
  - Para conjuntos de dados, você precisará capturar os conjuntos de dados produzidos por cada caso de teste no mainframe usando ferramentas comuns, como FTP ou o serviço de transferência de conjuntos de dados que faz parte da Modernização do AWS Mainframe.
  - Para alterações no banco de dados, use a documentação Replicação de dados do AWS Mainframe Modernization com a Precisely para capturar e gerar diários da CDC contendo as alterações.
- Reproduzir: o cenário de teste é reproduzido no ambiente de destino. Todos os casos de teste especificados na execução do cenário de teste. Os tipos de dados especificados criados pelos casos de teste individuais, como conjuntos de dados, alterações no banco de dados relacional ou telas 3270, serão capturados com a automação. Esses dados são conhecidos como dados da reprodução e serão comparados com os dados de referência capturados durante a fase de gravação.

  > **Note**
  >
  > As alterações no banco de dados relacional exigirão opções de configuração específicas do DMS em seu modelo de condição inicial. CloudFormation

- Comparar: os dados de referência do teste de origem e os dados de reprodução de destino serão comparados e os resultados serão exibidos para você como dados idênticos, diferentes, equivalentes ou ausentes.

## Diferenças

Indica que foram detectadas diferenças entre os conjuntos de dados de referência e de reprodução pela comparação de dados. Por exemplo, um campo em uma tela 3270 on-line que mostre valores diferentes do ponto de vista da lógica de negócios entre o mainframe de origem e a aplicação

modernizada de destino será considerado uma diferença. Outro exemplo é um registro em um conjunto de dados que não seja idêntico entre as aplicações de origem e de destino.

### Equivalências

Registros equivalentes são registros que são diferentes entre os conjuntos de dados de referência e de reprodução, mas não devem ser tratados como diferentes do ponto de vista da lógica de negócios. Por exemplo, um registro contendo o timestamp de quando o conjunto de dados foi produzido (tempo de execução da workload). Usando regras de equivalência personalizáveis, é possível instruir o Application Testing a tratar essa diferença de falso positivo como uma equivalência, mesmo que ela mostre valores diferentes entre os dados de referência e de reprodução.

### Aplicação de origem

O aplicação de origem do mainframe com a qual fazer a comparação.

### Aplicação de destino

A aplicação nova ou modificada na qual o teste é feito e que será comparada à aplicação de origem para detectar quaisquer defeitos e obter equivalência funcional entre as aplicações de origem e de destino. O aplicativo de destino normalmente está sendo executado na AWS nuvem.

## Tutorial: Configurar a aplicação de exemplo CardDemo

> O AWS Application Testing está em versão de pré-visualização para o AWS Mainframe Modernization e está sujeito a alterações. Recomendamos o uso desse recurso somente com dados e aplicações de teste, e não em ambientes de produção.

Para este tutorial, você cria uma pilha do AWS CloudFormation que ajuda a configurar a aplicação de exemplo CardDemo para a redefinição da plataforma com o Micro Focus no serviço gerenciado AWS Mainframe Modernization e recursos como o Application Testing do AWS Mainframe Modernization. O tutorial descreve um modelo de exemplo do AWS CloudFormation que pode ser usado para criar a pilha. Também fornecemos um arquivo compactado com os artefatos necessários da aplicação. O modelo de exemplo fornece um banco de dados, um ambiente de runtime, uma aplicação e um ambiente de rede totalmente isolado.

Esse modelo cria vários recursos da AWS. Você receberá cobrança por eles se criar uma pilha com base nesse modelo.

## Pré-requisitos

- Baixe e descompacte o `IC3-card-demo-zip` e o `datasets_Mainframe_ebcdic.zip`. Esses arquivos contêm o CardDemo de exemplo e os conjuntos de dados de exemplo para uso com o AWS Application Testing.
- Crie um bucket do Amazon S3 para manter os arquivos do CardDemo e outros artefatos. Por exemplo, `my-carddemo-bucket`.

## Etapa 1: Preparar para configurar o CardDemo

Faça upload dos arquivos de exemplo do CardDemo e edite o modelo do AWS CloudFormation que criará a aplicação CardDemo.

1. Faça upload das pastas `IC3-card-demo` e `datasets_Mainframe_ebcdic` que você descompactou anteriormente para o bucket.
2. Baixe o modelo `aws-m2-math-mf-carddemo.yaml` do AWS CloudFormation do bucket. Ele está na pasta `IC3-card-demo`.
3. Edite o modelo `aws-m2-math-mf-carddemo.yaml` do AWS CloudFormation da seguinte forma:
   - Altere o parâmetro `BucketName` para o nome do bucket que você definiu anteriormente, como `my-carddemo-bucket`.
   - Altere o `ImportJsonPath` para o local do arquivo `mf-carddemo-datasets-import.json` no bucket. Por exemplo, `s3://my-carddemo-bucket/IC3-card-demo/mf-carddemo-datasets-import.json`. Ao atualizar esse valor, garanta que a saída `M2ImportJson` tenha o valor correto.
   - (Opcional) Adapte os parâmetros `EngineVersion` e `InstanceType` para que correspondam aos seus padrões.

> **Note**
>
> Não modifique as saídas `M2EnvironmentId` e `M2ApplicationId`. O Application Testing usa esses valores para localizar os recursos com os quais ele vai interagir.

## Etapa 2: Criar todos os recursos necessários

Execute seu modelo personalizado do AWS CloudFormation para criar todos os recursos necessários para concluir este tutorial com êxito. Esse modelo configura a aplicação CardDemo para que você possa usá-la em testes.

1. Faça login no console do AWS CloudFormation, escolha Criar pilha e, depois, escolha Com novos recursos (padrão).
2. Em Pré-requisito: preparar modelo, selecione O modelo está pronto.
3. Em Especificar modelo, escolha Fazer upload de um arquivo de modelo e selecione Escolher arquivo.
4. Navegue até onde baixou `aws-m2-math-mf-carddemo.yaml` e escolha esse arquivo. Depois, selecione Próximo.
5. Em Especificar detalhes da pilha, forneça um nome para a pilha para que você possa encontrá-la facilmente em uma lista e escolha Próximo.
6. Em Configurar opções da pilha mantenha os valores padrão e escolha Próximo.
7. Em Revisão, verifique o que o AWS CloudFormation está criando para você e escolha Enviar.

Leva cerca de 10 a 15 minutos para o AWS CloudFormation criar a pilha.

> **Note**
>
> O modelo é configurado para acrescentar um sufixo exclusivo aos nomes dos recursos que ele cria. Isso significa que você pode criar várias instâncias desse modelo de pilha em paralelo, um recurso importante para o Application Testing, que permite executar vários cenários de teste ao mesmo tempo.

## Etapa 3: Implantar e iniciar a aplicação

Implante a aplicação CardDemo que o AWS CloudFormation criou para você e verifique se ela está em execução.

1. Abra o console do AWS Mainframe Modernization e selecione Aplicações na navegação esquerda.
2. Escolha a aplicação CardDemo, que tem um nome parecido com `aws-m2-math-mf-carddemo-abc1d2e3`.
3. Escolha Ações e, depois, escolha Implantar aplicação.
4. Em Ambientes, escolha o ambiente de runtime que corresponde à aplicação. Ele terá o mesmo identificador exclusivo anexado ao final do nome. Por exemplo, `aws-m2-math-mf-carddemo-abc1d2e3`.
5. Escolha Implantar. Espere até que a aplicação seja implantada com êxito e esteja no estado Pronta.
6. Escolha a aplicação e, depois, escolha Ações e Iniciar aplicação. Espere até que a aplicação esteja no estado Em execução.
7. Na página de detalhes da aplicação, copie a porta e o nome do host DNS necessários para se conectar à aplicação em execução.

## Etapa 4: Importar os dados iniciais

Para usar a aplicação de exemplo CardDemo, você deve importar um conjunto inicial de dados. Execute as etapas a seguir.

1. Faça download do arquivo `mf-carddemo-datasets-import.json`.
2. Edite o arquivo usando o editor de texto de sua preferência.
3. Localize o parâmetro `s3Location` e atualize o valor para apontar para o bucket do Amazon S3 que você criou.
4. Faça essa mesma alteração para todas as ocorrências de `s3Location` e salve o arquivo.
5. Faça login no console do Amazon S3 e navegue até o bucket que você criou anteriormente.
6. Faça upload do arquivo `mf-carddemo-datasets-import.json` personalizado.
7. Abra o console do AWS Mainframe Modernization e selecione Aplicações na navegação esquerda.

8. Escolha a aplicação CardDemo.
9. Escolha Conjunto de dados e selecione Importar.
10. Navegue até o local no Amazon S3 em que você fez upload do arquivo JSON personalizado e escolha Enviar.

Esse trabalho importa 23 conjuntos de dados. Para monitorar o resultado do trabalho de importação, verifique o console. Quando todos os conjuntos de dados forem importados com êxito, conecte-se à aplicação.

> **Note**
>
> Quando você usa esse modelo no Application Testing, a saída `M2ImportJson` lida automaticamente com o processo de importação.

## Etapa 5: Conectar-se à aplicação CardDemo

Conecte-se à aplicação de exemplo CardDemo usando o emulador 3270 de sua escolha.

- Quando a aplicação estiver em execução, use o emulador 3270 para se conectar à aplicação, especificando o nome do host DNS e o nome da porta, se necessário.

  Por exemplo, se você estiver usando o emulador c3270 de código aberto, o comando será semelhante a:

  ```
  c3270 -port port-number DNS-hostname
  ```

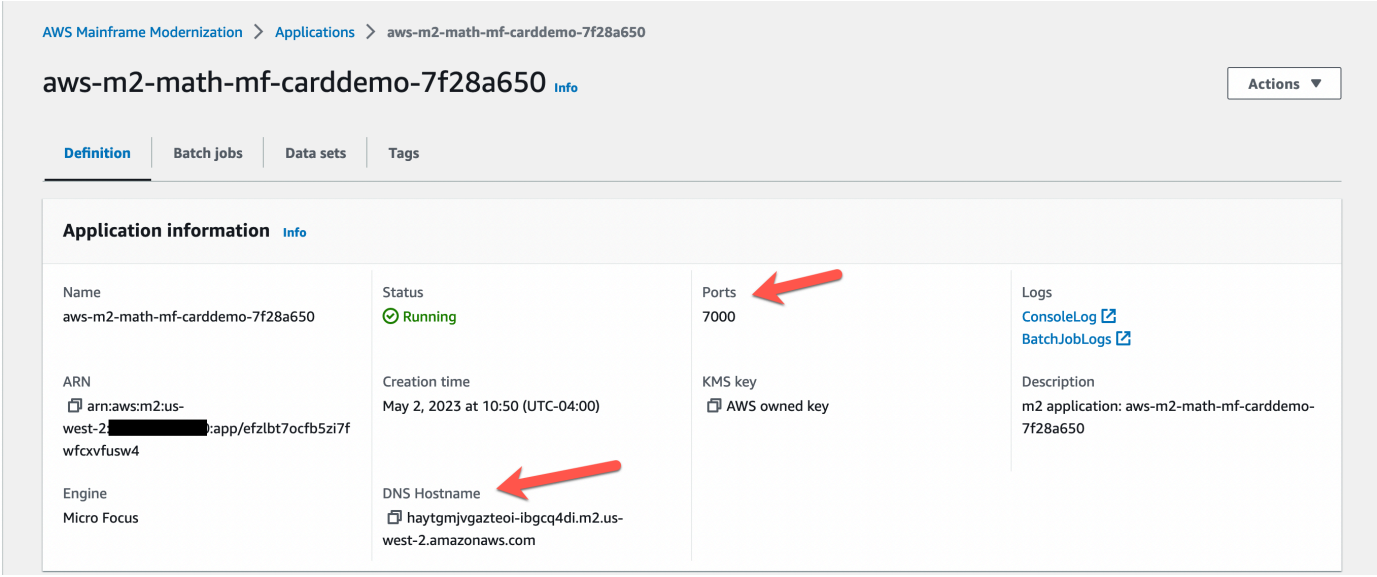
  porta

  A porta especificada na página de detalhes da aplicação. Por exemplo, 6000.

  Hostname

  O nome do host DNS especificado na página de detalhes da aplicação.

  A figura a seguir mostra onde encontrar a porta e o nome do host DSN.




# Tutorial: Teste de aplicativos de modernização de AWS mainframe, reproduza e compare o uso do AWS Blu Age CardDemo implantado no Amazon EC2

O AWS Application Testing está em versão de pré-visualização para o AWS Mainframe Moderniza tion e está sujeito a alterações. Recomendamos o uso desse recurso somente com dados e aplicações de teste, e não em ambientes de produção.

Neste tutorial, você concluirá as etapas necessárias para reproduzir e comparar cargas de trabalho de teste com o CardDemo aplicativo executado no AWS Blu Age implantado no Amazon EC2.

## Etapa 1: Obter a imagem de máquina da Amazon (AMI) do Amazon EC2 AWS Blu Age

Siga as instruções no tutorial de configuração do AWSAWSBlu Age Runtime (no Amazon EC2) para ver as etapas de integração necessárias para obter acesso AWS ao Blu Age na AMI do Amazon EC2.

## Etapa 2: iniciar uma instância do Amazon EC2 usando a AMI AWS Blu Age

1. Configure suas credenciais da AWS.

2. Identifique a localização do arquivo binário 3.5.0 do Amazon EC2 AMI (somente CLI/versão Blu AWS Age) do bucket do 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**
>
> O recurso Application Testing está disponível para uso somente em quatro regiões em produção (us-east-1, sa-east-1, eu-central-1 e ap-southeast-2).

3. Restaure a AMI na sua conta com o seguinte comando:

```
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**
>
> Substitua o nome do arquivo bin da AMI e a região onde você deseja criar a AMI.

4. Depois de criar uma instância do Amazon EC2, você poderá encontrar o ID da AMI correta que foi restaurada no bucket do Amazon S3 no catálogo de imagens do Amazon EC2.

> **Note**
>
> Neste tutorial, o ID da AMI é ami-0d0fafcc636fd1e6d, e você deve alterar esse ID nos diferentes arquivos de configuração para aquele fornecido a você.

1. Se o aws ec2 create-restore-image-task falhar, verifique sua versão do Python e da CLI usando o seguinte comando:

```
aws --version
```

> **Note**
>
> A versão do Python deve ser >= 3 e a versão da CLI deve ser >= 2.

2. Se essas versões estiverem obsoletas, a CLI deverá ser atualizada. Como atualizar a CLI:
   a. Siga as instruções em Instalar ou atualize para versão mais recente da AWS CLI.
   b. Remova a CLI v1 com o seguinte comando:

```
sudo yum remove awscli
```

   c. Instale a CLI v2 com o seguinte comando:

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
 "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

   d. Por fim, verifique a versão do Python e da CLI com o seguinte comando:

```
aws --version
```

3. Você pode então refazer o aws ec2 create-restore-image-task.

## Etapa 3: Carregar arquivos CardDemo dependentes para o S3

Copie o conteúdo das pastas, bancos de dados, sistema de arquivos e dados do usuário. Baixe e descompacte os CardDemo aplicativos. Essas três pastas devem ser copiadas em um dos seus buckets chamado your-s3-bucket nesta documentação.

## Etapa 4: Carregar bancos de dados e inicializar o aplicativo CardDemo

Crie uma instância temporária do Amazon EC2 que você usará como recurso computacional para gerar os snapshots de banco de dados necessários para o aplicativo. CardDemo Essa instância do EC2 não executará o CardDemo aplicativo em si, mas gerará os instantâneos do banco de dados que serão usados posteriormente.

Comece editando o CloudFormation modelo fornecido chamado 'load-and-create-ba-snapshots.yml.' Esse é o CloudFormation modelo usado para criar a instância do Amazon EC2 usada para gerar os instantâneos do banco de dados.

1. Gere e forneça seu par de chaves do EC2 que será usado para a instância do EC2. Para obter mais informações, consulte Criar pares de chaves.

   Exemplo:

   ```
   Ec2KeyPair:
       Description: 'ec2 key pair'
       Default: 'm2-tests-us-west-2'
       Type: String
   ```

2. Especifique o caminho do Amazon S3 da pasta em que você colocou a pasta banco de dados da etapa anterior:

   ```
   S3DBScriptsPath:
       Description: 'S3 DB scripts folder path'
       Type: String
       Default: 's3://your-s3-bucket/databases'
   ```

3. Especifique o caminho do Amazon S3 da pasta em que você colocou a pasta sistema de arquivos da etapa anterior:

   ```
   S3ApplicationFilesPath:
       Description: 'S3 application files folder path'
       Type: String
       Default: 's3://your-s3-bucket/file-system'
   ```

4. Especifique o caminho do Amazon S3 da pasta em que você colocou a pasta dados do usuário da etapa anterior:

   ```
   S3UserDataPath:
       Description: 'S3 userdata folder path'
       Type: String
       Default: 's3://your-s3-bucket/userdata'
   ```

5. Especifique também um caminho do Amazon S3 em que você salvará os arquivos de resultados a serem usados na próxima etapa.

```
S3SaveProducedFilesPath:
    Description: 'S3 path folder to save produced files'
    Type: String
    Default: 's3://your-s3-bucket/post-produced-files'
```

6. Altere o ID da AMI pelo correto, obtido anteriormente neste tutorial, usando o modelo abaixo:

```
  BaaAmiId:
    Description: 'ami id (AL2) for ba anywhere'
    Default: 'ami-0bd41245734fd20d9'
    Type: String
```

- Opcionalmente, você pode alterar o nome dos três instantâneos que serão criados pela execução dos bancos de dados de carga. withCloudFormation Eles ficarão visíveis na CloudFormation pilha à medida que ela for criada e serão usados posteriormente neste tutorial. Lembre-se de anotar os nomes usados para os snapshots do banco de dados.

```
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**
>
> Neste documento, presumimos que o nome dos instantâneos permaneça consistente.

7. Execute o CloudFormation com CLI ou AWS console usando o botão Create Stack e o assistente. Ao final do processo, você deverá ver três snapshots no console do RDS com o nome que você escolheu seguido por um ID exclusivo. Você precisará desses nomes na próxima etapa.

   > **Note**
   >
   > O RDS adicionará postfixes aos nomes dos snapshots definidos no modelo do AWS CloudFormation. Certifique-se de obter o nome do snapshot completo do RDS antes de passar para a próxima etapa.

   Exemplo de comando da 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
   ```

   Você também pode verificar no caminho do Amazon S3 que você forneceu para o S3 SaveProducedFilesPath se os conjuntos de dados foram criados corretamente.

## Etapa 5: iniciar o tempo de execução do AWS Blu Age CloudFormation

Use CloudFormation para executar a instância do Amazon EC2 com o aplicativo CardDemo AWS Blu Age. Você deve substituir algumas variáveis no CloudFormation nome `m2-with-ba-using-snapshots-https-authentication.yml` editando o arquivo YAML ou modificando os valores no console durante a inicialização do CFN.

1. Modifique o `AllowedVpcEndpointPrincipals` para especificar qual conta alcançará o VPC endpoint para acessar o tempo de execução do AWS Blu Age, usando os seguintes comandos:

   ```
   AllowedVpcEndpointPrincipals:
       Description: 'comma-separated list of IAM users, IAM roles, or AWS accounts'
       Default: 'apptest.amazonaws.com'
       Type: String
   ```

2. Altere o valor das variáveis SnapshotPrimaryDb SnapshotBlusamDb, e SnapshotJicsDb para o nome dos instantâneos. Obtenha também os nomes dos snapshots do RDS depois que foram criados na etapa anterior.

```
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**
>
> O RDS adicionará seu próprio postfix aos nomes dos snapshots.

3. Forneça seu par de chaves do Amazon EC2 para a instância do EC2 usando este comando:

```
Ec2KeyPair:
    Description: 'ec2 key pair'
    Default: 'm2-tests-us-west-2'
    Type: String
```

4. Forneça a ID da AMI que você obteve durante o processo de registro da AMI para a variável BaaAmiId, usando:

```
BaaAmiId:
    Description: 'ami id (AL2) for ba anywhere'
    Default: 'ami-0d0fafcc636fd1e6d'
    Type: String
```

5. Forneça o caminho da pasta do Amazon S3 que você usou na etapa anterior para salvar os arquivos produzidos, usando o seguinte comando:

```
S3ApplicationFilesPath:
    Description: 'bucket name'
    Type: String
    Default: 's3://your-s3-bucket/post-produced-files'
```

6. Por fim, forneça o caminho da pasta s3-: userdata-folder-path

```
S3UserDataPath:
    Description: 'S3 userdata folder path'
    Type: String
    Default: 's3://your-s3-bucket/userdata'
```

- (Opcional) É possível habilitar o modo HTTPS e a autenticação HTTP básica para o tomcat. Embora as configurações padrão também funcionem.

> **Note**
>
> Por padrão, o modo HTTPS está desativado e definido para o modo HTTP no parâmetro BacHttpsMode:

Por exemplo: .

```
BacHttpsMode:
    Description: 'http or https for Blue Age Runtime connection mode '
    Default: 'http'
    Type: String
    AllowedValues: [http, https]
```

- (Opcional) Para ativar o modo HTTPS, você deve alterar o valor para HTTPS e fornecer o ARN do certificado ACM alterando o valor da variável ACM: CertArn

```
ACMCertArn:
    Type: String
    Description: 'ACM certificate ARN'
    Default: 'your arn certificate'
```

- (Opcional) A autenticação básica é desativada por padrão com o parâmetro WithBacBasicAuthenticationdefinido como false. Você poderá habilitá-lo definindo o valor como verdadeiro.

```
WithBacBasicAuthentication:
    Description: 'false or true for Blue Age Runtime Basic Authentication '
    Default: false
    Type: String
    AllowedValues: [true, false]
```

7. Depois de concluir a configuração, você pode criar a pilha usando o CloudFormation modelo editado.

## Etapa 6: Testando a instância AWS Blu Age do Amazon EC2

Execute manualmente o CloudFormation modelo para criar a instância AWS Blu Age do Amazon EC2 para CardDemo o aplicativo, a fim de garantir que ele inicie sem erros. Isso é feito para verificar se o CloudFormation modelo e todos os pré-requisitos são válidos, antes de usar o CloudFormation modelo com o recurso de teste de aplicativos. Em seguida, você pode usar o Application Testing para criar automaticamente a instância AWS Blu Age Amazon EC2 de destino durante a reprodução e compará-la por meio de uma condição inicial.

1. Execute o comando CloudFormation create stack para criar a instância AWS Blu Age do Amazon EC2, fornecendo o modelo m2 with-ba-using-snapshots - CloudFormation -https-authentication.yml que você editou na etapa anterior:

```
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**
>
> Lembre-se de especificar a região correta em que a AMI AWS Blu Age foi restaurada.

2. Verifique se tudo está funcionando corretamente conferindo o console para encontrar a instância do Amazon EC2 em execução. Conecte-se a ela usando o Gerenciador de sessões.
3. Depois de se conectar à instância do Amazon EC2, use os seguintes comandos:

```
sudo su
cd /m2-anywhere/tomcat.gapwalk/velocity/logs
cat catalina.log
```

4. Certifique-se de que não haja exceções ou erros no log.
5. Depois, verifique se a aplicação está respondendo usando este comando:

```
curl http://localhost:8080/gapwalk-application/
```

Você verá a mensagem “A aplicação Jics está em execução”.

## Etapa 7: Validar que as etapas anteriores foram concluídas corretamente

Nas próximas etapas, usaremos o AWS Mainframe Modernization Application Testing para reproduzir e comparar conjuntos de dados criados pelo aplicativo. CardDemo Essas etapas dependem da conclusão com êxito de todas as etapas anteriores deste tutorial. Valide o seguinte antes de continuar:

1. Você criou com sucesso a instância AWS Blu Age na Amazon EC2 por meio do AWS CloudFormation modelo.
2. O serviço Tomcat no AWS Blu Age no Amazon EC2 está funcionando, sem exceções.

Ao executar a instância do EC2 com o CardDemo aplicativo, conclua as etapas a seguir no console do Application Testing para executar a repetição e a comparação de conjuntos de dados em lote.

## Etapa 8. Criar condição inicial

Nesta etapa, você cria uma condição inicial fornecendo o CloudFormation modelo que você usou para implantar o CardDemo aplicativo AWS Blu Age no Amazon EC2.

1. Abra o console do AWS Mainframe Modernization em https://console.aws.amazon.com/m2/.
2. No painel de navegação esquerdo, escolha Application Testing.
3. Em Application Testing, escolha Criar condição inicial.
4. Use sua cópia local com o caminho modificado do Amazon S3 e os IDs dos snapshots que apontam para seus recursos.

## Etapa 9: Criar o caso de teste

Nesta etapa, você cria o caso de teste que será usado para comparar os conjuntos de dados criados na aplicação Card Demo.

1. Crie um novo caso de teste. Dê a ele um nome e uma descrição.
2. Especifique `CREASTMT.JCL` como o nome do JCL.
3. Adicione os seguintes conjuntos de dados à definição do caso de teste:

| Nome | CCSID | RecordFormat | RecordLength |
|---|---|---|---|
| AWS.M2.CARDDEMO.STATEMNT.PS | “037” | FB | 80 |
| AWS.M2.CARDDEMO.STATEMNT.HTML | “037” | FB | 100 |

> **Note**
>
> O nome do JCL e os detalhes do conjunto de dados devem ser correspondentes.

## Etapa 10: Criar um cenário de teste

1. Crie um novo cenário de teste e forneça um nome e uma descrição para ele.
2. Adicione o caso de teste criado na etapa anterior ao cenário de teste.
3. Depois que o cenário de teste for criado, selecione a condição inicial criada na etapa 1 na página de visão geral do cenário de teste.

## Etapa 11: Gravar seu cenário de teste

Nessa etapa, você executará casos de teste na origem. Para fazer isso:

1. Baixe e execute os conjuntos de dados que se originaram da execução do aplicativo no mainframe. CardDemo
2. Faça upload da pasta descompactada para seu bucket do Amazon S3. Esse bucket do Amazon S3 deve estar na mesma região que os outros recursos do Application Testing.

> **Note**
>
> Deve haver dois arquivos com nomes correspondentes aos nomes dos conjuntos de dados passados no caso de teste anterior.

3. Na página Visão geral do cenário de teste, escolha o botão Gravar.
4. Na página Gravação do cenário de teste, especifique a localização do Amazon S3 para onde você fez upload dos conjuntos de dados obtidos do mainframe de origem.
5. Clique em Enviar para iniciar o processo de gravação.

> **Note**
>
> Aguarde a conclusão da gravação antes de reproduzir e comparar.

## Etapa 12: Reproduzir e comparar

Execute o cenário de teste e os casos de teste no ambiente AWS AWS Blu Age de destino no Amazon EC2. O Application Testing capturará os conjuntos de dados produzidos pela reprodução e os comparará com os conjuntos de dados de referência que foram registrados no mainframe.

1. Escolha Reproduzir e comparar. A criação da pilha, a comparação e a exclusão da CloudFormation pilha devem levar cerca de três minutos.

Quando tudo for concluído, você deverá ter resultados de comparação com algumas diferenças criadas intencionalmente para o propósito desta demonstração.

# AWS Modernização de mainframe, testes de aplicativos, conjuntos de dados suportados, páginas de código

AWS O teste de aplicativos está na versão prévia da modernização do AWS mainframe e está sujeito a alterações. Recomendamos o uso desse recurso somente com dados e aplicações de teste, e não em ambientes de produção.

Use a tabela a seguir para determinar se o identificador de conjunto de caracteres codificado (CCSID) de seus dados é compatível com testes de aplicativos. AWS Se os dados usarem um CCSID incompatível, recomendamos que você os converta em um CCSID compatível ou entre em contato conosco para obter ajuda.

| CCSID | Conjunto de caracteres | Descrição |
|---|---|---|
| 37 | IBM037, IBM-037, Cp037 | Host: EUA, Canadá (ESA), Holanda, Portugal, Brasil, Austrália, Nova Zelândia |
| 273 | IBM273, IBM-273, Cp273 | Host: Áustria, Alemanha |
| 277 | IBM277, IBM-277, Cp277 | Host: Dinamarca, Noruega |
| 278 | IBM278, IBM-278, Cp278 | Host: Finlândia, Suécia |
| 280 | IBM280, IBM-280, Cp280 | Host: Itália |
| 284 | IBM284, IBM-284, Cp284 | Host: Espanha, América Latina (espanhol) |
| 285 | IBM285, IBM-285, Cp285 | Host: Reino Unido |
| 297 | IBM297, IBM-297, Cp297 | Host: França |
| 300 | IBM-300 | DB EBCDIC JAPÃO |
| 301 | IBM-301 | Dados do PC: DB Japão |

| CCSID | Conjunto de caracteres | Descrição |
| --- | --- | --- |
| 437 | IBM437, IBM-437, US-ASCII, ASCII, Cp437, US-ASCII | Dados do PC: PC Base USA, muitos outros países |
| 500 | IBM500, IBM-500, Cp500 | Host: Bélgica, Canadá (AS/400), Suíça, Latin-1 Internacional |
| 720 | IBM-720 | MSDOS ÁRABE |
| 737 | IBM-737, x-IBM737 | MSDOS GREGO |
| 775 | IBM775, IBM-775 | MSDOS BÁLTICO |
| 808 | IBM-808 | Dados do PC: cirílico, Rússia, com euro |
| 813 | ISO-8859-7, ISO8859_7 | ISO 8859-7: Grécia |
| 819 | ISO-8859-1, ISO8859_1 | ISO 8859-1: países de Latin-1 |
| 833 | IBM-833 | EBCDIC COREANO |
| 834 | IBM-834, x-IBM834 | DB EBCDIC COREANO |
| 835 | IBM-835 | DB EBCD CHINÊS TRADICIONAL |
| 836 | IBM-836 | EBCDIC CHINÊS SIMPLIFIC ADO |
| 837 | IBM-837 | EBCDIC CHINÊS SIMPLIFIC ADO |
| 850 | IBM850, IBM-850, Cp850 | Dados de PC: países de Latin-1 |
| 855 | IBM855, IBM-855, Cp855 | Dados do PC: cirílico |
| 856 | IBM-856, x-IBM856, Cp856 | Dados do PC: hebraico |

| CCSID | Conjunto de caracteres | Descrição |
| --- | --- | --- |
| 858 | IBM00858, IBM-858, Cp858 | Dados de PC: países de Latin-1, com euro |
| 859 | IBM-859 | Dados do PC: LATIN-9 |
| 860 | IBM860, IBM-860 | Dados do PC: português |
| 861 | IBM861, IBM-861 | Dados do PC: Islândia |
| 862 | IBM862, IBM-862, Cp862 | Dados do PC: hebraico (migração) |
| 863 | IBM863, IBM-863 | Dados do PC: Canadá |
| 865 | IBM865, IBM-865, Cp865 | Dados do PC: Dinamarca/ Noruega |
| 866 | IBM866, IBM-866, Cp866 | Dados do PC: cirílico, Rússia |
| 867 | IBM-867 | Dados do PC: hebraico com euro |
| 870 | IBM870, IBM-870, Cp870 | Host: Latin-2 multilíngue |
| 871 | IBM871, IBM-871, Cp871 | Host: Islândia |
| 874 | x-IBM874 | Dados do PC: tailandês |
| 875 | IBM-875, x-IBM875, Cp875 | Host: Grécia |
| 897 | IBM-897 | Dados do PC: Japan SB |
| 912 | ISO-8859-2, ISO8859_2 | ISO 8859-2: Latin-2 multilíng ue |
| 915 | ISO-8859-5, ISO8859_5 | ISO 8859-5: cirílico |
| 916 | ISO-8859-8, ISO8859_8 | ISO 8859-8: hebraico |

| CCSID | Conjunto de caracteres | Descrição |
| --- | --- | --- |
| 918 | IBM918, IBM-918, Cp918 | Host: urdu |
| 920 | ISO-8859-9, ISO8859_9 | ISO 8859-9: Latin-5 (ECMA-128, Turquia TS-5881) |
| 921 | IBM-921, x-IBM921, Cp921 | Dados do PC: Letônia, Lituânia |
| 922 | IBM-922, x-IBM922, Cp922 | Dados do PC: Estônia |
| 923 | ISO-8859-15, Cp923, ISO8859_15_FDIS | ISO 8859-15: Latin-9 |
| 924 | IBM-924 | ISO 8859-15: Latin-9 |
| 927 | IBM-927 | Dados do PC: chinês tradicional |
| 930 | IBM-930, x-IBM930, Cp930 | Host Katakana: SBCS estendido. Host Kanji: DBCS incluindo 4.370 caracteres definidos pelo usuário |
| 932 | IBM-932 | Dados do PC: Japão Mix |
| 933 | IBM-933, x-IBM933, Cp933 | Host: SBCS estendido. Host: DBCS incluindo 1.880 caracteres definidos pelo usuário e 11.172 caracteres hangul completos |
| 935 | IBM-935, x-IBM935, Cp935 | Host: SBCS estendido. Host Kanji: DBCS incluindo 1.880 caracteres definidos pelo usuário. |

| CCSID | Conjunto de caracteres | Descrição |
| --- | --- | --- |
| 937 | IBM-937, x-IBM937, Cp937 | Host: SBCS estendido. Host Kanji: DBCS incluindo 6.204 caracteres definidos pelo usuário |
| 939 | IBM-939, x-IBM939, Cp939 | Host latino: SBCS estendido . Host Kanji: DBCS incluindo 4.370 caracteres definidos pelo usuário. |
| 942 | IBM-942, IBM-942C, x-IBM942, x-IBM942C, Cp942, Cp942C | Dados do PC: SBCS estendido. Host Kanji: DBCS incluindo 1.880 caracteres definidos pelo usuário |
| 943 | IBM-943, IBM-943C, Shift_JIS, windows-31j, windows-932, x-IBM943, x-IBM943C, Cp943, Cp943C, MS932 | Dados do PC: SBCS. Host Kanji: DBCS para ambiente aberto incluindo 1.880 caracteres definidos pelo usuário do IBM |
| 947 | IBM-947 | BIG-5 CHINÊS TRADICIONAL |
| 948 | IBM-948, x-IBM948, Cp948 | Dados do PC: SBCS estendido. Host Kanji: DBCS incluindo 6.204 caracteres definidos pelo usuário |
| 949 | IBM-949, IBM-949C, x-IBM949, x-IBM949C, Cp949, Cp949C | Código IBM KS, dados do PC: SBCS. Código IBM KS, dados do PC: DBCS incluindo 1.880 caracteres definidos pelo usuário |

| CCSID | Conjunto de caracteres | Descrição |
| --- | --- | --- |
| 950 | Big5, IBM-950, x-IBM950, Cp950 | Dados do PC: SBCS (IBM BIG5). Dados do PC: DBCS incluindo 13.493 CNS, 566 selecionados da IBM, 6.204 caracteres definidos pelo usuário |
| 951 | IBM-951 | Dados do PC: IBM KS |
| 954 | EUC-JP, IBM-954, IBM-954C | G0: JIS X201 romano. G1: JIS X208-1990. G1: JIS X201 Katakana. G1: JIS X212 |
| 964 | EUC-TW, IBM-964, x-IBM964, Cp964 | G0: ASCII. G1: CNS 11.643 plano 1. G1: CNS 11.643 plano 2. |
| 970 | EUC-KR, x-IBM970, Cp970 | G0: ASCII. G1: KSC X5601-1989 incluindo 1.880 caracteres definidos pelo usuário |
| 971 | IBM-971 | EUC COREANO |
| 1006 | IBM-1006, x-IBM1006, Cp1006 | ISO-8: Urdu |
| 1025 | IBM-1025, x-IBM1025, Cp1025 | Host: cirílico multilíngue |
| 1026 | IBM1026, IBM-1026, Cp1026 | Host: Latin-5 (Turquia) |
| 1027 | IBM-1027 | EBCD JAPÃO LATINO |
| 1041 | IBM-1041 | Dados do PC: Japão |

| CCSID | Conjunto de caracteres | Descrição |
|---|---|---|
| 1043 | IBM-1043 | Dados do PC: chinês tradicional |
| 1046 | IBM-1046, IBM-1046S, x-IBM1046 | ÁRABE: PC |
| 1047 | IBM1047, IBM-1047 | Host: Latin-1 |
| 1051 | hp-roman8 | EMULAÇÃO HP |
| 1088 | IBM-1088 | Dados do PC: KS Coreia |
| 1089 | ISO-8859-6, ISO8859_6 | ISO 8859-6: Árabe |
| 1097 | IBM-1097, x-IBM1097, Cp1097 | Host: Farsi |
| 1098 | IBM-1098, x-IBM1098, Cp1098 | Dados do PC: farsi |
| 1112 | IBM-1112, x-IBM1112, Cp1112 | Host: Letônia, Lituânia |
| 1114 | IBM-1114 | Dados do PC: SB T-CH |
| 1115 | IBM-1115 | Dados do PC: SB S-CH |
| 1122 | IBM-1122, x-IBM1122, Cp1122 | Host: Estônia |
| 1123 | IBM-1123, x-IBM1123, Cp1123 | Host: Ucrânia cirílica |
| 1124 | IBM-1124, x-IBM1124, Cp1124 | 8 bits: cirílico, Bielorrússia |

| CCSID | Conjunto de caracteres | Descrição |
| --- | --- | --- |
| 1140 | IBM01140, IBM-1140, Cp1140 | Host: EUA, Canadá (ESA), Holanda, Portugal, Brasil, Austrália, Nova Zelândia com euro |
| 1141 | IBM01141, IBM-1141, Cp1141 | Host: Áustria, Alemanha, com euro |
| 1142 | IBM01142, IBM-1142, Cp1142 | Host: Dinamarca, Noruega, com euro |
| 1143 | IBM01143, IBM-1143, Cp1143 | Host: Finlândia, Suécia, com euro |
| 1144 | IBM01144, IBM-1144, Cp1144 | Host: Itália, com euro |
| 1145 | IBM01145, IBM-1145, Cp1145 | Host: Espanha, América Latina (espanhol) |
| 1146 | IBM01146, IBM-1146, Cp1146 | Host: Reino Unido, com euro |
| 1147 | IBM01147, IBM-1147, Cp1147 | Host: França, com euro |
| 1148 | IBM01148, IBM-1148, Cp1148 | Host: Bélgica, Canadá (AS/400), Suíça, Latin-1 Internacional, com euro |
| 1149 | IBM01149, IBM-1149, Cp1149 | Host: Islândia, com euro |
| 1200 | UTF-16BE | Unicode com conjunto de caracteres 65535. Na ausência de uma marca de ordem de byte (BOM), presume-se que seja UTF-16 BE (big-endian). |
| 1202 | UTF-16LE | UTF-16 LE com IBM PUA |

| CCSID | Conjunto de caracteres | Descrição |
| --- | --- | --- |
| 1204 | UTF-16 | UTF-16 com IBM PUA |
| 1208 | UTF-8, UTF-8J, UTF8 | Unicode com conjunto de caracteres 65535. UTF-8. |
| 1232 | UTF-32BE | UTF-32 BE com IBM PUA |
| 1234 | UTF-32LE | UTF-32 LE com IBM PUA |
| 1236 | UTF-32 | UTF-32 com IBM PUA |
| 1351 | IBM-1351 | JAPÃO ABERTO |
| 1362 | IBM-1362 | MS-WIN COREANO |
| 1363 | IBM-1363, IBM-1363C, windows-949, MS949 | Dados do PC: MS Windows SBCS Coreano. Dados do PC: MS Windows DBCS Coreano incluindo 11.172 Hangul completos |
| 1364 | IBM-1364 | Host: SBCS estendido. Host: DBCS incluindo 1.880 caracteres definidos pelo usuário e 11.172 caracteres hangul completos |
| 1370 | IBM-1370 | Dados do PC: SBCS estendido, com euro. Dados do PC: DBCS incluindo 6.204 caracteres definidos pelo usuário, com euro |
| 1371 | IBM-1371 | Host: SBCS estendido, com euro. Host: DBCS incluindo 6.204 caracteres definidos pelo usuário, com euro |

| CCSID | Conjunto de caracteres | Descrição |
| --- | --- | --- |
| 1375 | Big5-HKSCS | Ext Big-5 Misto para HKSCS |
| 1380 | IBM-1380 | Dados do PC: SB S-CH |
| 1381 | IBM-1381, x-IBM1381, Cp1381 | Dados do PC: SBCS estendido (IBM GB). Dados do PC: DBCS (IBM GB) incluindo 31 selecionados da IBM, 1.880 caracteres definidos pelo usuário |
| 1382 | IBM-1382 | EUC CHINÊS SIMPLIFICADO |
| 1383 | EUC-CN, GB2312, IBM-1383, x-IBM1383, Cp1383 | G0: ASCII. G1: conjunto GB 2312-80 |
| 1385 | IBM-1385 | Dados do PC: GBK S-CH |
| 1386 | GBK, IBM-1386, windows-936, MS936 | Dados do PC: GBK Chinês Simplificado e IBM BIG-5 Chinês Tradicional. Dados do PC: GBK Chinês Simplificado |
| 1388 | IBM-1388 | Host: SBCS estendido. Host Kanji: DBCS incluindo 1.880 caracteres definidos pelo usuário |
| 1390 | IBM-1390 | Host Katakana: SBCS estendido, com euro. Host Kanji: DBCS incluindo 6.205 caracteres definidos pelo usuário |

| CCSID | Conjunto de caracteres | Descrição |
| --- | --- | --- |
| 1399 | IBM-1399 | Host Latino: SBCS estendido , com euro. Host Kanji: DBCS incluindo 4.370 caracteres definidos pelo usuário, com euro |
| 5050 | JIS0201, JIS0208, JIS0212, JIS0201, JIS0208, JIS0212 | G0: JIS X201 romano. G1: JIS X208-1990. G1: JIS X201 Katakana. G1: JIS X212 |
| 5054 | ISO-2022-JP | TCP JAPONÊS |
| 5346 | windows-1250, Cp1250 | MS Windows: Latin-2, versão 2 com euro |
| 5347 | windows-1251, Cp1251 | MS Windows: cirílico, versão 2 com euro |
| 5348 | windows-1252, Cp1252 | MS Windows: países de Latin-1, versão 2 com euro |
| 5349 | windows-1253, Cp1253 | MS Windows: Grécia, versão 2 com euro |
| 5350 | windows-1254, Cp1254 | MS Windows: Turquia, versão 2 com euro |
| 5351 | windows-1255, Cp1255 | MS Windows: Hebraico, versão 2 com euro |
| 5352 | windows-1256, windows-1 256S, Cp1256 | MS Windows: Árabe, versão 2 com euro |
| 5353 | windows-1257, Cp1257 | MS Windows: Borda do Báltico, versão 2 com euro |

| CCSID | Conjunto de caracteres | Descrição |
| --- | --- | --- |
| 5354 | windows-1258, Cp1258 | MS Windows: vietnamita, versão 2 com euro |
| 5488 | GB18030 | GB18030, dados de 1 byte<br>GB18030, dados de 2 bytes<br>GB18030, dados de 4 bytes |
| 9030 | IBM-838, Cp838 | Host: SBCS tailandês estendido. |
| 9066 | IBM-874, Cp874 | Dados do PC: SBCS tailandês estendido |
| 9400 | CESU-8 | CESU-8 com IBM PUA |
| 25546 | ISO-2022-KR | TCP COREANO |
| 33722 | IBM-33722, IBM-33722C | IBMeucJP |

# Transferência de arquivos na modernização AWS do mainframe

O AWS Mainframe Modernization File Transfer permite transferir e converter conjuntos de dados de mainframe para o Amazon S3 para casos de uso de modernização, migração e aumento de mainframe.

Tópicos



## O que é o Transferência de Arquivos do AWS Mainframe Modernization?

Com o AWS Mainframe Modernization File Transfer, você pode transferir e converter conjuntos de dados e arquivos com um serviço totalmente gerenciado para acelerar e simplificar os casos de uso de modernização, migração e aumento para o serviço de AWS modernização de mainframe e o Amazon S3.

Tópicos



### Benefícios do Transferência de Arquivos do AWS Mainframe Modernization

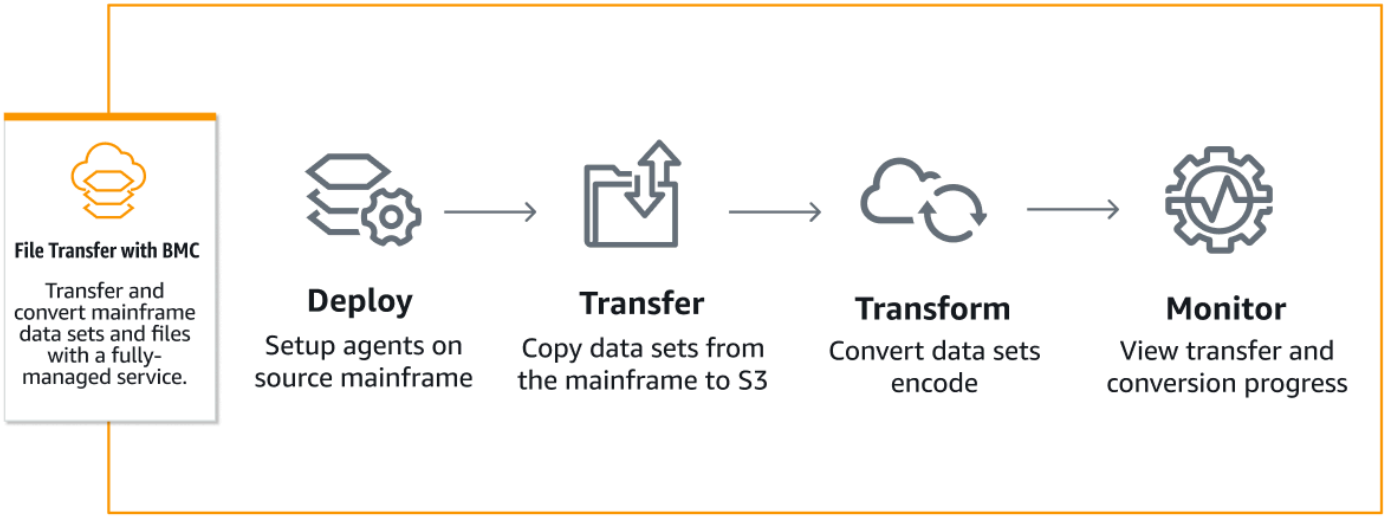
O Transferência de Arquivos do AWS Mainframe Modernization ajuda você a transferir conjuntos de dados do mainframe para o Amazon S3. Alguns benefícios incluem:

- Descoberta de conjuntos de dados e artefatos do mainframe de origem
- Transferências automatizadas e conversão de conjuntos de dados

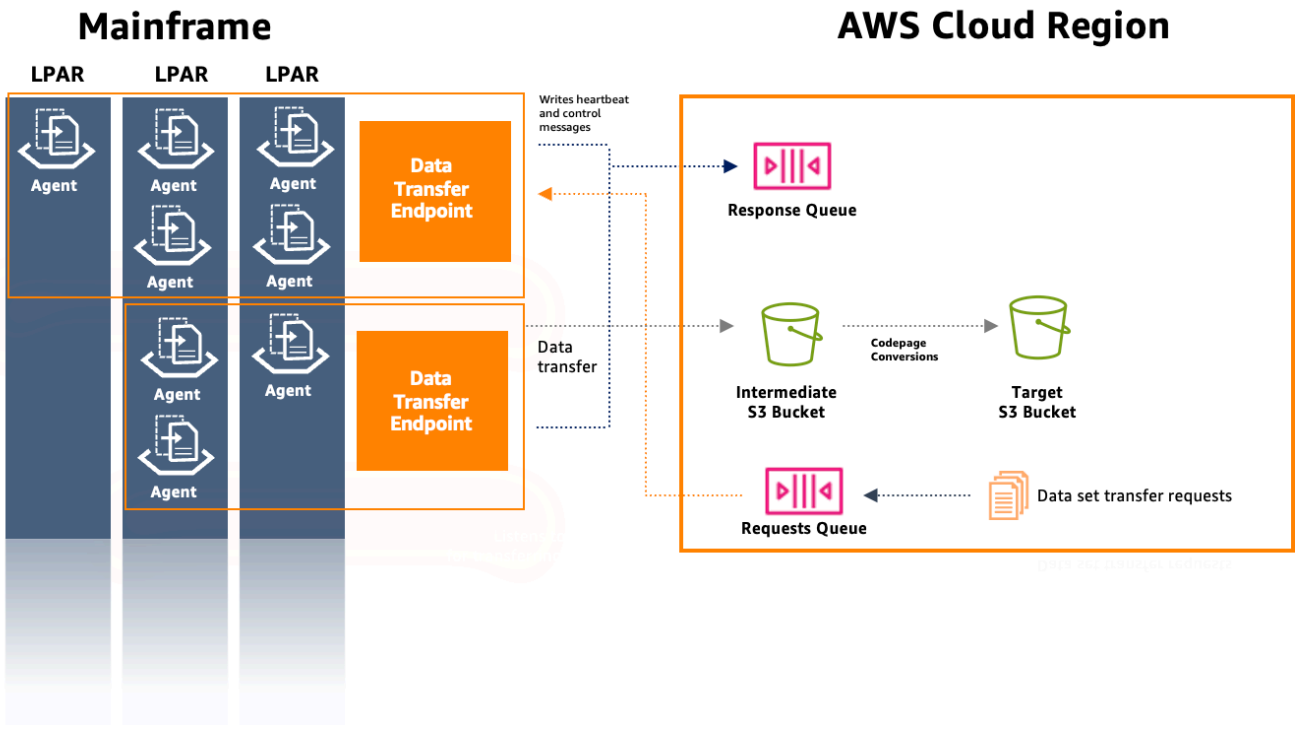
- Escalabilidade, eficiência e velocidade para obter transferências mais rápidas de conjuntos de dados para a AWS

## Como funciona o Transferência de Arquivos do AWS Mainframe Modernization

A figura a seguir é uma visão geral de como funciona o Transferência de Arquivos do AWS Mainframe Modernization em um nível conceitual.




A figura a seguir é uma visão geral da arquitetura do recurso Transferência de Arquivos do AWS Mainframe Modernization.




# Instalar um agente do Transferência de Arquivos

Siga este guia step-by-step para preencher os pré-requisitos para instalar um agente no mainframe de origem e configurar o agente.

Tópicos

- Etapa 11: Copiar o pacote tar de modernização do AWS mainframe para o diretório de trabalho no z/OS
- Etapa 12: Assumir o usuário raiz
- Configurar permissões e STC
- Criar um usuário do IAM com credenciais de acesso de longo prazo
- Criar um perfil do IAM para que o agente o assuma
- Configuração do agente

## Etapa 1: Fazer login no ISPF

Faça login na sua sessão do ISPF (Interactive System Productivity Facility). Isso geralmente é feito por meio de um emulador de terminal 3270.

## Etapa 2: Alocar um conjunto de dados para o z/FS

Usando o utilitário de conjunto de dados do ISPF, aloque um novo conjunto de dados para o z/FS. Normalmente, isso é feito no utilitário de lista de conjuntos de dados na etapa 1.

1. Acesse a opção 3.4 (Utilitários -> Conjunto de dados).
2. Pressione a tecla 'C' para criar um conjunto de dados.
3. Insira o nome do conjunto de dados (por exemplo, 'yourhlq.M2AGENT.ZFS').
4. Especifique o tipo de conjunto de dados como “Formato grande” com um tamanho primário de 1.000 cilindros e um tamanho secundário de 200.
5. Defina a organização do conjunto de dados (DSORG) como PS e o formato de registro (RECFM) como 'U' (Indefinido).
6. Conclua o processo de criação.

## Etapa 3: Formatar o conjunto de dados como z/FS

Depois de criar o conjunto de dados, formate-o como um sistema de arquivos z/FS.

Uma maneira de fazer isso é usando a seguinte Job Control Language (JCL):

```
//FORMAT  EXEC PGM=IOEAGFMT,PARM='AGGRNAME(yourhlq.M2AGENT.ZFS),FORMAT,AGGRSIZE(1200)'
//SYSPRINT DD  SYSOUT=A
```

Envie esse trabalho e verifique se ele foi concluído com êxito.

## Etapa 4: Definir o sistema de arquivos como z/OS

Defina o z/FS para o z/OS usando o comando `DEF FILESYSTEM` ou por meio de um trabalho em lotes. Use o seguinte comando:

```
DEF FILESYSTEM('yourhlq.M2AGENT.ZFS') TYPE(ZFS) MODE(R/W) MOUNTPOINT('/usr/lpp/aws/m2-agent')
```

> **Note**
>
> Essa etapa requer autoridade em nível de sistema.

## Etapa 5: Montar o sistema de arquivos

Para montar o sistema de arquivos, use o comando `MOUNT`. É possível montar o sistema de arquivos na linha de comando no ISPF ou em lote.

Por exemplo: .

```
MOUNT FILESYSTEM('yourhlq.M2AGENT.ZFS') TYPE(ZFS) MODE(RDWR) MOUNTPOINT('/usr/lpp/aws/m2-agent')
```

## Etapa 6: Verificar a montagem

Verifique se o sistema de arquivos foi montado corretamente usando o comando `D OMVS,A` ou verificando no Unix System Service (USS).

## Etapa 7: Inserir OMVs

Use o seguinte comando para inserir OMVs:

```
TSO OMVS
```

## Etapa 8: Definir a variável de ambiente do diretório de instalação do agente

Use o seguinte comando para definir o ambiente do diretório de instalação do agente:

```
export AGENT_DIR=/usr/lpp/aws/m2-agent
```

## Etapa 9: Definir a variável de ambiente do diretório de instalação do agente

Use o seguinte comando para definir a variável de ambiente do diretório de trabalho:

```
export WORK_DIR=$AGENT_DIR/tmp
```

## Etapa 10: Criar um diretório de trabalho

Use o seguinte comando para definir a variável de ambiente do diretório de trabalho:

```
mkdir -p $WORK_DIR
```

## Etapa 11: Copiar o pacote tar de modernização do AWS mainframe para o diretório de trabalho no z/OS

Ao usar FTP ou qualquer método de transferência, certifique-se de que o arquivo tar seja transferido no modo binário.

## Etapa 12: Assumir o usuário raiz

Use o seguinte comando para assumir o usuário raiz:

```
su
```

Siga estas etapas para concluir a instalação do agente:

> **Note**
>
> Você deve assumir o usuário raiz antes de prosseguir.

1. Defina a variável de ambiente da versão m2-agent para a versão que está sendo instalada no momento usando o seguinte comando:

   ```
   export M2_AGENT_VERSION=1.0.0
   ```

2. Crie o pacote tar do agente usando o seguinte comando:

```
tar -xpf m2-agent-package-$M2_AGENT_VERSION.tar -C $AGENT_DIR
```

3. Crie um link `current-version` simbólico para o diretório atual de instalação do agente com o seguinte comando:

```
ln -s $AGENT_DIR/m2-agent-v$M2_AGENT_VERSION $AGENT_DIR/current-version
```

4. Atualize e envie `CPY#PDS` para criar os conjuntos de dados do agente do Transferência de Arquivos.

> **Note**
>
> A JCL usa o SYS2.AWS.M2 HLQ.

Para criar o agente do Transferência de Arquivos, defina as linhas de parâmetros 000006 a 000012. Além disso, atualize as três variáveis simbólicas `HLQ`, `VOLSER` e `AGNTPATH` para serem usadas posteriormente na JCL:

```
oedit $AGENT_DIR/current-version/installation/CPY#PDS
submit $AGENT_DIR/current-version/installation/CPY#PDS
```

> **Note**
>
> Essa JCL é personalizada para configurar certos aspectos da instalação do agente no mainframe. Ela aloca os conjuntos de dados necessários e depois copia arquivos específicos do sistema de arquivos Unix nesses conjuntos de dados.

## Configurar permissões e STC

1. Atualize e envie uma dos `SYS2.AWS.M2.SAMPLIB(SEC#RACF)` (para configurar as permissões do RACF) ou `SYS2.AWS.M2.SAMPLIB(SEC#TSS)` (para configurar as permissões do TSS) de acordo com suas instruções. Esses membros foram criados pela etapa anterior de `CPY#PDS`.

2. Atualize a exportação de PWD no STC JCL `SYS2.AWS.M2.SAMPLIB(M2AGENT)`, se o caminho padrão do diretório do agente do Transferência de Arquivos (`/usr/lpp/aws/m2-agent`) tiver sido alterado.
3. Atualize e copie o JCL `SYS2.AWS.M2.SAMPLIB(M2AGENT)` no `SYS1.PROCLIB`.
4. Adicione `SYS2.AWS.M2.LOADLIB` à lista de APF usando o seguinte comando:

```
SETPROG APF ADD DSNAME(SYS2.AWS.M2.LOADLIB) SMS
```

5. Defina o grupo e o proprietário das pastas `logs` e `diag` do agente como o usuário/grupo do agente (M2USER/M2GROUP). Use o seguinte comando:

```
chown -R M2USER:M2GROUP $AGENT_DIR/current-version/logs
chown -R M2USER:M2GROUP $AGENT_DIR/current-version/diag
```

## Criar um usuário do IAM com credenciais de acesso de longo prazo

É necessário criar o agente de mainframe exigido por um usuário do IAM para usar filas de resposta e solicitação e salvar conjuntos de dados nos buckets do Amazon S3.

Ao criar esse usuário:

1. Em Conceder permissões, escolha Anexar políticas existentes diretamente.
2. Depois que o usuário for criado, abra a guia Credenciais de segurança e crie uma chave de acesso. Para obter mais informações sobre a criação de chaves de acesso do IAM, consulte Gerenciar chaves de acesso para usuários do IAM.
3. Na seção Chave de acesso, selecione Outro quando solicitado para Caso de uso.

> **Note**
>
> Salve a chave de acesso e a chave de acesso secreta exibidas na última página do assistente de criação da chave de acesso, antes de escolher Concluído. Essas chaves são usadas para configurar o agente de mainframe.

> **Note**
>
> Salve o ARN do usuário do IAM usado para configurar um relacionamento de confiança com um perfil do IAM.

## Criar um perfil do IAM para que o agente o assuma

Você cria um perfil do IAM com uma política de confiança personalizada para o tipo de entidade confiável. A política usará o seguinte modelo:

```
{
    "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>"
        }
    ]
```

```
}
```

Em que:

- `request-queue-arn` e `response-queue-arn` são o ARN da fila de solicitação do Amazon SQS criada durante a inicialização do endpoint de transferência de dados.
- `transfer-bucket-arn` é o ARN do bucket de transferência criado anteriormente.

> **Note**
>
> É possível pesquisar todos esses valores usando o console da AWS.

> **Note**
>
> Salve o nome da função, que será usado posteriormente para configurar o agente de mainframe.

## Configuração do agente

Como configurar o agente do Transferência de Arquivos:

1. Acesse `$AGENT_DIR/current-version/config`.
2. Edite o arquivo de configuração do agente `appication.properties` para adicionar uma configuração de ambientes usando o seguinte comando:

   ```
   oedit $AGENT_DIR/current-version/config/application.properties
   ```

   Por exemplo: .

   ```
   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>
   ```

Em que:

- `AWS_ACCOUNT_ID` é o ID da conta do cliente.
- `AWS_IAM_ROLE_NAME` é o nome do perfil do IAM criado em the section called “Criar um perfil do IAM para que o agente o assuma”.
- `AWS_IAM_ROLE_ACCESS_KEY` é a chave de acesso do usuário do IAM criado em the section called “Criar um usuário do IAM com credenciais de acesso de longo prazo”.
- `AWS_IAM_ROLE_SECRET_KEY` é a chave secreta de acesso para o usuário do IAM criado em the section called “Criar um usuário do IAM com credenciais de acesso de longo prazo”.
- `AWS_S3_BUCKET_NAME` é o nome do bucket de transferência criado com o endpoint de transferência de dados.
- `AWS_REGION` é a região na qual você configura o agente do Transferência de Arquivos.

> **Note**
>
> Pode haver várias seções desse tipo, desde que o índice entre colchetes (`[0]`) seja incrementado para cada uma.

Reinicie o agente para que as alterações entrem em vigor.

Requisitos

1. Quando um parâmetro é adicionado ou removido, o agente deve ser interrompido e iniciado. Inicie o agente do Transferência de Arquivos usando o seguinte comando na CLI:

   ```
   /S M2AGENT
   ```

   Para interromper o agente M2, use o seguinte comando na CLI:

   ```
   /P M2AGENT
   ```

2. Você pode fazer com que o agente de transferência de arquivos seja transferido para várias regiões e contas AWS definindo vários ambientes.

   ```
   #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
```

# Endpoints de transferência de dados

Os endpoints de transferência de dados permitem alta disponibilidade, escalabilidade e gerenciamento simplificado de agentes no mainframe de origem. Agentes individuais são instalados em LPARs do mainframe e podem ser agrupados em um endpoint de transferência de dados. Quando uma solicitação é feita para transferir um conjunto de dados, um agente no endpoint de transferência de dados cuidará da transferência. Para iniciar as transferências de dados, pelo menos um agente no endpoint de transferência de dados deve estar on-line.

Esse procedimento pressupõe que você tenha concluído as etapas em Configurando a modernização AWS do mainframe e configurado o agente do Transferência de Arquivos no mainframe de origem.

## Criar um endpoint de transferência de dados

Para criar endpoints de transferência de dados para transferência de arquivos, você deve seguir estas etapas no console de modernização do AWS mainframe.

Como criar um endpoint de transferência de dados

1. Abra o console de modernização do AWS mainframe em https://console.aws.amazon.com/m2/.
2. No Região da AWS seletor, escolha a região para a qual você deseja transferir arquivos do seu mainframe para um bucket do Amazon S3.

3. Na página Endpoints de transferência de dados, em Transferência de arquivos, escolha Criar endpoint de transferência de dados.
4. Na página Pré-requisitos do endpoint de transferência de dados, leia todas as instruções para garantir que você tenha concluído essas etapas. Depois de confirmado, escolha Próximo.
5. Na página Configurar endpoint de transferência de dados, adicione as informações básicas sobre seu endpoint de transferência de dados.

    1. Na seção de informações básicas, insira o nome, a descrição e a chave do KMS do endpoint de transferência de dados. Para obter mais informações sobre chaves do KMS, consulte Creating keys.

        > **Note**
        >
        > O nome do endpoint de transferência de dados deve corresponder ao nome definido ao configurar o agente do Transferência de Arquivos no mainframe de origem.

        > **Note**
        >
        > Você deve adicionar a seguinte política baseada em recursos para o KMS para que o serviço de modernização do AWS mainframe possa ler e usar essas chaves para criptografia/descriptografia:

        ```
        {
          "Sid" : "Enable AWS M2 Permissions",
          "Effect" : "Allow",
          "Principal" : {
              "Service" : "m2.amazonaws.com"
                    },
           "Action" : [
               "kms:Encrypt",
               "kms:Decrypt"
                      ],
           "Resource" : "*"
        }
        ```

    2. Especifique o local do S3 para dados intermediários, que é o local intermediário do S3 em que os conjuntos de dados transferidos do mainframe serão armazenados.

> **Note**
>
> É recomendável criar um novo bucket do Amazon S3 para suas tarefas de transferência. Para obter mais informações, consulte Criação de um bucket. Também é possível pesquisar seus buckets do Amazon S3 existentes escolhendo a opção Procurar no S3.

3. Depois de inserir os campos obrigatórios, escolha Próximo.

6. Na página Revisar e criar endpoint de transferência de dados, verifique se você preencheu os pré-requisitos e revise as informações básicas. Depois de confirmado, escolha Criar e conectar.
7. Na página Verificar conectividade, você verá todos os agentes exibidos com seus IDs e heartbeats quando a conectividade do agente for estabelecida. Depois que a conectividade for estabelecida, você receberá a mensagem “O endpoint de transferência de dados foi criado com êxito”.

> **Note**
>
> Se a mensagem “Falha ao estabelecer a conectividade do agente” for exibida, revise a etapa 4 para verificar se você concluiu todos os pré-requisitos necessários.

8. Escolha Terminar.

Você será redirecionado para a página Visão geral dos endpoints de transferência de dados, onde poderá ver a lista de todos os endpoints de transferência de dados. Também será possível ver os endpoints de transferência de dados que estão disponíveis ou que tiveram falha.

Também será possível pesquisar endpoints de transferência de dados por nome e acessar informações adicionais para cada agente disponível.

## Tarefas de transferência

As tarefas de transferência são usadas para definir a codificação de origem e a codificação de destino dos conjuntos de dados. A codificação de origem é o formato dos conjuntos de dados de origem, e a codificação de destino é o formato em que esses conjuntos de dados serão armazenados no bucket de destino do Amazon S3. Esses buckets de destino são definidos pelas tarefas de transferência.

Esta procedimento pressupõe que você tenha concluído as etapas em Configurando a modernização AWS do mainframe e configurado the section called “Endpoints de transferência de dados”.

Tópicos



## Criar tarefas de transferência

Para criar tarefas de transferência para transferência de arquivos, você deve seguir estas etapas no console de modernização do AWS mainframe.

Criar uma tarefa de transferência

> **Note**
>
> Você deve ter pelo menos um endpoint de transferência de dados para criar tarefas de transferência.

1. Abra o console de modernização do AWS mainframe em https://console.aws.amazon.com/m2/.
2. No Região da AWS seletor, escolha a região para a qual você deseja transferir arquivos do seu mainframe para um bucket do Amazon S3.
3. Na página Transferir tarefas, em Transferência de arquivos, selecione um ponto final de transferência de dados para criar tarefas de transferência.
4. Se o seu endpoint de transferência de dados não tiver nenhuma tarefa de transferência, você poderá criar tarefas escolhendo Criar tarefa de transferência.
5. Na página Criar tarefa de transferência, configure as propriedades da tarefa de transferência.

    - Nessa página, insira as informações básicas da sua tarefa de transferência, incluindo nome da tarefa de transferência, descrição, chave secreta e critérios de pesquisa de conjuntos de dados.

    > **Note**
    >
    > - Criptografe o segredo usando a chave KMS definida com o endpoint de transferência de dados. O segredo também deve conter as credenciais do

mainframe necessárias para acessar o conjunto de dados no mainframe usando as chaves `userId` e `password`. Para obter mais informações, consulte Segredo secreto do AWS Secrets Manager.

- Você deve configurar a chave secreta com a seguinte política baseada em recursos para que o serviço de modernização do AWS mainframe possa acessá-la para realizar a tarefa de transferência de dados:

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

> **Note**
>
> O tamanho máximo atual do conjunto de dados suportado é 90 GiB.

- Insira ou navegue até o local de destino do bucket do Amazon S3 de seus arquivos.
- O endpoint padrão de transferência de dados estará selecionado. Também é possível optar por alterar o endpoint a partir dos endpoints disponíveis.

6. Escolha Próximo.
7. Na página Selecionar conjuntos de dados, você deve atualizar manualmente a codificação de origem e a codificação de destino para cada um dos conjuntos de dados escolhidos. A codificação de origem é o formato do conjunto de dados de origem e a codificação de destino é o formato do conjunto de dados de destino e é usada para converter os conjuntos de dados.
8. Depois de confirmar a codificação de origem e de destino, escolha Próximo.
9. Na página Revisar e criar, é possível revisar ou editar informações da tarefa de transferência.
10. Escolha Criar tarefa de transferência.

Você verá a mensagem “Tarefa de transferência criada com êxito”.

## Visualizar tarefas de transferência

Para visualizar as tarefas de transferência para transferência de arquivos, você deve seguir estas etapas no console de modernização do AWS mainframe.

Como visualizar tarefas de transferência

1. Abra o console de modernização do AWS mainframe em https://console.aws.amazon.com/m2/.
2. No Região da AWS seletor, escolha a região para a qual você deseja transferir arquivos do seu mainframe para um bucket do Amazon S3.
3. Na página Transferir tarefas, em Transferência de arquivos, selecione o endpoint de transferência de dados para visualizar suas tarefas de transferência.
4. Para endpoints que tenham tarefas de transferência preexistentes, elas serão preenchidas na seção Tarefas de transferência. Você pode optar por visualizar os detalhes de qualquer tarefa de transferência nessa lista.

# Tutorial: Conceitos básicos do Transferência de Arquivos do AWS Mainframe Modernization

O AWS Mainframe Modernization File Transfer permite transferir e converter conjuntos de dados de mainframe para casos de uso de modernização, migração e aumento de mainframe.

Siga as etapas deste tutorial para entender como o Transferência de Arquivos do AWS Mainframe Modernization funciona.

## Visão geral

O Transferência de Arquivos consiste no seguinte:

1. Um agente a ser instalado no mainframe de origem.
2. Acesso aos recursos de descoberta, transferência e conversão de conjuntos de dados diretamente do console do serviço de gerenciamento de modernização do AWS mainframe.

Como usuário, você pode transferir conjuntos de dados do mainframe para o bucket do Amazon S3.

Tópicos



## Etapa 1: Transferir o pacote tar dos binários do agente AWS para a partição lógica do mainframe

Baixe os arquivos tar do link tar do M2-Agent.

## Etapa 2: Configurar o agente do Transferência de Arquivos no mainframe de origem

Nesta etapa, você configura e inicia o agente do Transferência de Arquivos do AWS Mainframe Modernization no mainframe de origem. O agente deve facilitar a comunicação entre o recurso do serviço Transferência de Arquivos e o mainframe de origem. É necessário pelo menos um agente por mainframe. Mais de um agente pode ser iniciado para obter alta disponibilidade e escalabilidade aprimorada.

Siga as instruções no guia the section called “Instalar um agente do Transferência de Arquivos” para concluir a instalação do agente do Transferência de Arquivos no mainframe.

## Etapa 3: Criar um endpoint de transferência de dados

Siga as etapas na página the section called “Endpoints de transferência de dados” para criar um endpoint de transferência de dados.

## Etapa 4: Criar uma tarefa de transferência

Siga as etapas na página the section called “Tarefas de transferência” para criar e gerenciar suas tarefas de transferência.

# Etapa 5: Visualizar o progresso da tarefa de transferência

Você pode ver o progresso da sua tarefa de transferência no console de modernização do AWS mainframe. Para obter mais detalhes, consulte a seção the section called “Visualizar tarefas de transferência”.

# Segurança no AWS Mainframe Modernization

A segurança para com a nuvem na AWS é a nossa maior prioridade. Como cliente da AWS, você se contará com um datacenter e uma arquitetura de rede criados para atender aos requisitos das organizações com as maiores exigências de segurança.

A segurança é uma responsabilidade compartilhada entre a AWS e você. O modelo de responsabilidade compartilhada descreve isso como a segurança da nuvem e a segurança na nuvem:

- Segurança da nuvem: a AWS é responsável pela proteção da infraestrutura que executa produtos da AWS na Nuvem AWS. A AWS também fornece serviços que podem ser usados com segurança. Auditores de terceiros testam e verificam regularmente a eficácia da nossa segurança como parte dos Programas de conformidade da AWS. Para saber mais sobre os programas de conformidade que se aplicam ao AWS Mainframe Modernization, consulte Serviços da AWS no escopo por programa de conformidade.
- Segurança na nuvem: sua responsabilidade é determinada pelo serviço da AWS que você usa. Você também é responsável por outros fatores, incluindo a confidencialidade dos dados, os requisitos da empresa e as leis e os regulamentos aplicáveis

Esta documentação ajuda você a entender como aplicar o modelo de responsabilidade compartilhada ao usar o AWS Mainframe Modernization. Ele mostra como configurar o AWS Mainframe Modernization para atender aos seus objetivos de segurança e conformidade. Você também aprenderá a usar outros serviços do AWS que o ajudam a monitorar e proteger seus recursos do AWS Mainframe Modernization.

AWS Mainframe Modernization fornece seus próprios recursos protegidos pelo IAM (aplicativo, ambiente, implantação etc.), que são os recursos administrativos do AWS Mainframe Modernization, nos quais qualquer ação deve ser permitida pelas políticas do IAM.

AWS Mainframe Modernization para reformulação de plataformas também é garantida pelo IAM. O IAM concede ou nega permissão a um diretor para uma ação específica em um recurso definido, derivado do ambiente original do mainframe, também por meio de políticas padrão do IAM. O tempo de execução de replataforma do AWS Mainframe Modernization chama o serviço de autorização do IAM quando um aplicativo tenta realizar essa ação em um recurso protegido. O IAM retornará permitir ou negar com base nos mecanismos padrão de avaliação de políticas do IAM.

Índice



# Proteção de dados na modernização AWS do mainframe

O modelo de responsabilidade AWS compartilhada O modelo se aplica à proteção de dados na modernização do AWS mainframe. Conforme descrito neste modelo, AWS é responsável por proteger a infraestrutura global que executa todos os Nuvem AWS. Você é responsável por manter o controle sobre seu conteúdo hospedado nessa infraestrutura. Você também é responsável pelas tarefas de configuração e gerenciamento de segurança dos Serviços da AWS que usa. Para obter mais informações sobre a privacidade de dados, consulte as Perguntas frequentes sobre privacidade de dados. Para mais informações sobre a proteção de dados na Europa, consulte o artigo AWS Shared Responsibility Model and GDPR no Blog de segurança da AWS .

Para fins de proteção de dados, recomendamos que você proteja Conta da AWS as credenciais e configure usuários individuais com AWS IAM Identity Center ou AWS Identity and Access Management (IAM). Dessa maneira, cada usuário receberá apenas as permissões necessárias para cumprir suas obrigações de trabalho. Recomendamos também que você proteja seus dados das seguintes formas:

- Use uma autenticação multifator (MFA) com cada conta.
- Use SSL/TLS para se comunicar com os recursos. AWS Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Configure a API e o registro de atividades do usuário com AWS CloudTrail.
- Use soluções de AWS criptografia, juntamente com todos os controles de segurança padrão Serviços da AWS.
- Use serviços gerenciados de segurança avançada, como o Amazon Macie, que ajuda a localizar e proteger dados sigilosos armazenados no Amazon S3.
- Se você precisar de módulos criptográficos validados pelo FIPS 140-2 ao acessar AWS por meio de uma interface de linha de comando ou de uma API, use um endpoint FIPS. Para ter mais

informações sobre endpoints do FIPS, consulte Federal Information Processing Standard (FIPS) 140-2.

É altamente recomendável que nunca sejam colocadas informações de identificação confidenciais, como endereços de e-mail dos seus clientes, em marcações ou campos de formato livre, como um campo Nome. Isso inclui quando você trabalha com a modernização de AWS mainframe ou outros Serviços da AWS usando o console, a API ou AWS os AWS CLI SDKs. Quaisquer dados inseridos em tags ou campos de texto de formato livre usados para nomes podem ser usados para logs de faturamento ou de diagnóstico. Se você fornecer um URL para um servidor externo, recomendemos fortemente que não sejam incluídas informações de credenciais no URL para validar a solicitação a esse servidor.

## Dados que a modernização AWS do mainframe coleta

AWS A modernização do mainframe coleta vários tipos de dados de você:

- `Application configuration`: é um arquivo JSON que você cria para configurar seu aplicativo. Ele contém suas opções para as diferentes opções que a modernização do AWS mainframe oferece. O arquivo também contém informações sobre AWS recursos dependentes, como os caminhos do Amazon Simple Storage Service, nos quais os artefatos do aplicativo são armazenados, ou o Amazon Resource Name (ARN) onde as credenciais do banco AWS Secrets Manager de dados são armazenadas.
- `Application executable (binary)`: esse é um binário que você compila e pretende implantar na modernização do AWS mainframe.
- `Application JCL or scripts`: esse código-fonte gerencia trabalhos em lotes ou outros processamentos em nome do seu aplicativo.
- `User application data`: quando você importa conjuntos de dados, a modernização do AWS mainframe os armazena no banco de dados relacional para que seu aplicativo possa acessá-los.
- `Application source code`: Por meio do Amazon AppStream 2.0, a modernização do AWS mainframe fornece um ambiente de desenvolvimento para você escrever e compilar código.

AWS A modernização do mainframe armazena esses dados nativamente em. AWS Os dados que coletamos de você são armazenados em um AWS Mainframe Modernization-managed Amazon S3 bucket. Quando você implanta um aplicativo, a modernização do AWS mainframe baixa os dados em uma instância do Amazon Elastic Compute Cloud baseada na Amazon Elastic Block Store. Quando

a limpeza é acionada, os dados são removidos do volume do Amazon EBS e do Amazon S3. Os volumes do Amazon EBS são de inquilino único, o que significa que uma instância é usada para um cliente. As instâncias nunca são compartilhadas. Quando você exclui um ambiente de execução, o volume do Amazon EBS também é excluído. Quando você exclui uma aplicação, os artefatos e a configuração são excluídos do Amazon S3.

Os registros do aplicativo são armazenados na Amazon CloudWatch. As mensagens de registro do aplicativo do cliente também são CloudWatch exportadas para. Os CloudWatch registros podem conter dados confidenciais do cliente, como dados comerciais ou informações de segurança em mensagens de depuração). Para ter mais informações, consulte Monitorando a modernização AWS do mainframe com a Amazon CloudWatch.

Além disso, se você optar por anexar um ou mais sistemas de arquivos Amazon Elastic File System ou Amazon FSx ao seu ambiente de execução, os dados dentro desses sistemas serão armazenados em AWS. Você precisará limpar esses dados se decidir parar de usar os sistemas de arquivos.

Você pode usar todas as opções de criptografia do Amazon S3 disponíveis para proteger seus dados ao colocá-los no bucket do Amazon S3 AWS que o Mainframe Modernization usa para implantação de aplicativos e importações de conjuntos de dados. Além disso, você pode usar as opções de criptografia do Amazon EFS e do Amazon FSx se anexar um ou mais desses sistemas de arquivos ao seu ambiente de execução.

## Criptografia de dados em repouso para o serviço de modernização AWS de mainframe

AWS A modernização do mainframe se integra AWS Key Management Service para fornecer criptografia transparente do lado do servidor (SSE) em todos os recursos dependentes que armazenam dados permanentemente, ou seja, Amazon Simple Storage Service, Amazon DynamoDB e Amazon Elastic Block Store. AWS A modernização do mainframe cria e gerencia AWS KMS chaves de criptografia simétricas para você em. AWS KMS

A criptografia de dados em repouso por padrão ajuda a reduzir a sobrecarga operacional e a complexidade envolvidas na proteção de dados confidenciais. Ao mesmo tempo, ele permite que você migre aplicativos que exigem rigorosa conformidade com criptografia e requisitos regulatórios.

Você não pode desativar essa camada de criptografia nem selecionar um tipo de criptografia alternativo ao criar ambientes e aplicativos de tempo de execução.

Você pode usar sua própria chave gerenciada pelo cliente para aplicativos de modernização de AWS mainframe e ambientes de execução para criptografar recursos do Amazon S3 e do Amazon EBS.

Para seus aplicativos de modernização de AWS mainframe, você pode usar essa chave para criptografar a definição do seu aplicativo, bem como outros recursos do aplicativo, como arquivos JCL, que são salvos no bucket do Amazon S3 criado na conta do serviço. Para ter mais informações, consulte Cria uma aplicação .

Para seus ambientes de execução de modernização de AWS mainframe, a modernização de AWS mainframe usa sua chave gerenciada pelo cliente para criptografar o volume do Amazon EBS que ele cria e anexa à sua instância AWS Amazon EC2 de modernização de mainframe, que também está na conta do serviço. Para ter mais informações, consulte Criar um ambiente de tempo de execução.

> **Note**
>
> Os recursos do DynamoDB são sempre criptografados usando Chave gerenciada pela AWS uma conta de serviço na modernização AWS do mainframe. Não é possível criptografar recursos do DynamoDB usando uma chave gerenciada pelo cliente.

AWS A modernização do mainframe usa sua chave gerenciada pelo cliente para as seguintes tarefas:

- Implantação de uma aplicação
- Substituindo uma instância de modernização de AWS mainframe do Amazon EC2.

AWS A modernização do mainframe não usa sua chave gerenciada pelo cliente para criptografar bancos de dados Amazon Relational Database Service ou Amazon Aurora, filas do Amazon Simple Queue Service e caches da ElastiCache Amazon criados para dar suporte a AWS um aplicativo de modernização de mainframe, porque nenhum deles contém dados do cliente.

Para obter mais informações, consulte Chaves mestras do cliente (CMKs) no AWS Key Management Service Guia do desenvolvedor.

A tabela a seguir resume como a modernização do AWS mainframe criptografa seus dados confidenciais.

| Tipo de dados | Chave gerenciada pela AWS criptografia | Criptografia de chave gerenciada pelo cliente |
| --- | --- | --- |
| `Definition`<br><br>Contém a definição de um aplicativo específico. | Habilitado | Habilitado |
| `EnvironmentSummary`<br><br>Contém informações sobre o ambiente de runtime. | Habilitado | Habilitado |
| `ApplicationSummary`<br><br>Contém informações sobre o aplicativo de modernização do AWS mainframe. | Habilitado | Habilitado |
| `DeploymentSummary`<br><br>Contém informações sobre a implantação de um aplicativ o de modernização de AWS mainframe. | Habilitado | Habilitado |

> **Note**
>
> AWS A modernização do mainframe ativa automaticamente a criptografia em repouso, usando Chaves gerenciadas pela AWS para proteger seus dados confidenciais sem nenhum custo. No entanto, AWS KMS cobranças são cobradas pelo uso de uma chave gerenciada pelo cliente. Para obter mais informações sobre precificação, consulte Precificação do AWS Key Management Service.

Para obter mais informações sobre AWS KMS, consulte AWS Key Management Service.

# Como a modernização AWS do mainframe usa subsídios em AWS KMS

AWS A modernização do mainframe exige uma concessão para usar sua chave gerenciada pelo cliente.

Quando você cria um aplicativo ou ambiente de execução, ou implanta um aplicativo na Modernização de AWS Mainframe criptografado com uma chave gerenciada pelo cliente, a Modernização de AWS Mainframe cria uma concessão em seu nome enviando uma solicitação para. CreateGrant AWS KMS As concessões AWS KMS são usadas para dar à modernização do AWS mainframe acesso a uma chave KMS em uma conta de cliente.

AWS A modernização do mainframe exige que a concessão use sua chave gerenciada pelo cliente para as seguintes operações internas:

- Envie DescribeKeysolicitações AWS KMS para verificar se a ID simétrica da chave gerenciada pelo cliente inserida ao criar um aplicativo, ambiente de execução ou implantação de aplicativo é válida.
- Envie GenerateDataKeysolicitações para AWS KMS criptografar o volume do Amazon EBS anexado às instâncias do Amazon EC2 que AWS hospedam ambientes de execução de modernização de mainframe.
- Envie solicitações de descriptografia para AWS KMS descriptografar conteúdo criptografado no Amazon EBS.

AWS A modernização do mainframe usa AWS KMS concessões para decifrar seus segredos armazenados no Secrets Manager e ao criar um ambiente de tempo de execução, criar ou reimplantar um aplicativo e criar uma implantação. Os subsídios criados pela modernização do AWS mainframe dão suporte às seguintes operações:

- Criar ou atualizar uma concessão de ambiente de runtime:
  - Decrypt
  - Encrypt
  - ReEncryptFrom
  - ReEncryptTo
  - GenerateDataKey
  - DescribeKey
  - CreateGrant

- Crie ou reimplante uma concessão de aplicativo:
  - GenerateDataKey
- Crie uma concessão de implantação:
  - Decrypt

É possível revogar o acesso à concessão, ou remover o acesso do serviço à chave gerenciada pelo cliente a qualquer momento. Se você fizer isso, a modernização do AWS mainframe não poderá acessar nenhum dado criptografado pela chave gerenciada pelo cliente, o que afeta as operações que dependem dos dados. Por exemplo, se a modernização do AWS mainframe tentasse acessar uma definição de aplicativo criptografada por uma chave gerenciada pelo cliente sem a concessão dessa chave, a operação de criação do aplicativo falharia.

AWS A modernização do mainframe coleta configurações de aplicativos do usuário (arquivos JSON) e artefatos (binários e executáveis). Ele também cria metadados que rastreiam várias entidades usadas para a operação do AWS Mainframe Modernization e cria registros e métricas. Os registros e métricas que são visíveis para o cliente incluem:

- CloudWatch registros que refletem o aplicativo e o mecanismo de tempo de execução ( AWS Blu Age ou Micro Focus).
- CloudWatch métricas para painéis de operação.

Além disso, a modernização do AWS mainframe coleta dados e métricas de uso para medição, relatórios de atividades e assim por diante sobre os serviços. Esses dados não são visíveis para o cliente.

AWS A modernização do mainframe armazena esses dados em locais diferentes, dependendo do tipo de dados. Os dados do cliente que você carrega são armazenados em um bucket do Amazon S3. Os dados do serviço são armazenados no Amazon S3 e no DynamoDB. Quando você implanta um aplicativo, tanto os dados quanto os dados do serviço são baixados nos volumes do Amazon EBS. Se você optar por conectar o armazenamento Amazon EFS ou Amazon FSx ao seu ambiente de execução, os dados armazenados nesses sistemas de arquivos também serão baixados para o volume do Amazon EBS.

A criptografia em repouso é configurada por padrão. Você não pode desativá-lo ou alterá-lo. No momento, também não é possível alterar a configuração.

# Criação de uma chave gerenciada pelo cliente

Você pode criar uma chave simétrica gerenciada pelo cliente usando as AWS Management Console ou as AWS KMS APIs.

Para criar uma chave simétrica gerenciada pelo cliente

Siga as etapas para criar uma chave simétrica gerenciada pelo cliente no Guia do desenvolvedor AWS Key Management Service .

Política de chave

As políticas de chaves controlam o acesso à chave gerenciada pelo seu cliente. Cada chave gerenciada pelo cliente deve ter exatamente uma política de chaves, que contém declarações que determinam quem pode usar a chave e como pode usá-la. Ao criar a chave gerenciada pelo cliente, você pode especificar uma política de chaves. Para obter mais informações, consulte Gerenciamento do acesso às chaves gerenciadas pelo cliente no Guia do desenvolvedor do AWS Key Management Service .

Para usar sua chave gerenciada pelo cliente com seus recursos de modernização de AWS mainframe, as seguintes operações de API devem ser permitidas na política de chaves:

- `kms:CreateGrant`: adiciona uma concessão a uma chave gerenciada pelo cliente. Concede acesso de controle a uma chave KMS especificada, o que permite o acesso às operações de concessão exigidas pela modernização do AWS mainframe. Para obter mais informações, consulte Usar concessões no Guia do desenvolvedor do AWS Key Management Service .

  Isso permite que a modernização do AWS mainframe faça o seguinte:

  - Ligar para `GenerateDataKey` para gerar uma chave de dados criptografada e armazená-la, porque a chave de dados não é usada imediatamente para criptografar.
  - Ligue `Decrypt` para usar a chave de dados criptografada armazenada para acessar os dados criptografados.
  - Configure uma entidade principal aposentada para permitir que o serviço para `RetireGrant`.
- `kms:DescribeKey`— fornece os principais detalhes gerenciados pelo cliente para permitir que a modernização do AWS mainframe valide a chave.

AWS A modernização do mainframe exige `kms:CreateGrant` `kms:DescribeKey` permissões na política principal do cliente. AWS A modernização do mainframe usa essa política para criar uma concessão para si mesma.

```
{
    "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**
>
> A função mostrada `Principal` no exemplo anterior é aquela que você usa para operações de modernização de AWS mainframe, como e. `CreateApplication` `CreateEnvironment`

Para obter mais informações sobre specifying permissions in a policy, consulte o Guia do Desenvolvedor AWS Key Management Service .

Para obter mais informações sobre solução de problemas de acesso à chave, consulte o AWS Key Management Service Guia do Desenvolvedor.

## Especificação de uma chave gerenciada pelo cliente para o AWS Mainframe Modernization

Você pode especificar uma chave gerenciada pelo cliente para os seguintes recursos:

- Aplicativo
- Ambiente

Ao criar um recurso, você pode especificar a chave inserindo uma ID do KMS, que a Modernização do AWS Mainframe usa para criptografar os dados confidenciais armazenados pelo recurso.

- KMS ID - Um identificador de chave para uma chave gerenciada pelo cliente. Insira uma ID de chave, um ARN de chave, um nome de alias ou um ARN de alias.

Você pode especificar uma chave gerenciada pelo cliente usando o AWS Management Console ou AWS CLI o.

Para especificar sua chave gerenciada pelo cliente ao criar um ambiente de tempo de execução no AWS Management Console, consulteCrie um ambiente de tempo de execução do AWS Mainframe Modernization. Para especificar sua chave gerenciada pelo cliente ao criar um aplicativo no AWS Management Console, consulteCriar um aplicativo do AWS Mainframe Modernization.

Para adicionar sua chave gerenciada pelo cliente ao criar um ambiente de tempo de execução com o AWS CLI, especifique o `kms-key-id` parâmetro da seguinte forma:

```
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
```

Para adicionar sua chave gerenciada pelo cliente ao criar um aplicativo com o AWS CLI, especifique o `kms-key-id` parâmetro da seguinte forma:

```
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 Contexto de criptografia da modernização do mainframe

Um contexto de criptografia é um conjunto opcional de pares chave-valor que contêm informações contextuais adicionais sobre os dados.

AWS KMS usa o contexto de criptografia como dados autenticados adicionais para oferecer suporte à criptografia autenticada. Quando você inclui um contexto de criptografia em uma solicitação para criptografar dados, AWS KMS vincula o contexto de criptografia aos dados criptografados. Para descriptografar os dados, você inclui o mesmo contexto de criptografia na solicitação.

## AWS Contexto de criptografia da modernização do mainframe

AWS A modernização do mainframe usa o mesmo contexto de criptografia em todas as operações AWS KMS criptográficas relacionadas a um aplicativo (criar aplicativo e criar implantação), onde a chave está `aws:m2:app` e o valor é o identificador exclusivo do aplicativo.

Example

```
"encryptionContextSubset": {
        "aws:m2:app": "a1bc2defabc3defabc4defabcd"
}
```

### Uso do contexto de criptografia para monitoramento

Quando você usa uma chave simétrica gerenciada pelo cliente para criptografar seus aplicativos ou ambientes de tempo de execução, também pode usar o contexto de criptografia em registros e logs de auditoria para identificar como a chave gerenciada pelo cliente está sendo usada.

### Uso do contexto de criptografia para controlar o acesso à chave gerenciada pelo cliente

Você pode usar o contexto de criptografia nas políticas de chaves e políticas do IAM como `conditions` e controlar o acesso à sua chave simétrica gerenciada pelo cliente. Você também pode usar restrições no contexto de criptografia em uma concessão.

AWS A modernização do mainframe usa uma restrição de contexto de criptografia nas concessões para controlar o acesso à chave gerenciada pelo cliente em sua conta ou região. A restrição de concessão exige que as operações permitidas pela concessão usem o contexto de criptografia especificado. O exemplo a seguir é uma concessão que a modernização do AWS mainframe aproveita para criptografar artefatos do aplicativo ao criar um aplicativo.

```
//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”
```

```
    }
}
```

# Monitoramento de suas chaves de criptografia para AWS Mainframe Modernization

Ao usar uma chave gerenciada pelo AWS KMS cliente com seus recursos de modernização de AWS mainframe, você pode usar o AWS CloudTrailAmazon CloudWatch Logs para rastrear solicitações enviadas pela modernização de AWS mainframe. AWS KMS

## Exemplos de ambientes de runtime

Os exemplos a seguir são AWS CloudTrail eventos para`DescribeKey`, `CreateGrantGenerateDataKey`, e `Decrypt` para monitorar as operações do KMS chamadas pela modernização do AWS mainframe para acessar dados criptografados pela chave gerenciada pelo cliente:

DescribeKey

AWS A modernização do mainframe usa a `DescribeKey` operação para verificar se a chave gerenciada pelo AWS KMS cliente associada ao seu ambiente de tempo de execução existe na conta e na região.

O evento de exemplo a seguir registra a operação `DescribeKey`:

```
{
    "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

Quando você usa uma chave gerenciada pelo AWS KMS cliente para criptografar seu ambiente de execução, a modernização do AWS mainframe envia várias `CreateGrant` solicitações em

seu nome para realizar as operações necessárias do KMS. Algumas das concessões criadas pela modernização do AWS mainframe são retiradas imediatamente após o uso. Outros são retirados quando você exclui o ambiente de execução.

O evento de exemplo a seguir registra a operação `CreateGrant` da função de execução do Lambda associada ao fluxo de trabalho 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"
}
```

O exemplo de evento a seguir registra a operação `CreateGrant` para a função vinculada ao serviço do grupo do Auto Scaling. A função de execução do Lambda associada ao fluxo de trabalho Create Environment chama essa operação `CreateGrant`. Ele concede permissão para que a função de execução crie uma subconcessão em relação à função vinculada ao serviço do grupo 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

Quando você habilita uma chave gerenciada pelo AWS KMS cliente para seu recurso de ambiente de tempo de execução, o Auto Scaling cria uma chave exclusiva para criptografar o volume do Amazon EBS associado ao ambiente de tempo de execução. Ele envia uma `GenerateDataKey` solicitação AWS KMS que especifica a chave gerenciada pelo AWS KMS cliente para o recurso.

O evento de exemplo a seguir registra a operação `GenerateDataKey`:

```
{
    "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

Quando você acessa um ambiente de tempo de execução criptografado, o Amazon EBS chama a operação `Decrypt` para usar a chave de dados criptografados armazenada para acessar os dados criptografados.

O evento de exemplo a seguir registra a operação `Decrypt`:

```
{
    "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"
}
```

## Exemplos de aplicativos

Os exemplos a seguir são AWS CloudTrail eventos para `CreateGrant` e `GenerateDataKey` para monitorar as operações do KMS chamadas pela modernização do AWS mainframe para acessar dados criptografados pela chave gerenciada pelo cliente:

CreateGrant

Quando você usa uma chave gerenciada pelo AWS KMS cliente para criptografar os recursos do seu aplicativo, a função de execução do Lambda envia `CreateGrant` uma solicitação em seu nome para acessar a chave KMS em sua conta. AWS A concessão permite que a função de execução do Lambda carregue recursos de aplicativos do cliente para o Amazon S3 usando sua chave gerenciada pelo cliente. Esse subsídio é retirado imediatamente após a criação do aplicativo.

O evento de exemplo a seguir registra a operação `CreateGrant`:

```
{
    "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

Quando você habilita uma chave gerenciada pelo AWS KMS cliente para seu recurso de aplicativo, a função de execução do Lambda cria uma chave que é usada para criptografar e carregar dados do cliente para o Amazon Simple Storage Service. A função de execução do Lambda envia uma `GenerateDataKey` solicitação AWS KMS que especifica a chave gerenciada pelo AWS KMS cliente para o recurso.

O evento de exemplo a seguir registra a operação `GenerateDataKey`:

```
{
    "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"
```

```
}
```

## Exemplos de implantações

Os exemplos a seguir são AWS CloudTrail eventos para `CreateGrant` e `Decrypt` para monitorar as operações do KMS chamadas pela modernização do AWS mainframe para acessar dados criptografados pela chave gerenciada pelo cliente:

CreateGrant

Quando você usa uma chave gerenciada pelo AWS KMS cliente para criptografar seus recursos de implantação, a modernização do AWS mainframe envia duas `CreateGrant` solicitações em seu nome. A primeira concessão é atribuída à função de execução atual do Lambda a ser chamada ListBatchJobScriptFiles e é retirada imediatamente após o término da implantação. A segunda concessão é destinada à função de instância com escopo reduzido do Amazon EC2, para que o Amazon EC2 possa baixar recursos de aplicativos de clientes do Amazon S3. Essa concessão é retirada quando o aplicativo é excluído do ambiente de execução.

O evento de exemplo a seguir registra a operação `CreateGrant`:

```
{
    "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

Quando você acessa uma implantação, o Amazon EC2 chama a operação `Decrypt` para usar a chave de dados criptografada armazenada para descriptografar e baixar dados criptografados do cliente do Amazon S3.

O evento de exemplo a seguir registra a operação `Decrypt`:

```
{
    "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"
}
```

## Saiba mais

Os recursos a seguir fornecem mais informações sobre a criptografia de dados em pausa.

- Para obter mais informações sobre AWS Key Management Service conceitos básicos, consulte o AWS Key Management Service Guia do Desenvolvedor.
- Para obter mais informações sobre as melhores práticas de segurança para AWS Key Management Service, consulte o Guia do desenvolvedor AWS Key Management Service .

## Criptografia em trânsito

Para aplicativos interativos que fazem parte de cargas de trabalho transacionais, as trocas de dados entre o emulador de terminal e o endpoint do serviço de modernização de AWS mainframe para o protocolo TN3270 não são criptografadas em trânsito. Se o aplicativo exigir criptografia em trânsito, talvez você queira implementar alguns mecanismos adicionais de tunelamento.

AWS A modernização do mainframe usa HTTPS para criptografar as APIs do serviço. Todas as outras comunicações na modernização do AWS mainframe são protegidas pelo serviço VPC ou pelo grupo de segurança, bem como pelo HTTPS. AWS A modernização do mainframe transfere artefatos, configurações e dados do aplicativo. Os artefatos do aplicativo são copiados de um bucket do Amazon S3 que você possui, assim como os dados do aplicativo. Você pode fornecer configurações de aplicativos usando um link para o Amazon S3 ou fazendo o upload de um arquivo localmente.

A criptografia básica em trânsito é configurada por padrão, mas não se aplica ao protocolo TN3270. AWS A modernização do mainframe usa HTTPS para endpoints de API, que também são configurados por padrão.

# Identity and Access Management para modernização AWS do mainframe

AWS Identity and Access Management (IAM) é uma ferramenta AWS service (Serviço da AWS) que ajuda o administrador a controlar com segurança o acesso aos AWS recursos. Os administradores do IAM controlam quem pode ser autenticado (conectado) e autorizado (tem permissões) a usar os recursos de modernização do AWS mainframe. O IAM é um AWS service (Serviço da AWS) que você pode usar sem custo adicional.

Tópicos



## Público

A forma como você usa o AWS Identity and Access Management (IAM) difere, dependendo do trabalho que você faz na modernização do AWS mainframe.

Usuário do serviço — Se você usa o serviço de modernização do AWS mainframe para fazer seu trabalho, seu administrador fornecerá as credenciais e as permissões de que você precisa. À medida que você usa mais recursos de modernização de AWS mainframe para fazer seu trabalho, talvez precise de permissões adicionais. Entender como o acesso é gerenciado pode ajudá-lo a solicitar as permissões corretas ao seu administrador. Se você não conseguir acessar um recurso no AWS Mainframe Modernization, consulte Solução de problemas de identidade e AWS acesso à modernização do mainframe

Administrador de serviços — Se você é responsável pelos recursos de modernização de AWS mainframe em sua empresa, provavelmente tem acesso total à modernização de AWS mainframe. É seu trabalho determinar quais recursos e recursos de modernização de AWS mainframe seus usuários do serviço devem acessar. Assim, você deve enviar solicitações ao administrador do IAM para alterar as permissões dos usuários de seu serviço. Revise as informações nesta página para entender os Introdução ao IAM. Para saber mais sobre como sua empresa pode usar o IAM com a modernização do AWS mainframe, consulte. Como a modernização AWS do mainframe funciona com o IAM

Administrador do IAM — Se você for administrador do IAM, talvez queira saber detalhes sobre como criar políticas para gerenciar o acesso à modernização do AWS mainframe. Para ver exemplos de políticas baseadas em identidade de modernização de AWS mainframe que você pode usar no IAM, consulte. Exemplos de políticas baseadas em identidade para AWS modernização de mainframe

## Autenticando com identidades

A autenticação é a forma como você faz login AWS usando suas credenciais de identidade. Você deve estar autenticado (conectado AWS) como o Usuário raiz da conta da AWS, como usuário do IAM ou assumindo uma função do IAM.

Você pode entrar AWS como uma identidade federada usando credenciais fornecidas por meio de uma fonte de identidade. AWS IAM Identity Center Usuários (IAM Identity Center), a autenticação de login único da sua empresa e suas credenciais do Google ou do Facebook são exemplos de identidades federadas. Quando você faz login como uma identidade federada, o administrador já configurou anteriormente a federação de identidades usando perfis do IAM. Ao acessar AWS usando a federação, você está assumindo indiretamente uma função.

Dependendo do tipo de usuário que você é, você pode entrar no AWS Management Console ou no portal de AWS acesso. Para obter mais informações sobre como fazer login em AWS, consulte Como fazer login Conta da AWS no Guia do Início de Sessão da AWS usuário.

Se você acessar AWS programaticamente, AWS fornece um kit de desenvolvimento de software (SDK) e uma interface de linha de comando (CLI) para assinar criptograficamente suas solicitações usando suas credenciais. Se você não usa AWS ferramentas, você mesmo deve assinar as solicitações. Para obter mais informações sobre como usar o método recomendado para assinar solicitações por conta própria, consulte Assinatura de solicitações de AWS API no Guia do usuário do IAM.

Independentemente do método de autenticação usado, também pode ser exigido que você forneça informações adicionais de segurança. Por exemplo, AWS recomenda que você use a autenticação multifator (MFA) para aumentar a segurança da sua conta. Para saber mais, consulte Autenticação multifator no Guia do usuário do AWS IAM Identity Center e Usar a autenticação multifator (MFA) na AWS no Guia do usuário do IAM.

## Conta da AWS usuário root

Ao criar uma Conta da AWS, você começa com uma identidade de login que tem acesso completo a todos Serviços da AWS os recursos da conta. Essa identidade é chamada de usuário Conta da AWS raiz e é acessada fazendo login com o endereço de e-mail e a senha que você usou para criar a conta. É altamente recomendável não usar o usuário raiz para tarefas diárias. Proteja as credenciais do usuário raiz e use-as para executar as tarefas que somente ele pode executar. Para obter a lista completa das tarefas que exigem login como usuário raiz, consulte Tarefas que exigem credenciais de usuário raiz no Guia do usuário do IAM.

## Identidade federada

Como prática recomendada, exija que usuários humanos, incluindo usuários que precisam de acesso de administrador, usem a federação com um provedor de identidade para acessar Serviços da AWS usando credenciais temporárias.

Uma identidade federada é um usuário do seu diretório de usuários corporativo, de um provedor de identidade da web AWS Directory Service, do diretório do Identity Center ou de qualquer usuário que acesse usando credenciais fornecidas Serviços da AWS por meio de uma fonte de identidade. Quando as identidades federadas são acessadas Contas da AWS, elas assumem funções, e as funções fornecem credenciais temporárias.

Para o gerenciamento de acesso centralizado, recomendamos usar o AWS IAM Identity Center. Você pode criar usuários e grupos no IAM Identity Center ou pode se conectar e sincronizar com um conjunto de usuários e grupos em sua própria fonte de identidade para uso em todos os seus Contas

da AWS aplicativos. Para obter mais informações sobre o Centro de Identidade do IAM, consulte O que é o Centro de Identidade do IAM? no Guia do usuário do AWS IAM Identity Center .

## Grupos e usuários do IAM

Um usuário do IAM é uma identidade dentro da sua Conta da AWS que tem permissões específicas para uma única pessoa ou aplicativo. Sempre que possível, recomendamos depender de credenciais temporárias em vez de criar usuários do IAM com credenciais de longo prazo, como senhas e chaves de acesso. No entanto, se você tiver casos de uso específicos que exijam credenciais de longo prazo com usuários do IAM, recomendamos alternar as chaves de acesso. Para obter mais informações, consulte Alterne as chaves de acesso regularmente para casos de uso que exijam credenciais de longo prazo no Guia do usuário do IAM.

Um grupo do IAM é uma identidade que especifica uma coleção de usuários do IAM. Não é possível fazer login como um grupo. É possível usar grupos para especificar permissões para vários usuários de uma vez. Os grupos facilitam o gerenciamento de permissões para grandes conjuntos de usuários. Por exemplo, você pode ter um grupo chamado IAMAdmins e atribuir a esse grupo permissões para administrar recursos do IAM.

Usuários são diferentes de perfis. Um usuário é exclusivamente associado a uma pessoa ou a uma aplicação, mas um perfil pode ser assumido por qualquer pessoa que precisar dele. Os usuários têm credenciais permanentes de longo prazo, mas os perfis fornecem credenciais temporárias. Para saber mais, consulte Quando criar um usuário do IAM (em vez de um perfil) no Guia do usuário do IAM.

## Perfis do IAM

Uma função do IAM é uma identidade dentro da sua Conta da AWS que tem permissões específicas. Ela é semelhante a um usuário do IAM, mas não está associada a uma pessoa específica. Você pode assumir temporariamente uma função do IAM no AWS Management Console trocando de funções. Você pode assumir uma função chamando uma operação de AWS API AWS CLI ou usando uma URL personalizada. Para obter mais informações sobre métodos para o uso de perfis, consulte Usar perfis do IAM no Guia do usuário do IAM.

Perfis do IAM com credenciais temporárias são úteis nas seguintes situações:

- Acesso de usuário federado: para atribuir permissões a identidades federadas, você pode criar um perfil e definir permissões para ele. Quando uma identidade federada é autenticada, essa identidade é associada ao perfil e recebe as permissões definidas pelo mesmo. Para obter mais

informações sobre perfis para federação, consulte Criar um perfil para um provedor de identidades de terceiros no Guia do usuário do IAM. Se você usar o IAM Identity Center, configure um conjunto de permissões. Para controlar o que suas identidades podem acessar após a autenticação, o IAM Identity Center correlaciona o conjunto de permissões a um perfil no IAM. Para obter informações sobre conjuntos de permissões, consulte Conjuntos de permissões no Guia do usuário do AWS IAM Identity Center .

- Permissões temporárias para usuários do IAM: um usuário ou um perfil do IAM pode assumir um perfil do IAM para obter temporariamente permissões diferentes para uma tarefa específica.
- Acesso entre contas: é possível usar um perfil do IAM para permitir que alguém (uma entidade principal confiável) em outra conta acesse recursos em sua conta. Os perfis são a principal forma de conceder acesso entre contas. No entanto, com alguns Serviços da AWS, você pode anexar uma política diretamente a um recurso (em vez de usar uma função como proxy). Para saber a diferença entre perfis e políticas baseadas em recurso para acesso entre contas, consulte Como os perfis do IAM diferem das políticas baseadas em recurso no Guia do usuário do IAM.
- Acesso entre serviços — Alguns Serviços da AWS usam recursos em outros Serviços da AWS. Por exemplo, quando você faz uma chamada em um serviço, é comum que esse serviço execute aplicações no Amazon EC2 ou armazene objetos no Amazon S3. Um serviço pode fazer isso usando as permissões da entidade principal da chamada, usando um perfil de serviço ou uma função vinculada ao serviço.
  - Sessões de acesso direto (FAS) — Quando você usa um usuário ou uma função do IAM para realizar ações AWS, você é considerado principal. Ao usar alguns serviços, você pode executar uma ação que inicia outra ação em um serviço diferente. O FAS usa as permissões do diretor chamando um AWS service (Serviço da AWS), combinadas com a solicitação AWS service (Serviço da AWS) para fazer solicitações aos serviços posteriores. As solicitações do FAS são feitas somente quando um serviço recebe uma solicitação que requer interações com outros Serviços da AWS ou com recursos para ser concluída. Nesse caso, você precisa ter permissões para executar ambas as ações. Para obter detalhes da política ao fazer solicitações de FAS, consulte Encaminhar sessões de acesso.
  - Perfil de serviço: um perfil de serviço é um perfil do IAM que um serviço assume para realizar ações em seu nome. Um administrador do IAM pode criar, modificar e excluir um perfil de serviço do IAM. Para obter mais informações, consulte Criar um perfil para delegar permissões a um AWS service (Serviço da AWS) no Guia do usuário do IAM.
  - Função vinculada ao serviço — Uma função vinculada ao serviço é um tipo de função de serviço vinculada a um. AWS service (Serviço da AWS) O serviço pode assumir a função de executar uma ação em seu nome. As funções vinculadas ao serviço aparecem em você Conta da AWS e

são de propriedade do serviço. Um administrador do IAM pode visualizar, mas não pode editar as permissões para perfis vinculados ao serviço.

- Aplicativos em execução no Amazon EC2 — Você pode usar uma função do IAM para gerenciar credenciais temporárias para aplicativos que estão sendo executados em uma instância do EC2 e fazendo AWS CLI solicitações de API. AWS É preferível fazer isso a armazenar chaves de acesso na instância do EC2. Para atribuir uma AWS função a uma instância do EC2 e disponibilizá-la para todos os seus aplicativos, você cria um perfil de instância anexado à instância. Um perfil de instância contém o perfil e permite que os programas em execução na instância do EC2 obtenham credenciais temporárias. Para mais informações, consulte Usar um perfil do IAM para conceder permissões a aplicações em execução nas instâncias do Amazon EC2 no Guia do usuário do IAM.

Para saber se deseja usar os perfis do IAM, consulte Quando criar um perfil do IAM (em vez de um usuário) no Guia do usuário do IAM.

## Gerenciamento do acesso usando políticas

Você controla o acesso AWS criando políticas e anexando-as a AWS identidades ou recursos. Uma política é um objeto AWS que, quando associada a uma identidade ou recurso, define suas permissões. AWS avalia essas políticas quando um principal (usuário, usuário raiz ou sessão de função) faz uma solicitação. As permissões nas políticas determinam se a solicitação será permitida ou negada. A maioria das políticas é armazenada AWS como documentos JSON. Para obter mais informações sobre a estrutura e o conteúdo de documentos de políticas JSON, consulte Visão geral das políticas JSON no Guia do usuário do IAM.

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

Por padrão, usuários e funções não têm permissões. Para conceder aos usuários permissão para executar ações nos recursos de que eles precisam, um administrador do IAM pode criar políticas do IAM. O administrador pode então adicionar as políticas do IAM a perfis, e os usuários podem assumir os perfis.

As políticas do IAM definem permissões para uma ação, independentemente do método usado para executar a operação. Por exemplo, suponha que você tenha uma política que permite a ação `iam:GetRole`. Um usuário com essa política pode obter informações de função da AWS Management Console AWS CLI, da ou da AWS API.

## Políticas baseadas em identidade

As políticas baseadas em identidade são documentos de políticas de permissões JSON que você pode anexar a uma identidade, como um usuário do IAM, grupo de usuários ou função do IAM. Essas políticas controlam quais ações os usuários e perfis podem realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte Criação de política do IAM no Guia do usuário do IAM.

As políticas baseadas em identidade podem ser categorizadas ainda mais como políticas em linha ou políticas gerenciadas. As políticas em linha são anexadas diretamente a um único usuário, grupo ou perfil. As políticas gerenciadas são políticas autônomas que você pode associar a vários usuários, grupos e funções em seu Conta da AWS. As políticas AWS gerenciadas incluem políticas gerenciadas e políticas gerenciadas pelo cliente. Para saber como escolher entre uma política gerenciada ou uma política em linha, consulte Escolher entre políticas gerenciadas e políticas em linha no Guia do usuário do IAM.

## Políticas baseadas em recurso

Políticas baseadas em recurso são documentos de políticas JSON que você anexa a um recurso. São exemplos de políticas baseadas em recursos as políticas de confiança de perfil do IAM e as políticas de bucket do Amazon S3. Em serviços compatíveis com políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o recurso ao qual a política está anexada, a política define quais ações uma entidade principal especificada pode executar nesse recurso e em que condições. Você deve especificar uma entidade principal em uma política baseada em recursos. Os diretores podem incluir contas, usuários, funções, usuários federados ou. Serviços da AWS

Políticas baseadas em recursos são políticas em linha que estão localizadas nesse serviço. Você não pode usar políticas AWS gerenciadas do IAM em uma política baseada em recursos.

## Listas de controle de acesso (ACLs)

As listas de controle de acesso (ACLs) controlam quais entidades principais (membros, usuários ou perfis da conta) têm permissões para acessar um recurso. As ACLs são semelhantes às políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

O Amazon S3 e o Amazon VPC são exemplos de serviços que oferecem suporte a ACLs. AWS WAF Para saber mais sobre ACLs, consulte Visão geral da lista de controle de acesso (ACL) no Guia do desenvolvedor do Amazon Simple Storage Service.

## Outros tipos de política

AWS oferece suporte a tipos de políticas adicionais menos comuns. Esses tipos de política podem definir o máximo de permissões concedidas a você pelos tipos de política mais comuns.

- Limites de permissões: um limite de permissões é um atributo avançado no qual você define o máximo de permissões que uma política baseada em identidade pode conceder a uma entidade do IAM (usuário ou perfil do IAM). É possível definir um limite de permissões para uma entidade. As permissões resultantes são a interseção das políticas baseadas em identidade de uma entidade e dos seus limites de permissões. As políticas baseadas em recurso que especificam o usuário ou a função no campo `Principal` não são limitadas pelo limite de permissões. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações sobre limites de permissões, consulte Limites de permissões para identidades do IAM no Guia do usuário do IAM.
- Políticas de controle de serviço (SCPs) — SCPs são políticas JSON que especificam as permissões máximas para uma organização ou unidade organizacional (OU) em. AWS Organizations AWS Organizations é um serviço para agrupar e gerenciar centralmente vários Contas da AWS que sua empresa possui. Se você habilitar todos os atributos em uma organização, poderá aplicar políticas de controle de serviço (SCPs) a qualquer uma ou a todas as contas. O SCP limita as permissões para entidades nas contas dos membros, incluindo cada uma Usuário raiz da conta da AWS. Para obter mais informações sobre o Organizations e SCPs, consulte Como os SCPs funcionam no Guia do usuário do AWS Organizations .
- Políticas de sessão: são políticas avançadas que você transmite como um parâmetro quando cria de forma programática uma sessão temporária para um perfil ou um usuário federado. As permissões da sessão resultante são a interseção das políticas baseadas em identidade do usuário ou do perfil e das políticas de sessão. As permissões também podem ser provenientes de uma política baseada em recurso. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações, consulte Políticas de sessão no Guia do usuário do IAM.

## Vários tipos de política

Quando vários tipos de política são aplicáveis a uma solicitação, é mais complicado compreender as permissões resultantes. Para saber como AWS determinar se uma solicitação deve ser permitida quando vários tipos de políticas estão envolvidos, consulte Lógica de avaliação de políticas no Guia do usuário do IAM.

# Como a modernização AWS do mainframe funciona com o IAM

Antes de usar o IAM para gerenciar o acesso à modernização do AWS mainframe, saiba quais recursos do IAM estão disponíveis para uso com a modernização do AWS mainframe.

Recursos do IAM que você pode usar com a modernização AWS do mainframe

| Recurso do IAM | AWS Suporte à modernização do mainframe |
|---|---|
| Políticas baseadas em identidade | Sim |
| Políticas baseadas em recursos | Não |
| Ações de políticas | Sim |
| Recursos de políticas | Sim |
| Chaves de condição de políticas | Sim |
| ACLs | Não |
| ABAC (tags em políticas) | Sim |
| Credenciais temporárias | Sim |
| Sessões de acesso direto (FAS) | Sim |
| Perfis de serviço | Sim |
| Perfis vinculados ao serviço | Sim |

Para ter uma visão de alto nível de como a modernização do AWS mainframe e outros AWS serviços funcionam com a maioria dos recursos do IAM, consulte AWS os serviços que funcionam com o IAM no Guia do usuário do IAM.

## Políticas baseadas em identidade para AWS modernização do mainframe

| | |
|---|---|
| É compatível com políticas baseadas em identidade | Sim |

As políticas baseadas em identidade são documentos de políticas de permissões JSON que você pode anexar a uma identidade, como usuário, grupo de usuários ou perfil do IAM. Essas políticas controlam quais ações os usuários e perfis podem realizar, em quais recursos e em que condições. ‘Para saber como criar uma política baseada em identidade, consulte Criar políticas do IAM no Guia do usuário do IAM.

Com as políticas baseadas em identidade do IAM, é possível especificar ações ou recursos permitidos ou negados, bem como as condições sob as quais as ações são permitidas ou negadas. Você não pode especificar a entidade principal em uma política baseada em identidade porque ela se aplica ao usuário ou função à qual ela está anexada. Para saber mais sobre todos os elementos que podem ser usados em uma política JSON, consulte Referência de elementos da política JSON do IAM no Guia do Usuário do IAM.

Exemplos de políticas baseadas em identidade para AWS modernização de mainframe

Para ver exemplos de políticas baseadas em identidade de modernização de AWS mainframe, consulte. Exemplos de políticas baseadas em identidade para AWS modernização de mainframe

## Políticas baseadas em recursos na modernização do mainframe AWS

| | |
|---|---|
| Oferece suporte a políticas baseadas em recursos | Não |

Políticas baseadas em recurso são documentos de políticas JSON que você anexa a um recurso. São exemplos de políticas baseadas em recursos as políticas de confiança de perfil do IAM e as políticas de bucket do Amazon S3. Em serviços compatíveis com políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o recurso ao qual a política está anexada, a política define quais ações uma entidade principal especificada pode executar nesse recurso e em que condições. Você deve especificar uma entidade principal em uma política baseada em recursos. Os diretores podem incluir contas, usuários, funções, usuários federados ou. Serviços da AWS

Para permitir o acesso entre contas, você pode especificar uma conta inteira ou as entidades do IAM em outra conta como a entidade principal em uma política baseada em recurso. Adicionar um principal entre contas à política baseada em recurso é apenas metade da tarefa de estabelecimento da relação de confiança. Quando o principal e o recurso são diferentes Contas da AWS, um

administrador do IAM na conta confiável também deve conceder permissão à entidade principal (usuário ou função) para acessar o recurso. Eles concedem permissão ao anexar uma política baseada em identidade para a entidade. No entanto, se uma política baseada em recurso conceder acesso a um principal na mesma conta, nenhuma política baseada em identidade adicional será necessária. Para obter mais informações, consulte Como os perfis do IAM diferem de políticas baseadas em recursos no Guia do usuário do IAM.

## Ações políticas para a modernização AWS do mainframe

| | |
|---|---|
| Oferece suporte a ações de políticas | Sim |

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos, e em que condições.

O elemento `Action` de uma política JSON descreve as ações que você pode usar para permitir ou negar acesso em uma política. As ações de política geralmente têm o mesmo nome da operação de AWS API associada. Existem algumas exceções, como ações somente de permissão, que não têm uma operação de API correspondente. Há também algumas operações que exigem várias ações em uma política. Essas ações adicionais são chamadas de ações dependentes.

Incluem ações em uma política para conceder permissões para executar a operação associada.

Para ver uma lista das ações de modernização do AWS mainframe, consulte Ações definidas pela modernização do AWS mainframe na Referência de autorização de serviço.

As ações de política na modernização do AWS mainframe usam o seguinte prefixo antes da ação:

```
m2
```

Para especificar várias ações em uma única instrução, separe-as com vírgulas.

```
"Action": [
      "m2:StartApplication",
      "m2:StopApplication"
         ]
```

Você também pode especificar várias ações usando caracteres-curinga (*). Por exemplo, para especificar todas as ações que começam com a palavra `List`, inclua a seguinte ação:

```
"Action": "m2:List*"
```

Para ver exemplos de políticas baseadas em identidade de modernização de AWS mainframe, consulte. Exemplos de políticas baseadas em identidade para AWS modernização de mainframe

## Recursos de políticas para a modernização AWS do mainframe

| | |
|---|---|
| Oferece suporte a recursos de políticas | Sim |

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos, e em que condições.

O elemento `Resource` de política JSON especifica o objeto ou os objetos aos quais a ação se aplica. As instruções devem incluir um elemento `Resource` ou um elemento `NotResource`. Como prática recomendada, especifique um recurso usando seu nome do recurso da Amazon (ARN). Isso pode ser feito para ações que oferecem suporte a um tipo de recurso específico, conhecido como permissões em nível de recurso.

Para ações que não oferecem suporte a permissões em nível de recurso, como operações de listagem, use um caractere curinga (*) para indicar que a instrução se aplica a todos os recursos.

```
"Resource": "*"
```

Você pode restringir o acesso a recursos específicos de modernização de AWS mainframe usando seus ARNs para identificar o recurso ao qual a política do IAM se aplica. Para obter mais informações sobre o formato de ARNs, consulte Nomes de recursos da Amazon (ARNs) na Referência geral da AWS.

Por exemplo, um ambiente de modernização de AWS mainframe tem o seguinte ARN.

```
"Resource": "arn:aws:m2:regionId:accountId:env/service-generated-unique-identifier"
```

Um aplicativo de modernização de AWS mainframe tem o seguinte ARN.

```
"Resource": "arn:aws:m2:regionId:accountId:app/service-generated-unique-identifier"
```

Nem todas as ações de modernização do AWS mainframe oferecem suporte a permissões em nível de recurso. Para ações que não suportam permissões em nível de recurso, você deve usar o curinga (*).

As ações de modernização do AWS mainframe a seguir não oferecem suporte a permissões em nível de recurso.

```
ListApplications
            ListApplicationVersions
            ListBatchJobDefinitions
            ListBatchJobExecutions
            ListDataSetImportHistory
            ListDataSets
            ListDeployments
            ListEngineVersions
            ListEnvironments
            ListTagsForResource
```

Para ver uma lista dos tipos de recursos de modernização do AWS mainframe e seus ARNs, consulte Recursos definidos pela modernização do AWS mainframe na Referência de autorização de serviço. Para saber com quais ações você pode especificar o ARN de cada recurso, consulte Ações definidas pela modernização do AWS mainframe.

Para ver exemplos de políticas baseadas em identidade de modernização de AWS mainframe, consulte. Exemplos de políticas baseadas em identidade para AWS modernização de mainframe

## AWS Permissões da API de modernização de mainframe: referência de ações, recursos e condições

Ao escrever políticas de permissões que podem ser anexadas a uma identidade do IAM (políticas baseadas em identidade), você pode usar a tabela a seguir como referência. A tabela inclui o seguinte:

- Cada operação da API de modernização de AWS mainframe
- As ações correspondentes para as quais você pode conceder permissões para executar a ação

- O AWS recurso para o qual você pode conceder as permissões

Especifique as ações no campo `Action` da política e o valor do recurso no campo `Resource` da política.

Você pode usar chaves de condição AWS globais em suas políticas de modernização de AWS mainframe para expressar condições. Para obter uma lista completa das AWS chaves, consulte Chaves de condição globais disponíveis no Guia do usuário do IAM.

> **Note**
>
> Para especificar uma ação, use o prefixo `m2:` seguido do nome da operação da API (por exemplo, `m2:CreateApplication`).

AWS API de modernização de mainframe e permissões necessárias para ações

| AWS Operações de API de modernização de mainframe | Permissões obrigatórias (ações de API) | Recursos |
|---|---|---|
| CancelBatchJobExecution | | Aplicativo |
| CreateApplication | `s3:GetObject`<br>`s3:ListBucket` | Aplicativo |
| CreateDataSetImportTask | `m2:CreateDataSetImportTask`<br>`s3:GetObject` | Aplicativo |
| CreateDeployment | `elasticloadbalancing:AddTags`<br>`elasticloadbalancing:CreateListener`<br>`elasticloadbalancing:CreateTargetGroup` | Aplicativo |

| AWS Operações de API de modernização de mainframe | Permissões obrigatórias (ações de API) | Recursos |
| --- | --- | --- |
| | `elasticloadbalancing:RegisterTargets` | |
| CreateEnvironment | `ec2:CreateNetworkInterface`<br>`ec2:CreateNetworkInterfacePermission`<br>`ec2:DescribeNetworkInterfaces`<br>`ec2:DescribeSecurityGroups`<br>`ec2:DescribeSubnets`<br>`ec2:DescribeVpcAttribute`<br>`ec2:DescribeVpcs`<br>`ec2:ModifyNetworkInterfaceAttribute`<br>`elasticfilesystem:DescribeMountTargets`<br>`elasticloadbalancing:AddTags`<br>`elasticloadbalancing:CreateLoadBalancer`<br>`fsx:DescribeFileSystems`<br>`iam:CreateServiceLinkedRole` | Ambiente |

| AWS Operações de API de modernização de mainframe | Permissões obrigatórias (ações de API) | Recursos |
| --- | --- | --- |
| DeleteApplication | `elasticloadbalancing:DeleteListener`<br>`elasticloadbalancing:DeleteTargetGroup`<br>`logs:DeleteLogDelivery` | Aplicativo |
| DeleteApplicationFromEnvironment | `elasticloadbalancing:DeleteListener`<br>`elasticloadbalancing:DeleteTargetGroup` | Aplicativo<br>Ambiente |
| DeleteEnvironment | `elasticloadbalancing:DeleteLoadBalancer` | Ambiente |
| GetApplication | | Aplicativo |
| GetApplicationVersion | | Aplicativo |
| GetBatchJobExecution | | Aplicativo |
| GetDataSetDetails | | Aplicativo |
| GetDataSetImportTask | | Aplicativo |
| GetDeployment | | Aplicativo |
| GetEnvironment | | Ambiente |
| ListApplications | | * |
| ListApplicationVersions | | * |

| AWS Operações de API de modernização de mainframe | Permissões obrigatórias (ações de API) | Recursos |
| --- | --- | --- |
| ListBatchJobDefinitions | | * |
| ListBatchJobExecutions | | * |
| ListDataSetImportHistory | | * |
| ListDataSets | | * |
| ListDeployments | | * |
| ListEngineVersions | | * |
| ListEnvironments | | * |
| ListTagsForResource | | * |
| StartApplication | | Aplicativo |
| StartBatchJob | | Aplicativo |
| StopApplication | | Aplicativo |
| TagResource | | * |
| UntagResource | | * |
| UpdateApplication | `s3:GetObject`<br>`s3:ListBucket` | Aplicativo |
| UpdateEnvironment | | Ambiente |

## Chaves de condição de política para a AWS modernização do mainframe

| | |
|---|---|
| Compatível com chaves de condição de política específicas do serviço | Sim |

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento `Condition` (ou bloco de `Condition`) permite que você especifique condições nas quais uma instrução está em vigor. O elemento `Condition` é opcional. É possível criar expressões condicionais que usam atendentes de condição, como “igual a” ou “menor que”, para fazer a condição da política corresponder aos valores na solicitação.

Se você especificar vários elementos `Condition` em uma instrução ou várias chaves em um único elemento `Condition`, a AWS os avaliará usando uma operação lógica `AND`. Se você especificar vários valores para uma única chave de condição, AWS avalia a condição usando uma `OR` operação lógica. Todas as condições devem ser atendidas para que as permissões da instrução sejam concedidas.

Você também pode usar variáveis de espaço reservado ao especificar as condições. Por exemplo, é possível conceder a um usuário do IAM permissão para acessar um recurso somente se ele estiver marcado com seu nome de usuário do IAM. Para obter mais informações, consulte Elementos de política do IAM: variáveis e tags no Guia do usuário do IAM.

AWS suporta chaves de condição globais e chaves de condição específicas do serviço. Para ver todas as chaves de condição AWS globais, consulte as chaves de contexto de condição AWS global no Guia do usuário do IAM.

As seguintes chaves de condição são específicas para a modernização AWS do mainframe

```
m2:EngineType
            m2:InstanceType
```

Para ver uma lista das chaves de condição de modernização de AWS mainframe, consulte Chaves de condição para modernização de AWS mainframe na Referência de autorização de serviço. Para saber com quais ações e recursos você pode usar uma chave de condição, consulte Ações definidas pela modernização do AWS mainframe.

Para ver exemplos de políticas baseadas em identidade de modernização de AWS mainframe, consulte. Exemplos de políticas baseadas em identidade para AWS modernização de mainframe

## Listas de controle de acesso (ACLs) no modernização do AWS Mainframe Modernization

| | |
|---|---|
| Oferece suporte a ACLs | Não |

As listas de controle de acesso (ACLs) controlam quais entidades principais (membros, usuários ou perfis da conta) têm permissões para acessar um recurso. As ACLs são semelhantes às políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

## Controle de acesso baseado em atributos (ABAC) com modernização do mainframe AWS

| | |
|---|---|
| Oferece suporte a ABAC (tags em políticas) | Sim |

O controle de acesso por atributo (ABAC) é uma estratégia de autorização que define permissões com base em atributos. Em AWS, esses atributos são chamados de tags. Você pode anexar tags a entidades do IAM (usuários ou funções) e a vários AWS recursos. A marcação de entidades e recursos é a primeira etapa do ABAC. Em seguida, você cria políticas de ABAC para permitir operações quando a tag da entidade principal corresponder à tag do recurso que ela está tentando acessar.

O ABAC é útil em ambientes que estão crescendo rapidamente e ajuda em situações em que o gerenciamento de políticas se torna um problema.

Para controlar o acesso baseado em tags, forneça informações sobre as tags no elemento de condição de uma política usando as `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* ou `aws:TagKeys` chaves de condição.

Se um serviço oferecer suporte às três chaves de condição para cada tipo de recurso, o valor será Sim para o serviço. Se um serviço oferecer suporte às três chaves de condição somente para alguns tipos de recursos, o valor será Parcial.

Para obter mais informações sobre o ABAC, consulte O que é ABAC? no Guia do usuário do IAM. Para visualizar um tutorial com etapas para configurar o ABAC, consulte Usar controle de acesso baseado em atributos (ABAC) no Guia do usuário do IAM.

## Usando credenciais temporárias com a modernização do AWS mainframe

| | |
|---|---|
| Oferece suporte a credenciais temporárias | Sim |

Alguns Serviços da AWS não funcionam quando você faz login usando credenciais temporárias. Para obter informações adicionais, incluindo quais Serviços da AWS funcionam com credenciais temporárias, consulte Serviços da AWS “Trabalhe com o IAM” no Guia do usuário do IAM.

Você está usando credenciais temporárias se fizer login AWS Management Console usando qualquer método, exceto um nome de usuário e senha. Por exemplo, quando você acessa AWS usando o link de login único (SSO) da sua empresa, esse processo cria automaticamente credenciais temporárias. Você também cria automaticamente credenciais temporárias quando faz login no console como usuário e, em seguida, alterna perfis. Para obter mais informações sobre como alternar perfis, consulte Alternar para uma função (console) no Guia do usuário do IAM.

Você pode criar manualmente credenciais temporárias usando a AWS API AWS CLI ou. Em seguida, você pode usar essas credenciais temporárias para acessar AWS. AWS recomenda que você gere credenciais temporárias dinamicamente em vez de usar chaves de acesso de longo prazo. Para obter mais informações, consulte Credenciais de segurança temporárias no IAM.

## Sessões de acesso direto para modernização AWS do mainframe

| | |
|---|---|
| Suporte para o recurso Encaminhamento de sessões de acesso (FAS) | Sim |

Quando você usa um usuário ou uma função do IAM para realizar ações AWS, você é considerado um principal. Ao usar alguns serviços, você pode executar uma ação que inicia outra ação em um serviço diferente. O FAS usa as permissões do diretor chamando um AWS service (Serviço da AWS), combinadas com a solicitação AWS service (Serviço da AWS) para fazer solicitações aos serviços posteriores. As solicitações do FAS são feitas somente quando um serviço recebe uma solicitação que requer interações com outros Serviços da AWS ou com recursos para ser concluída.

Nesse caso, você precisa ter permissões para executar ambas as ações. Para obter detalhes da política ao fazer solicitações de FAS, consulte Encaminhar sessões de acesso.

> **Important**
>
> Esses tokens dão à Modernização do AWS Mainframe acesso aos dados do cliente sem seu acordo explícito; por exemplo, a Modernização do AWS Mainframe implanta artefatos de aplicativos com dados comerciais associados de um bucket do Amazon S3 sem obter permissão explícita do cliente. Talvez seja necessário atualizar qualquer documentação de conformidade de acordo.

## Funções de serviço para AWS Mainframe Modernization

| | |
|---|---|
| Oferece suporte a perfis de serviço | Sim |

O perfil de serviço é um perfil do IAM que um serviço assume para realizar ações em seu nome. Um administrador do IAM pode criar, modificar e excluir um perfil de serviço do IAM. Para obter mais informações, consulte Criar um perfil para delegar permissões a um AWS service (Serviço da AWS) no Guia do usuário do IAM.

AWS A modernização do mainframe oferece suporte a funções de serviço para ganchos de atividades (transações, atrasos ou conclusão de tarefas, etc.).

> **Warning**
>
> Alterar as permissões de uma função de serviço pode interromper a funcionalidade de modernização do AWS mainframe. Edite as funções de serviço somente quando a modernização do AWS mainframe fornecer orientação para fazer isso.

### Escolha de uma função do IAM na modernização AWS do mainframe

Se você já criou uma função do IAM que seus aplicativos executados no Amazon EC2 podem assumir, você pode escolher essa função ao criar um modelo de execução ou uma configuração de execução. AWS A modernização do mainframe fornece uma lista de funções para você escolher. Ao criar essas funções, é importante associar políticas do IAM de privilégio mínimo que restrinjam o

acesso às chamadas de API específicas necessárias para a aplicação. Para obter mais informações, consulte Função do IAM para aplicativos que são executados em instâncias do Amazon EC2 no Guia do usuário do Amazon EC2 Auto Scaling.

### Funções vinculadas a serviços para AWS modernização do mainframe

| | |
|---|---|
| Oferece suporte a funções vinculadas ao serviço | Sim |

Uma função vinculada ao serviço é um tipo de função de serviço vinculada a um. AWS service (Serviço da AWS) O serviço pode assumir a função de executar uma ação em seu nome. As funções vinculadas ao serviço aparecem em você Conta da AWS e são de propriedade do serviço. Um administrador do IAM pode visualizar, mas não pode editar as permissões para perfis vinculados ao serviço.

Para obter detalhes sobre como criar ou gerenciar funções vinculadas ao serviço de modernização de AWS mainframe, consulte. Usar funções vinculadas ao serviço do

Para obter detalhes sobre como criar ou gerenciar funções vinculadas a serviços, consulte Serviços do AWS que funcionam com o IAM. Encontre um serviço na tabela que inclua um `Yes` na coluna Função vinculada ao serviço. Escolha o link Sim para visualizar a documentação do perfil vinculado a esse serviço.

## Exemplos de políticas baseadas em identidade para AWS modernização de mainframe

Por padrão, usuários e funções não têm permissão para criar ou modificar recursos de modernização de AWS mainframe. Eles também não podem realizar tarefas usando a AWS API AWS Management Console, AWS Command Line Interface (AWS CLI) ou. Para conceder aos usuários permissões para executar ações nos recursos de que eles precisam, um administrador do IAM pode criar políticas do IAM. O administrador pode então adicionar as políticas do IAM aos perfis, e os usuários podem assumir os perfis.

Para saber como criar uma política baseada em identidade do IAM usando esses exemplos de documento de política JSON, consulte Criação de políticas do IAM no Guia do Usuário do IAM.

Para obter detalhes sobre ações e tipos de recursos definidos pela modernização do AWS mainframe, incluindo o formato dos ARNs para cada um dos tipos de recursos, consulte Ações,

recursos e chaves de condição para a modernização do AWS mainframe na Referência de autorização de serviço.

Tópicos

- Práticas recomendadas de políticas
- Usando o console de modernização AWS de mainframe
- Permitir que os usuários visualizem suas próprias permissões

## Práticas recomendadas de políticas

As políticas baseadas em identidade determinam se alguém pode criar, acessar ou excluir recursos de modernização de AWS mainframe em sua conta. Essas ações podem incorrer em custos para a Conta da AWS. Ao criar ou editar políticas baseadas em identidade, siga estas diretrizes e recomendações:

- Comece com as políticas AWS gerenciadas e avance para as permissões de privilégios mínimos — Para começar a conceder permissões aos seus usuários e cargas de trabalho, use as políticas AWS gerenciadas que concedem permissões para muitos casos de uso comuns. Eles estão disponíveis no seu Conta da AWS. Recomendamos que você reduza ainda mais as permissões definindo políticas gerenciadas pelo AWS cliente que sejam específicas para seus casos de uso. Para obter mais informações, consulte Políticas gerenciadas pela AWS ou Políticas gerenciadas pela AWS para perfis de trabalho no Guia do usuário do IAM.
- Aplique permissões de privilégio mínimo: ao definir permissões com as políticas do IAM, conceda apenas as permissões necessárias para executar uma tarefa. Você faz isso definindo as ações que podem ser executadas em recursos específicos sob condições específicas, também conhecidas como permissões de privilégio mínimo. Para obter mais informações sobre como usar o IAM para aplicar permissões, consulte Políticas e permissões no IAM no Guia do usuário do IAM.
- Use condições nas políticas do IAM para restringir ainda mais o acesso: você pode adicionar uma condição às políticas para limitar o acesso a ações e recursos. Por exemplo, você pode escrever uma condição de política para especificar que todas as solicitações devem ser enviadas usando SSL. Você também pode usar condições para conceder acesso às ações de serviço se elas forem usadas por meio de uma ação específica AWS service (Serviço da AWS), como AWS CloudFormation. Para obter mais informações, consulte Elementos de política JSON do IAM: condição no Guia do usuário do IAM.
- Use o IAM Access Analyzer para validar suas políticas do IAM a fim de garantir permissões seguras e funcionais: o IAM Access Analyzer valida as políticas novas e existentes para que elas

sigam a linguagem de política do IAM (JSON) e as práticas recomendadas do IAM. O IAM Access Analyzer oferece mais de 100 verificações de política e recomendações acionáveis para ajudar você a criar políticas seguras e funcionais. Para obter mais informações, consulte Validação de políticas do IAM Access Analyzer no Guia do usuário do IAM.

- Exigir autenticação multifator (MFA) — Se você tiver um cenário que exija usuários do IAM ou um usuário root, ative Conta da AWS a MFA para obter segurança adicional. Para exigir a MFA quando as operações de API forem chamadas, adicione condições de MFA às suas políticas. Para obter mais informações, consulte Configuração de acesso à API protegido por MFA no Guia do usuário do IAM.

Para obter mais informações sobre as práticas recomendadas do IAM, consulte Práticas recomendadas de segurança no IAM no Guia do usuário do IAM.

## Usando o console de modernização AWS de mainframe

Para acessar o console de modernização do AWS mainframe, você deve ter um conjunto mínimo de permissões. Essas permissões devem permitir que você liste e visualize detalhes sobre os recursos de modernização do AWS mainframe em seu. Conta da AWS Se você criar uma política baseada em identidade que seja mais restritiva do que as permissões mínimas necessárias, o console não funcionará como pretendido para entidades (usuários ou perfis) com essa política.

Você não precisa permitir permissões mínimas do console para usuários que estão fazendo chamadas somente para a API AWS CLI ou para a AWS API. Em vez disso, permita o acesso somente a ações que correspondam a operação de API que estiverem tentando executar.

Para garantir que os usuários e as funções ainda possam usar o console de modernização do AWS mainframe, anexe também a política `ReadOnly` AWS gerenciada `ConsoleAccess` ou a modernização do AWS mainframe às entidades. Para obter mais informações, consulte Adicionando Permissões a um Usuário no Guia do Usuário do IAM.

## Permitir que os usuários visualizem suas próprias permissões

Este exemplo mostra como você pode criar uma política que permite que os usuários do IAM visualizem as políticas gerenciadas e em linha anexadas a sua identidade de usuário. Essa política inclui permissões para concluir essa ação no console ou programaticamente usando a API AWS CLI ou 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": "*"
        }
    ]
}
```

## Solução de problemas de identidade e AWS acesso à modernização do mainframe

Use as informações a seguir para ajudá-lo a diagnosticar e corrigir problemas comuns que você pode encontrar ao trabalhar com a modernização do AWS mainframe e o IAM.

Tópicos

- Não estou autorizado a realizar iam: PassRole

- Quero permitir que pessoas de fora da minha tenham acesso Conta da AWS aos meus recursos de modernização de AWS mainframe

## Não estou autorizado a realizar iam: PassRole

Se você receber um erro informando que não está autorizado a realizar a `iam:PassRole` ação, suas políticas devem ser atualizadas para permitir que você passe uma função para a modernização do AWS mainframe.

Alguns Serviços da AWS permitem que você passe uma função existente para esse serviço em vez de criar uma nova função de serviço ou uma função vinculada ao serviço. Para fazer isso, é preciso ter permissões para passar o perfil para o serviço.

O exemplo de erro a seguir ocorre quando um usuário IAM chamado `marymajor` tenta usar o console para executar uma ação no AWS Mainframe Modernization. No entanto, a ação exige que o serviço tenha permissões concedidas por um perfil de serviço. Mary não tem permissões para passar a função para o serviço.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Nesse caso, as políticas de Mary devem ser atualizadas para permitir que ela realize a ação `iam:PassRole`.

Se precisar de ajuda, entre em contato com seu AWS administrador. Seu administrador é a pessoa que forneceu suas credenciais de login.

## Quero permitir que pessoas de fora da minha tenham acesso Conta da AWS aos meus recursos de modernização de AWS mainframe

Você pode criar um perfil que os usuários de outras contas ou pessoas fora da organização podem usar para acessar seus recursos. Você pode especificar quem é confiável para assumir o perfil. Para serviços que oferecem suporte a políticas baseadas em recursos ou listas de controle de acesso (ACLs), você pode usar essas políticas para conceder às pessoas acesso aos seus recursos.

Para saber mais, consulte:

- Para saber se a modernização do AWS mainframe é compatível com esses recursos, consulte. Como a modernização AWS do mainframe funciona com o IAM

- Para saber como fornecer acesso aos seus recursos em todos os Contas da AWS que você possui, consulte Como fornecer acesso a um usuário do IAM em outro Conta da AWS que você possui no Guia do usuário do IAM.
- Para saber como fornecer acesso aos seus recursos a terceiros Contas da AWS, consulte Como fornecer acesso Contas da AWS a terceiros no Guia do usuário do IAM.
- Para saber como conceder acesso por meio da federação de identidades, consulte Conceder acesso a usuários autenticados externamente (federação de identidades) no Guia do usuário do IAM.
- Para saber a diferença entre usar perfis e políticas baseadas em recursos para acesso entre contas, consulte Como os perfis do IAM diferem de políticas baseadas em recursos no Guia do usuário do IAM.

## Usar funções vinculadas ao serviço do

O AWS Mainframe Modernization usa perfis vinculados ao serviço do AWS Identity and Access Management (IAM). Um perfil vinculado ao serviço é um tipo especial de perfil do IAM vinculado diretamente ao Amazon EMR. Os perfis vinculados a serviço são predefinidos pelo S3 no Outposts e incluem todas as permissões que o serviço requer para chamar outros serviços da AWS em seu nome.

Uma função vinculada ao serviço facilita a configuração do DAX porque você não precisa adicionar as permissões necessárias manualmente. O define as permissões das funções vinculadas ao serviço e, exceto se definido de outra forma, somente o pode assumir suas funções. As permissões definidas incluem a política de confiança e a política de permissões, que não pode ser anexada a nenhuma outra entidade do IAM.

Um perfil vinculado ao serviço poderá ser excluído somente após excluir seus atributos relacionados. Isso protege seus recursos do , pois você não pode remover por engano as permissões para acessar os recursos.

Para obter informações sobre outros serviços compatíveis com perfis vinculados aos serviços, consulte Serviços da AWS que funcionam com o IAM e procure os serviços que apresentam Sim na coluna Funções vinculadas aos serviços. Escolha Sim com um link para visualizar a documentação da função vinculada a esse serviço.

## Permissões de função vinculada ao serviço para a modernização do mainframe

A modernização do mainframe usa a função vinculada ao serviço chamada AWSServiceRoleForAWSM2 — configure a rede para se conectar à sua VPC e acessar recursos como sistemas de arquivos.

A função vinculada ao serviço AWSServiceRoleForAWSM2 confia nos seguintes serviços para assumir a função:

- `m2.amazonaws.com`

A política de permissões de função denominada AWSM2ServicePolicy permite que o Mainframe Modernization conclua as seguintes ações nos recursos especificados:

- Crie, exclua, descreva e anexe permissões às interfaces de rede do Amazon EC2 para o ambiente de modernização do mainframe a fim de estabelecer conectividade com a VPC do cliente.
- Registre ou cancele o registro de entradas do Elastic Load Balancing, que é como os clientes se conectam ao ambiente de modernização do mainframe.
- Descreva o sistema de arquivos Amazon EFS ou Amazon FSx, se usado.
- Emita métricas para o CloudWatch do cliente a partir do ambiente de execução.

```
{
 "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"
     ]
    }
   }
  }
 ]
}
```

Você deve configurar permissões para que uma entidade do IAM (por exemplo, um usuário, grupo ou perfil) crie, edite ou exclua um perfil vinculado ao serviço. Para obter mais informações, consulte Permissões de função vinculada a serviços no Guia do usuário do IAM.

## Criar uma função vinculada a serviço para a modernização do mainframe

Não é necessário criar manualmente um perfil vinculado ao serviço. Quando você cria um ambiente de tempo de execução no AWS Management Console, na ou na AWS API daAWS CLI, o Mainframe Modernization cria o perfil vinculado ao serviço para você.

Se excluir esse perfil vinculado ao serviço e precisar criá-lo novamente, você poderá usar esse mesmo processo para recriar o perfil em sua conta. Quando você cria um ambiente de tempo de execução, a Modernização do Mainframe cria a função vinculada ao serviço para você novamente.

## Editar uma função vinculada ao serviço para a modernização do mainframe

O não permite editar a função vinculada ao serviço AWSServiceRoleForAWSLicenseManagerMasterAccountRole. Depois que criar um perfil vinculado ao serviço, você não poderá alterar o nome do perfil, pois várias entidades podem fazer referência a ela. No entanto, será possível editar a descrição do perfil usando o IAM. Para obter mais informações, consulte Editar uma função vinculada a serviço no Guia do usuário do IAM.

## Excluir uma função vinculada ao serviço para a modernização do mainframe

Se você não precisar mais usar um atributo ou serviço que requer um perfil vinculado ao serviço, é recomendável excluí-lo. Dessa forma, você não tem uma entidade não utilizada que não seja monitorada ativamente ou mantida. No entanto, você deve limpar os recursos de sua função vinculada ao serviço antes de excluí-la manualmente.

> **Note**
>
> Se o serviço Mainframe Modernization estiver usando a função quando você tentar excluir os recursos, a exclusão poderá falhar. Se isso acontecer, espere alguns minutos e tente a operação novamente.

Para excluir os recursos de Mainframe Modernization usados pelo AWSServiceRoleForAWSM2

- Exclua os ambientes de tempo de execução na modernização do mainframe. Certifique-se de excluir aplicativos de um ambiente antes de excluir o próprio ambiente.

Como excluir manualmente o perfil vinculado ao serviço usando o IAM

Use o console do IAM, a AWS CLI ou a AWS API para excluir o perfil vinculado ao serviço AWSServiceRoleForWord. Para obter mais informações, consulte Excluir uma função vinculada ao serviço no Guia do usuário do IAM.

### Regiões compatíveis com perfis vinculados ao serviço de modernização de mainframe

O oferece suporte a perfis vinculados ao serviço em todas as regiões em que o serviço do está disponível. Para obter mais informações, consulte Regiões e endpoints da AWS.

## Validação de conformidade do AWS Mainframe Modernization

Auditores externos avaliam a segurança e a conformidade dos serviços da AWS como parte de vários programas de conformidade da AWS. Isso inclui SOC, PCI, FedRAMP, HIPAA e outros.

Para obter uma lista de serviços da AWS no escopo de programas de conformidade específicos, consulte Serviços da AWS no escopo pelo programa de conformidade. Para obter informações gerais, consulte Programas de conformidade da AWS.

É possível fazer download de relatórios de auditoria de terceiros usando o AWS Artifact. Para obter mais informações, consulte Fazer download de relatórios no AWS Artifact.

Sua responsabilidade de conformidade ao usar o AWS Mainframe Modernization é determinada pela sensibilidade dos seus dados, pelos objetivos de conformidade da sua empresa e pelas leis e normas aplicáveis.O AWS fornece os seguintes recursos para ajudar na conformidade:

- Guias de início rápido de segurança e compatibilidade: estes guias de implantação abordam as considerações de arquitetura e fornecem etapas para implantação de ambientes de linha de base focados em compatibilidade e segurança na AWS.
- Whitepaper Architecting for HIPAA Security and Compliance: este whitepaper descreve como as empresas podem usar a AWS para criar aplicações em conformidade com a HIPAA.
- Recursos de conformidade da AWS: essa coleção de manuais e guias pode ser aplicada a seu setor e local.
- Avaliar recursos com regras no AWS Config Guia do desenvolvedor: AWS Config; avalia como suas configurações de recursos estão em conformidade com práticas internas, diretrizes do setor e regulamentos.
- AWS Security Hub: esse serviço da AWS fornece uma visão abrangente do estado de sua segurança na AWS que ajuda você a conferir sua conformidade com padrões e práticas recomendadas de segurança do setor.

# Resiliência no AWS Mainframe Modernization

A infraestrutura global da AWS é criada com base em regiões e zonas de disponibilidade da AWS. As regiões fornecem várias zonas de disponibilidade separadas e isoladas fisicamente, que são conectadas com baixa latência, alta throughput e redes altamente redundantes. Com as zonas de disponibilidade, é possível projetar e operar aplicações e bancos de dados que automaticamente executam o failover entre as zonas sem interrupção. As zonas de disponibilidade são mais altamente disponíveis, tolerantes a falhas e escaláveis que uma ou várias infraestruturas de data center tradicionais.

Para obter mais informações sobre regiões e zonas de disponibilidade da AWS, consulte Infraestrutura global da AWS.

# Segurança da infraestrutura no AWS Mainframe Modernization

Por ser um serviço gerenciado, o AWS Mainframe Modernization é protegido pela segurança da rede global da AWS. Para obter informações sobre serviços de segurança da AWS e como a AWS protege a infraestrutura, consulte Segurança na Nuvem AWS. Para projetar seu ambiente da AWS usando as práticas recomendadas de segurança de infraestrutura, consulte Proteção de infraestrutura em Pilar segurança: AWS Well-Architected Framework.

Você usa as chamadas de API publicadas pela AWS para acessar a Mainframe Modernization por meio da rede. Os clientes devem oferecer suporte para:

- Transport Layer Security (TLS). Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Conjuntos de criptografia com perfect forward secrecy (PFS) como DHE (Ephemeral Diffie-Hellman) ou ECDHE (Ephemeral Elliptic Curve Diffie-Hellman). A maioria dos sistemas modernos, como Java 7 e versões posteriores, comporta esses modos.

Além disso, as solicitações devem ser assinadas usando um ID da chave de acesso e uma chave de acesso secreta associada a uma entidade principal do IAM. Ou você pode usar o AWS Security Token Service (AWS STS) para gerar credenciais de segurança temporárias para assinar solicitações.

# Acesse o AWS Mainframe Modernization usando um VPC endpoint de interface (AWS PrivateLink)

É possível usar o AWS PrivateLink para criar uma conectividade privada entre a sua VPC e o AWS Mainframe Modernization. Você pode acessar o Mainframe Modernization como se ele estivesse na sua VPC, sem o uso de um gateway de Internet, dispositivo NAT, conexão VPN ou uma conexão AWS Direct Connect. As instâncias em sua VPC não precisam de endereços IP públicos para acessar o Mainframe Modernization.

Você estabelece essa conectividade privada criando um endpoint de interface, desenvolvido pelo AWS PrivateLink. Criaremos um endpoint de interface de rede em cada sub-rede que você habilitar para o endpoint de interface. Essas são interfaces de rede gerenciadas pelo solicitante que servem como ponto de entrada para o tráfego destinado ao Mainframe Modernization.

Para obter mais informações, consulte Acessar os Serviços da AWS pelo AWS PrivateLink no Guia do AWS PrivateLink.

## Considerações sobre como modernizar um mainframe

Antes de configurar um endpoint de interface para o , analise as considerações no Guia do AWS PrivateLink.

A modernização do mainframe oferece suporte a chamadas para todas as ações de API da interface por meio do endpoint da interface.

## Criação de um endpoint de interface para modernização do mainframe

Você pode criar um endpoint de interface para "Mainframe Modernization" usando o console do Amazon VPC ou o AWS Command Line Interface (AWS CLI). Para obter mais informações, consulte Criar um endpoint de interface no Guia do usuário do AWS PrivateLink.

Crie um endpoint de interface para Mainframe Modernization usando o seguinte nome de serviço:

```
com.amazonaws.region.m2
```

Se você ativar o DNS privado para o ponto de extremidade da interface, poderá fazer solicitações de API para Mainframe Modernization usando seu nome DNS regional padrão. Por exemplo, `m2.us-east-1.amazonaws.com`.

# Criar uma política de endpoint para o endpoint da interface

Política de endpoint é um recurso do IAM que você pode anexar ao endpoint de interface. A política de endpoint padrão permite acesso total ao Mainframe Modernization por meio do endpoint de interface. Para controlar o acesso permitido ao Mainframe Modernization a partir de sua VPC, anexe uma política de endpoint personalizada ao endpoint da interface.

Uma política de endpoint especifica as seguintes informações:

- As entidades principais que podem executar ações (Contas da AWS, usuários e funções do IAM).
- As ações que podem ser executadas.
- Os recursos nos quais as ações podem ser executadas.

Para obter mais informações, consulte Controlar o Acesso a Serviços Usando Políticas de Endpoint no AWS PrivateLink Guia.

Exemplo: política de VPC endpoint para ações de modernização de mainframe

O exemplo a seguir refere-se a uma política de endpoint personalizada. Quando você anexa essa política ao seu endpoint de interface, ela concede acesso às ações do Mainframe Modernization listadas para todos os diretores em todos os recursos.

```
//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":"*"
      }
   ]
}
```

# Monitorando a AWS modernização do mainframe

O monitoramento é uma parte importante da manutenção da confiabilidade, disponibilidade e desempenho da modernização do AWS mainframe e de suas outras soluções da AWS. A AWS fornece as seguintes ferramentas de monitoramento para observar a modernização do AWS mainframe, relatar quando algo está errado e realizar ações automáticas quando apropriado:

- A Amazon CloudWatch monitora seus AWS recursos e os aplicativos em que você executa AWS em tempo real. É possível coletar e rastrear métricas, criar painéis personalizados e definir alarmes que o notificam ou que realizam ações quando uma métrica especificada atinge um limite definido. Por exemplo, você pode CloudWatch rastrear o uso da CPU ou outras métricas de suas instâncias do Amazon EC2 e iniciar automaticamente novas instâncias quando necessário. Para obter mais informações, consulte o Guia CloudWatch do usuário da Amazon.
- O Amazon CloudWatch Logs permite que você monitore, armazene e acesse seus arquivos de log a partir de instâncias do Amazon EC2 e de outras fontes. CloudTrail CloudWatch Os registros podem monitorar as informações nos arquivos de log e notificá-lo quando determinados limites forem atingidos. É possível também arquivar seus dados de log em armazenamento resiliente. Para obter mais informações, consulte o Guia do usuário do Amazon CloudWatch Logs.
- AWS CloudTrailcaptura chamadas de API e eventos relacionados feitos por ou em nome de sua AWS conta e entrega os arquivos de log para um bucket do Amazon S3 que você especificar. Você pode identificar quais usuários e contas ligaram AWS, o endereço IP de origem a partir do qual as chamadas foram feitas e quando elas ocorreram. Para obter mais informações, consulte o AWS CloudTrail Guia do Usuário.

## Monitorando a modernização AWS do mainframe com a Amazon CloudWatch

Você pode monitorar a modernização do AWS mainframe usando CloudWatch, que coleta dados brutos e os processa em métricas legíveis, quase em tempo real. Essas estatísticas são mantidas por 15 meses, de maneira que você possa acessar informações históricas e ter uma perspectiva melhor de como o aplicativo web ou o serviço está se saindo. Você também pode definir alarmes que observam determinados limites e enviam notificações ou realizam ações quando esses limites são atingidos. Para obter mais informações, consulte o Guia CloudWatch do usuário da Amazon.

As tabelas a seguir listam as métricas e dimensões da modernização AWS do mainframe. O namespace para essas métricas é `AWS/M2`.

## Mainframe Metrics

| Métrica | Descrição |
| --- | --- |
| CPUUtilization | A utilização da CPU das instâncias no ambiente.<br><br>Diimensão: EnvironmentID<br><br>Unidades: percentual<br><br>Estatísticas válidas: média, mínimo, máximo |
| InboundNetworkThroughput | Taxa de transferência de rede de entrada de instâncias no ambiente.<br><br>Diimensão: EnvironmentID<br><br>Unidades: bytes por segundo<br><br>Estatísticas válidas: média, mínimo, máximo |
| MemoryUtilization | A utilização da memória das instâncias no ambiente.<br><br>Diimensão: EnvironmentID<br><br>Unidades: percentual<br><br>Estatísticas válidas: média, mínimo, máximo |
| OutboundNetworkThroughput | Taxa de transferência de rede de saída das instâncias no ambiente.<br><br>Diimensão: EnvironmentID<br><br>Unidades: bytes por segundo |

| Métrica | Descrição |
| --- | --- |
| | Estatísticas válidas: média, mínimo, máximo |

## Métricas de aplicativo

| Métrica | Descrição |
| --- | --- |
| BatchJobCompletedCount | O número de trabalhos concluídos durante o intervalo de tempo.<br><br>Essa métrica está disponível para a Micro Focus e para o AWS Blu Age 3.7.0 e versões posteriores.<br><br>Dimensões: applicationId<br><br>Unidades: contagem<br><br>Estatística válida: soma |
| BatchJobFailedCount | O número de trabalhos com falha durante o intervalo de tempo.<br><br>Essa métrica está disponível para a Micro Focus e para o AWS Blu Age 3.7.0 e versões posteriores.<br><br>Dimensões: applicationId<br><br>Unidades: contagem<br><br>Estatística válida: soma |
| JvmMemoryFree | A quantidade de memória disponível que não está sendo usada atualmente pela Java Virtual Machine.<br><br>Essa métrica está disponível somente para o mecanismo de tempo de execução do AWS Blu |

| Métrica | Descrição |
| --- | --- |
| | Age. Ele está disponível para o AWS Blu Age 3.7.0 e versões posteriores.<br><br>Dimensões: applicationId<br><br>Unidades: bytes<br><br>Estatísticas válidas: média, mínimo, máximo |
| JvmMemoryMax | A quantidade máxima de memória permitida para a Java Virtual Machine.<br><br>Essa métrica está disponível somente para o mecanismo de tempo de execução do AWS Blu Age. Ele está disponível para o AWS Blu Age 3.7.0 e versões posteriores.<br><br>Dimensões: applicationId<br><br>Unidades: bytes<br><br>Estatísticas válidas: média, mínimo, máximo |
| JvmMemoryUsed | A quantidade de memória usada ativamente pela Máquina Virtual Java.<br><br>Essa métrica está disponível somente para o mecanismo de tempo de execução do AWS Blu Age. Ele está disponível para o AWS Blu Age 3.7.0 e versões posteriores.<br><br>Dimensões: applicationId<br><br>Unidades: bytes<br><br>Estatísticas válidas: média, mínimo, máximo |

| Métrica | Descrição |
| --- | --- |
| ProcessesActiveCount | O número ativo de processos de execução simultânea de serviços que estão processando solicitações.<br><br>Essa métrica só está disponível para o mecanismo de execução da Micro Focus.<br><br>Dimensões: applicationId<br><br>Unidades: contagem<br><br>Estatística válida: soma |
| SessionCount | O número de sessões HTTP para o aplicativo.<br><br>Essa métrica está disponível somente para o mecanismo de tempo de execução do AWS Blu Age. Ele está disponível para o AWS Blu Age 3.7.0 e versões posteriores.<br><br>Dimensões: applicationId<br><br>Unidades: contagem<br><br>Estatísticas válidas: média, mínimo, máximo |
| SharedMemoryFree | A memória disponível para que o servidor corporativo armazene todas as informaçõ es necessárias para executar transações e trabalhos.<br><br>Essa métrica só está disponível para o mecanismo de execução da Micro Focus.<br><br>Dimensões: applicationId<br><br>Unidades: contagem<br><br>Estatísticas válidas: média, mínimo, máximo |

| Métrica | Descrição |
| --- | --- |
| ThreadActiveCount | O número de threads do mecanismo que estão processando solicitações.<br><br>Essa métrica está disponível somente para o mecanismo de tempo de execução do AWS Blu Age. Ele está disponível para o AWS Blu Age 3.7.0 e versões posteriores.<br><br>Dimensões: applicationId<br><br>Unidades: contagem<br><br>Estatísticas válidas: média, mínimo, máximo |
| TransactionCompletedCount | O número de transações confirmadas durante o intervalo de tempo.<br><br>Essa métrica está disponível para a Micro Focus e para o AWS Blu Age 3.7.0 e versões posteriores.<br><br>Dimensões: applicationId<br><br>Unidades: contagem<br><br>Estatística válida: soma |
| TransactionFailedCount | O número de transações com falha durante o intervalo de tempo.<br><br>Essa métrica está disponível para a Micro Focus e para o AWS Blu Age 3.7.0 e versões posteriores.<br><br>Dimensões: applicationId<br><br>Unidades: contagem<br><br>Estatística válida: soma |

| Métrica | Descrição |
| --- | --- |
| TransactionResponseTime | A quantidade de tempo desde o momento em que um usuário envia uma solicitação até o momento em que o aplicativo indica que a solicitação foi concluída.<br><br>Essa métrica está disponível para a Micro Focus e para o AWS Blu Age 3.7.0 e versões posteriores.<br><br>Dimensões: applicationId<br><br>Unidade: milissegundos<br><br>Estatísticas válidas: média, mínimo, máximo |

### Dimensões

| Dimensão | Descrição |
| --- | --- |
| ApplicationId | Essa dimensão filtra a métrica para o aplicativo identificado por ID. |
| EnvironmentId | Essa dimensão filtra a métrica para o ambiente identificado por ID. |

## Registro de chamadas da API do AWS Mainframe Modernization usando o AWS CloudTrail

O AWS Mainframe Modernization está integrado ao AWS CloudTrail, um serviço que fornece um registro das ações realizadas por um usuário, uma função ou um serviço do AWS no AWS Mainframe Modernization. O CloudTrail captura todas as chamadas de API para o AWS Mainframe Modernization como eventos. As chamadas capturadas incluem chamadas do console do AWS Mainframe Modernization e chamadas de código para as operações da API do AWS Mainframe Modernization. Se você criar uma trilha, poderá ativar a entrega contínua de eventos do CloudTrail

em um bucket do Amazon S3, incluindo eventos para o AWS Mainframe Modernization. Se você não configurar uma trilha, ainda poderá visualizar os eventos mais recentes no console do CloudTrail em Histórico de eventos. Usando as informações coletadas pelo CloudTrail, você pode determinar a solicitação que foi feita ao AWS Mainframe Modernization, o endereço IP a partir do qual a solicitação foi feita, quem fez a solicitação, quando foi feita e outros detalhes.

Para saber mais sobre o CloudTrail, consulte o Guia do usuário do AWS CloudTrail.

## Informações do AWS Mainframe Modernization no CloudTrail

O CloudTrail é habilitado em sua conta AWS quando ela é criada. Quando ocorre uma atividade no AWS Mainframe Modernization, essa atividade é registrada em um evento do CloudTrail junto com outros eventos de serviço do AWS no histórico de eventos. Você pode visualizar, pesquisar e baixar eventos recentes em sua conta AWS. Para obter mais informações, consulte Visualização de eventos com o histórico de eventos do CloudTrail.

Para obter um registro contínuo de eventos na sua conta do AWS, incluindo eventos do AWS Mainframe Modernization, crie uma trilha. Uma trilha permite que o CloudTrail entregue arquivos de log a um bucket Amazon S3. Por padrão, quando você cria uma trilha no console, ela é aplicada a todas as regiões da AWS. A trilha registra eventos de todas as regiões na partição da AWS e entrega os arquivos de registro no bucket do Amazon S3 que você especificou Além disso, é possível configurar outros serviços da AWS para analisar mais ainda e agir com base nos dados de eventos coletados nos logs do CloudTrail. Para mais informações, consulte:

- Visão geral da criação de uma trilha
- Serviços e integrações compatíveis com o CloudTrail
- Configuração notificações do Amazon SNS para o CloudTrail
- Receber arquivos de log do CloudTrail de várias regiões
- Receber arquivos de log do CloudTrail de várias contas

Todas as ações do AWS Mainframe Modernization são registradas pelo CloudTrail e estão documentadas na Referência da API do AWS Mainframe Modernization. Por exemplo, as chamadas às ações `CreateApplication`, `CreateEnvironment` e `CreateDeployment` geram entradas nos arquivos de log do CloudTrail.

Cada entrada de log ou evento contém informações sobre quem gerou a solicitação. As informações de identidade ajudam a determinar:

- Se a solicitação foi feita com credenciais de usuário raiz ou credenciais de usuário.
- Se a solicitação foi feita com credenciais de segurança temporárias de um perfil ou de um usuário federado.
- Se a solicitação foi feita por outro serviço da AWS.

Para obter mais informações, consulte Elemento de identidade do usuário do CloudTrail.

## Entendendo as entradas do arquivo de log do AWS Mainframe Modernization

Uma trilha é uma configuração que permite a entrega de eventos como arquivos de log a um bucket do Amazon S3 especificado. Os arquivos de log CloudTrail contêm uma ou mais entradas de log. Um evento representa uma única solicitação de qualquer fonte, e inclui informações sobre a ação solicitada, data e hora da ação, parâmetros de solicitação e assim por diante. Os arquivos de log do CloudTrail não são um rastreamento de pilha ordenada de chamadas de API pública, portanto não são exibidos em uma ordem específica.

O exemplo a seguir mostra uma entrada de log CloudTrail que demonstra a `CreateApplication` ação.


```
{
    "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"
}
```

# Solução de problemas

Use as informações desta seção para ajudá-lo a solucionar erros comuns nos aplicativos e ambientes de tempo de execução do AWS Mainframe Modernization usando os mecanismos AWS Blu Age e Micro Focus.

Tópicos



# Erro: tempo limite ao esperar que o nome do conjunto de dados seja desbloqueado

- Motor: AWS Blu Age
- Componente: Blusam

Se você ver esse erro nos CloudWatch registros da Amazon para um aplicativo de modernização de AWS mainframe usando o mecanismo AWS Blu Age e executado em um ambiente com o padrão de alta disponibilidade, isso indica que outro aplicativo está bloqueando um conjunto de dados compartilhado. Normalmente, essa situação ocorre se o outro aplicativo falhar ou falhar e não liberar o bloqueio.

## Como esse erro ocorre

O aplicativo `example-app-1` tenta bloquear um registro `example-record-1` para uma operação de gravação. Essa operação cria um bloqueio no conjunto de dados `example-dataset-1`, que possui `example-record-1`, e um bloqueio em `example-record-1` si mesmo. Agora, outro aplicativo `example-app-2`, tenta bloquear o mesmo registro `example-record-1`. O conjunto de dados e o registro já estão bloqueados, então `example-app-2` aguarda a liberação do bloqueio. Se `example-app-1` falhar, o bloqueio retido no conjunto de dados `example-dataset-1` ainda existe, o que faz com que `example-app-2` a tentativa de gravação seja cancelada e gere uma exceção de tempo limite. Essa situação de impasse impede que todos os aplicativos alcancem `example-dataset-1`.

## Como você sabe se essa é a sua situação?

Procure um aplicativo com falha e verifique se ele usa o mesmo conjunto de dados mencionado na mensagem de erro. Verifique se o aplicativo está sendo executado em um ambiente de execução com o padrão de alta disponibilidade. O aplicativo que gerou a exceção de tempo limite não pode continuar e exibirá o status `Failed`.

## O que você pode fazer?

Para resolver a situação imediatamente, você pode forçar a liberação da trava. Para evitar que uma situação semelhante ocorra no futuro, você pode configurar dois parâmetros que controlam o mecanismo de reparo automático Blusam.

## Forçar a liberação da trava

O gerenciador de bloqueio Blusam usa o Amazon ElastiCache for Redis para fornecer bloqueios compartilhados entre aplicativos. Para liberar bloqueios ElastiCache, use o utilitário CLI do Redis. Não é possível excluir uma bloqueios de registro individual. Você deve remover todos os bloqueios do conjunto de dados proprietário. Execute as etapas a seguir:

1. Conecte-se ao seu ElastiCache usando o seguinte comando:

   ```
   redis-cli -h hostname -p port
   ```

   Você pode encontrar seus detalhes ElastiCache no ElastiCache console em https://console.aws.amazon.com/elasticache/.
2. Insira a senha.
3. Digite o comando que você deseja executar, da seguinte forma:

| Command | Finalidade |
| --- | --- |
| KEYS * | Obtenha todas as chaves existentes. |
| CHAVES * *YOUR_DATASET_NAME* | Obtenha uma chave de bloqueio do conjunto de dados. |
| EXCLUA *THE_RETURNED_KEY* | Exclua um bloqueio de conjunto de dados. |

| Command | Finalidade |
| --- | --- |
| FLUSHDB | Limpe todo o Redis.<br><br>**Warning**<br>Todos os dados no cache do Redis serão perdidos. Se o Redis for usado para outras finalidades, como o tratamento de sessões http, talvez você não queira usar FLUSHDB. |

## Configure o mecanismo de reparo automático Blusam

O gerenciador de bloqueios Blusam inclui um mecanismo de reparo automático para evitar bloqueios em conjuntos de dados ou registros. Você pode ajustar os seguintes parâmetros na definição do aplicativo (`application-main.yml`) para configurar o mecanismo de reparo automático:

- `locksDeadTime`: refere-se ao tempo máximo que um aplicativo pode manter um bloqueio. Quando esse tempo passa, o bloqueio é declarado expirado e liberado imediatamente. O `locksDeadTime` valor está em milissegundos e o valor padrão é 1000.
- `locksCheck`: define a estratégia do gerenciador de bloqueios Blusam para verificar bloqueios. Todos os bloqueios Blusam ElastiCache têm data e hora de validade. O valor do `locksCheck` parâmetro determina se os bloqueios expirados são removidos.
  - `off`: nenhuma verificação é executada em nenhum momento. Podem ocorrer impasses. (Não recomendado)
  - `reboot`: as verificações são executadas quando uma instância do aplicativo do AWS Mainframe Modernization em execução em um ambiente de tempo de execução do AWS Mainframe Modernization é iniciada ou reinicializada. Todos os bloqueios expirados são liberados imediatamente. (Padrão)
  - `timeout`: as verificações são executadas quando uma instância do aplicativo do AWS Mainframe Modernization em execução em um ambiente de execução do AWS Mainframe Modernization é iniciada ou reinicializada, ou quando um tempo limite expira durante uma tentativa de bloquear um conjunto de dados. Os bloqueios expirados são liberados imediatamente.

Para obter mais informações sobre a definição do aplicativo AWS Blu Age, consulte AWS Exemplo de definição do aplicativo Blu Age.

## Gerenciar bloqueios

No contexto de um ambiente de execução do AWS Mainframe Modernization usando o padrão de alta disponibilidade, um aplicativo AWS Blu Age pode ser implantado várias vezes. Para os aplicativos que lidam com conjuntos de dados Blusam, problemas de acesso simultâneo podem ocorrer. O gerenciador de bloqueios Blusam garante a integridade dos dados e gerencia o acesso de leitura e gravação a registros e conjuntos de dados, fornecendo bloqueios compartilhados entre os aplicativos que usam ElastiCache. Esse mecanismo permite que mais de um aplicativo leia o registro simultaneamente e garante que somente um aplicativo por vez grave o registro.

### Bloqueios de gravação

Para atualizar ou excluir um registro específico, o aplicativo deve primeiro bloquear o conjunto de dados que possui o registro e, em seguida, bloquear o próprio registro. Quando o registro é bloqueado, o bloqueio do conjunto de dados é liberado e outros registros do mesmo conjunto de dados ficam disponíveis para uso. Quando a operação de atualização ou exclusão for concluída, o bloqueio de registro retido será liberado. Somente um aplicativo por vez pode atualizar o registro, o que impede que outros aplicativos leiam ou gravem até que o bloqueio seja liberado, se a política de aplicativo definida permitir aguardar a liberação.

### Bloqueios de leitura

Desde que nenhum bloqueio de gravação seja mantido no registro ou no conjunto de dados, vários aplicativos podem ler os mesmos registros ao mesmo tempo. Para bloquear um registro para uma operação de gravação, todos os bloqueios de leitura devem ser liberados.

> **Note**
>
> O gerenciador de bloqueios Blusam manipula o acesso de vários threads em um determinado aplicativo usando o mesmo mecanismo de bloqueio.

# Não é possível acessar o URL de um aplicativo

- Motor: AWS Blu Age e Micro Focus
- Componente: aplicativos

Se você não conseguir acessar a URL de um aplicativo do AWS Mainframe Modernization em execução que você criou e implantou em um ambiente de tempo de execução de AWS Mainframe Modernization, talvez seja necessário configurar as regras de entrada no grupo de segurança que você associou ao ambiente de tempo de execução.

## Como esse erro ocorre

Quando você cria um ambiente de tempo de execução, o grupo de segurança fornecido, incluindo o grupo de segurança padrão, deve ter regras de entrada configuradas para permitir o tráfego para os aplicativos implantados de fora da VPC, se você quiser permitir esse tipo de acesso.

## Como você sabe se essa é a sua situação?

O aplicativo foi iniciado com êxito e está sendo executado normalmente, mas você não consegue se conectar a ele usando sua URL.

## O que você pode fazer?

Verifique se o grupo de segurança do Amazon VPC associado ao ambiente de execução permite tráfego para o ambiente nas portas apropriadas do aplicativo. Para verificar as regras do grupo de segurança, conclua as etapas a seguir:

1. Abra o console do AWS Mainframe Modernization em https://console.aws.amazon.com/m2/.
2. Na barra de navegação à esquerda, selecione Ambientes.
3. Escolha o ambiente de execução que hospeda o aplicativo ao qual você deseja conectar-se.
4. Escolha Configurações.
5. Em Security & Network, escolha o grupo de segurança. O link abre os detalhes do grupo de segurança no console do Amazon VPC.
6. Se necessário, escolha Editar regras de entrada e adicione a seguinte regra, se ainda não estiver presente:

   Tipo

   TCP personalizado

   Porta

   8196 ou a porta que corresponde às propriedades do ouvinte especificadas na definição do aplicativo. Para ter mais informações, consulte Etapa 2: Criar o aplicativo .

Origem

O endereço IP de onde você está chamando o aplicativo. Você pode escolher MyIP no menu suspenso. Se você ainda tiver problemas de tempo limite, tente escolher Anywhere IPV4 ou Anywhere IPV6. Certifique-se de interromper o aplicativo e iniciá-lo novamente depois de adicionar a regra de entrada no grupo de segurança.

Para obter mais informações, consulte Trabalhar com regras de grupo de segurança no Guia do Usuário do Amazon VPC.

# O AWS Blu Insights não abre no console

- Motor: AWS Blu Age
- Componente: Blu Insights

Quando você tenta acessar o Blu Insights a partir do console do AWS Mainframe Modernization, ele não abre e a nova guia é fechada imediatamente.

## Como esse erro ocorre

A função que você está usando para acessar o Blu Insights não tem permissões suficientes.

## O que você pode fazer?

Anexe uma política do IAM à função para permitir que ela acesse o Blu Insights. Certifique-se de que a política inclua pelo menos as seguintes permissões.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "m2:GetSignedBluinsightsUrl"
            ],
            "Resource": "arn:aws:m2:region:account:*"
            }
    ]
```

```
}
```

Certifique-se de substituir `region` e `account` com o Região da AWS e Conta da AWS correto.

# Histórico de documentos do Guia do usuário do AWS Mainframe Modernization

A tabela a seguir descreve as versões da documentação do AWS Mainframe Modernization.

| Alteração | Descrição | Data |
| --- | --- | --- |
| Tutorial atualizado do Managed Runtime para Micro Focus | Este tutorial mostra como implantar e executar o aplicativo de CardDemo amostra em um ambiente de tempo de execução gerenciado de modernização de AWS mainframe com o mecanismo de tempo de execução da Micro Focus. | 5 de fevereiro de 2024 |
| Notas de lançamento do AWS Blu Age Runtime and Modernization Tools versão 3.9.0. | Esta versão do AWS Blu Age Runtime and Modernization Tools está focada em vários aprimoramentos transversais em todo o produto, buscando aumentar o desempenho em arquiteturas de alta disponibilidade, juntamente com novos recursos para elevar a execução de tarefas a um novo patamar. | 18 de dezembro de 2023 |
| Transferir arquivos entre o mainframe e a AWS | Novo recurso lançado para transferir arquivos do mainframe de origem para a AWS. | 27 de novembro de 2023 |
| Gerencie transações para aplicativos | Novo recurso lançado para exibir e editar transações | 16 de outubro de 2023 |

| | | |
|---|---|---|
| | de aplicativos para o AWS Mainframe Modernization. | |
| Notas de versão do tempo de execução do AWS Blu Age e das ferramentas de Modernization versão 3.6.0. | Esta versão do AWS Blu Age Runtime and Modernization Tools fornece novos recursos para migrações legadas do zOS e do AS400, principalmente orientadas para expandir os mecanismos de suporte do CICS, complementar os recursos da JCL, otimizar o desempenho em recursos simultâneos e de alto volume e adicionar recursos. multi-data-source | 4 de agosto de 2023 |
| Agora você pode implantar uma nova versão de um aplicativo quando o aplicativo é interrompido. | Anteriormente, para implantar uma nova versão de um aplicativo, era necessário excluir a versão implantada. Agora você pode simplesmente interromper a versão implantada e implantar uma nova versão. | 26 de julho de 2023 |
| Pacote de tempo de execução do AWS Blu Age para facilitar a implementação do Amazon EC2 | O AWS Mainframe Modernization com o tempo de execução do AWS Blu Age agora está disponível com mais flexibilidade para configurar a pilha completa e a implantação nas instâncias do Amazon EC2 em seu Conta da AWS. | 6 de julho de 2023 |

| | | |
|---|---|---|
| Login único no Blu Age AWSAWS Blu Insights. | AWS AWS Blu Age Blu Insights está disponível no AWS Management Console por meio de login único. | 31 de março de 2023 |
| Lançamento do GA | Versão GA do Guia do Usuário do AWS Mainframe Modernization. | 8 de junho de 2022 |
| Lançamento inicial | Lançamento inicial (pré-visualização pública) do Guia do Usuário do AWS Mainframe Modernization. | 30 de novembro de 2021 |

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.