用户指南

AWS 大型机现代化

AWS 大型机现代化: 用户指南

Table of Contents

AWS 大型机现代化服务（托管运行时环境体验）不再向新客户开放。有关类似于 AWS 大型机现代化服务（托管运行时环境体验）的功能，请浏览 AWS 大型机现代化服务（自我管理体验）。现有客户可以继续正常使用该服务。有关更多信息，请参阅AWS 大型机现代化可用性变更。

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。

什么是 AWS 大型机现代化？

AWS 大型机现代化可帮助您将大型机应用程序现代化为 AWS 托管运行时环境。它提供了工具和资源来帮助您规划和实施迁移与现代化。您可以分析现有的大型机应用程序，使用 COBOL 或对其进行开发或更新PL/I，并实施自动化管道以实现应用程序的持续集成和持续交付(CI/CD)。您可以选择自动重构模式或更换平台模式，具体取决于您的客户需求。如果您是帮助客户迁移大型机工作负载的顾问，则可以在迁移和现代化之旅的所有阶段（从初始规划到迁移后的云运营）使用 AWS 大型机现代化工具。

您可以使用 AWS 大型机现代化来帮助您高效地创建和管理大型机应用程序 AWS 的运行时环境，以及管理和监控现代化的应用程序。

主题

> **Note**
>
> 在 AWS 大型机现代化项目中，您是否与大型机迁移能力合作伙伴或 AWS 专业服务部门合作？ 如果没有，我们强烈建议您为项目聘请专家。
>
> - AWS 大型机现代化能力合作伙伴
> - AWS 专业服务团队

AWS 大型机现代化的功能和用例支持演进式现代化方法，这种方法通过提高敏捷性来提供短期收益，并在以后提供大量优化和创新的机会。有关更多信息，请参阅 现代化方法。

AWS 大型机现代化的特点

AWS 大型机现代化功能支持以下用例：

- 评估： AWS 大型机现代化的评估能力可以帮助您评估、确定和规划迁移和现代化项目。
- 重构：在 AWS Blu Age 的支持下，您可以使用重构来转换传统应用程序编程语言、创建宏服务或微服务，以及对用户界面 (UIs) 和应用程序软件堆栈进行现代化改造。

  AWS Blu Insights 现在可 AWS 管理控制台 通过单点登录获得。您无需再管理单独的 AWS Blu Insights 凭证。你可以直接从中访问 AWS AWS Blu Age Codebase 和 Transformation Center 功能。 AWS 管理控制台
- 更换平台：在 Micro Focus Enterprise 解决方案的支持下，您可以移植应用程序，并且无需更改即可重新编译大部分应用程序源代码。
- 开发者 IDE： AWS 大型机现代化提供了按需集成开发环境 (IDE)，因此开发人员可以通过智能编辑和调试、即时代码编译和单元测试更快地编写代码。
- 托管运行时： AWS 大型机现代化托管运行时环境持续监控您的集群，通过自我修复计算和自动扩展保持企业工作负载的运行。
- 持续集成和交付(CI/CD)： AWS 大型机现代化的CI/CD功能可帮助应用程序开发团队更频繁、更可靠地交付代码更改，从而加快迁移速度，提高质量，并有助于减少新业务功能 time-to-market的发布。
- 与其他 AWS 服务的集成： AWS 大型机现代化支持 AWS CloudFormation、 AWS PrivateLink、可重复部署以及 AWS Key Management Service 更高的安全性和合规性。
- 扩大可用性： AWS 大型机现代化现已在美国东部（俄亥俄州）、美国西部（加利福尼亚北部）、亚太地区（孟买）、亚太地区（首尔）、亚太地区（新加坡）、亚太地区（东京）、欧洲（伦敦）和欧洲（巴黎）推出。

有关更多信息，请参阅AWS 大型机现代化功能。

图案

由 AWS Blu Age 支持的自动重构模式侧重于通过将完整的传统应用程序堆栈及其数据层转换为基于 Java 的现代应用程序来加速现代化，同时保持功能等同性。在此自动转换过程中创建的多层应用程序，具有基于 Angular 的前端、支持 API 的 Java 后端和可访问现代数据存储的数据层。重构过程提供了与遗留堆栈同等的功能，可提高项目自动化程度来提升速度和质量并降低成本，从而更快地实现业务收益。有关更多信息，请参阅 AWS Mainframe Modernization 自动重构。

Replatforming 模式由 Rocket Software（前身为 Micro Focus）企业套件提供支持，其重点是保留应用程序语言、代码和工件，以最大限度地减少对应用程序资产和团队的影响。它可以帮助客户保持对应用程序知识和技能的使用。虽然应用程序的更改有限，但这种模式也有助于基础设施和流程的现代化。基础设施更改为基于云的现代托管服务，同时流程更改为遵循应用程序开发和 IT 运营的最佳实操。有关更多信息，请参阅 AWS Mainframe Modernization 更换平台

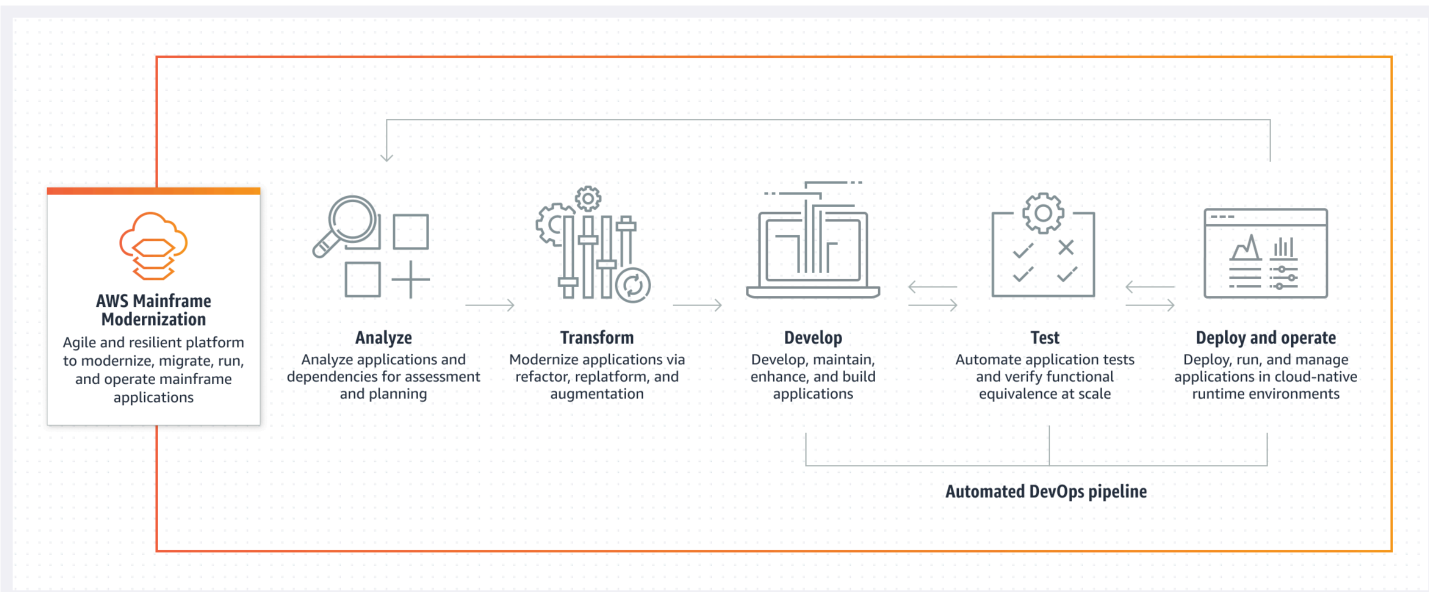
如何开始进行 AWS 大型机现代化

试试看！我们提供教程和示例应用程序，以帮助您了解 AWS 大型机现代化提供的功能。选择教程：为 AWS Blu Age 设置托管运行时或查看教程：为 Rocket Software（前身为 Micro Focus）完整的 step-by-step教程。

如果您对自动重构感兴趣，请查看 AWS Blu Age 工具，网址为。BluInsights你也可以将 WorkSpaces 应用程序设置为访问 AWS Blu Age Developer IDE、Rocket Enterprise Analyzer（前身为 Micro Focus Enterprise Analyzer）和 Rocket Enterprise D

教程和示例应用程序只能让你了解 AWS 大型机现代化所提供的内容。当您准备启动现代化项目时，请参阅现代化方法，了解有关现代化项目的各阶段和任务的详细信息。

下图显示了 AWS 大型机现代化服务用于分析、转换、开发、测试、部署和操作大型机应用程序的工作流程。



相关服务

除了用于自动重构的 Blu Insights 之外，您还可以在 AWS 大型机现代化中使用以下 AWS 服务。

- Amazon RDS，用于托管迁移的数据库
- Amazon S3，用于存储应用程序的二进制文件和定义文件
- Amazon FSx 或 Amazon EFS 用于存储应用程序数据
- 亚马逊 AppStream 可以访问火箭企业分析器和 Rocket Enterprise 开发者工具

- CloudFormation 用于为迁移的应用程序设置CI/CD的自动 DevOps 管道
- AWS Migration Hub
- AWS DMS 用于迁移数据库

访问 AWS 大型机现代化

目前，您可以通过控制台访问 AWS 大型机现代化，网址为https://console.aws.amazon.com/m2/。有关提供 AWS 大型机现代化的区域列表，请参阅中的AWS 大型机现代化终端节点和配额。Amazon Web Services 一般参考

您是首次使用 AWS 大型机现代化改造的用户吗？

如果您是首次使用 AWS 大型机现代化的用户，我们建议您先阅读以下章节：

- 开始进行 AWS 大型机现代化改造
- 为 AWS 大型机现代化做好准备

AWS 大型机现代化的定价

AWS 大型机现代化对支持托管运行时环境的实例的使用收费。此外， AWS 大型机现代化还提供一些工具，无需支付额外费用。您应负责支付与 AWS 大型机现代化相关的其他 AWS 服务所产生的费用。AWS 将在使用 AWS 大型机现代化的任何定价变更生效前 30 天发出通知。有关更多信息，请参阅使用进行大型机现代化。 AWS

使用 AWS Blu Insights，您需要为转型中心的使用付费。有关更多信息，请参阅 AWS Mainframe Modernization 定价。

AWS 大型机现代化可用性变更

AWS AWS 大型机现代化服务（托管运行时环境体验）已停止接受新客户，而现有客户可以继续使用该服务，而不会中断正在进行的现代化项目。该服务提供第三方软件解决方案Rocket的托管版本和自我管理版本，用于平台重建。此服务可用性更改仅影响托管版本。新项目将定向到自我管理版本。

虽然我们不会为该服务添加新功能，但我们仍然致力于提供安全更新并保持服务的可用性，以确保正在进行的现代化项目继续顺利进行。我们的重点是确保现有客户有一个稳定的环境，以完成其大型机现代化计划，同时为过渡到自我管理版本做准备。

AWS AWS 大型机现代化服务（自我管理体验）是我们推荐的解决方案，它为大型机现代化提供相同的核心功能。

如果您还有其他问题，请通过联系我们 AWS 支持 或阅读我们的 FAQs

1. 这对服务意味着什么（你要关闭服务吗？）？ AWS AWS 大型机现代化服务（托管运行时环境体验）已停止接受新客户。该服务将继续为现有客户提供服务，以完成他们正在进行的迁移项目。
2. 现有客户将受到怎样的影响？ 现有客户将继续获得全面支持，如果他们选择过渡到自我管理运行时， AWS Proserve 可以提供帮助。如AWS 支持需帮助，请联系。
3. 如果现有客户遇到问题，如何才能上报？ 现有的托管服务客户可以继续与 AWS Supp ort 合作以解决问题， AWS 团队将继续为客户提供支持。
4. 客户可以探索哪些替代方案？ AWS AWS 大型机现代化服务的自我管理版本为 Rocket Software（重平台）和 Blu Age（重构）功能提供运行时功能。
5. 客户如何从 AWS AWS 大型机现代化服务（托管运行时环境体验）迁移？ 将继续为当今使用托管运行时的客户提供支持。要迁移现有环境， AWS ProServe 可以邀请他们协助过渡到使用自我管理选项。

为 AWS 大型机现代化做好准备

在开始使用 AWS 大型机现代化之前，您或您的管理员需要先注册，使用管理设置创建用户 AWS 账户，并保护您的 IAM 用户。

主题

- 注册获取 AWS 账户
- 创建具有管理访问权限的用户

注册获取 AWS 账户

如果您没有 AWS 账户，请完成以下步骤来创建一个。

报名参加 AWS 账户

1. 打开https://portal.aws.amazon.com/billing/注册。
2. 按照屏幕上的说明操作。

   在注册时，将接到电话或收到短信，要求使用电话键盘输入一个验证码。

   当您注册时 AWS 账户，就会创建AWS 账户根用户一个。根用户有权访问该账户中的所有 AWS 服务 和资源。作为最佳安全实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

AWS 注册过程完成后会向您发送一封确认电子邮件。您可以随时前往 https://aws.amazon.com/并选择“我的账户”，查看您当前的账户活动并管理您的账户。

创建具有管理访问权限的用户

注册后，请保护您的安全 AWS 账户 AWS 账户根用户 AWS IAM Identity Center，启用并创建管理用户，这样您就不会使用 root 用户执行日常任务。

保护你的 AWS 账户根用户

1. 选择 Root 用户并输入您的 AWS 账户 电子邮件地址，以账户所有者的身份登录。AWS 管理控制台在下一页上，输入您的密码。

要获取使用根用户登录方面的帮助，请参阅《AWS 登录 用户指南》中的 Signing in as the root user。

2. 为您的根用户启用多重身份验证（MFA）。

   有关说明，请参阅 I A M 用户指南中的为 AWS 账户 根用户启用虚拟 MFA 设备（控制台）。

创建具有管理访问权限的用户

1. 启用 IAM Identity Center。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的 Enabling AWS IAM Identity Center。

2. 在 IAM Identity Center 中，为用户授予管理访问权限。

   有关使用 IAM Identity Center 目录 作为身份源的教程，请参阅《用户指南》 IAM Identity Center 目录中的使用默认设置配置AWS IAM Identity Center 用户访问权限。

以具有管理访问权限的用户身份登录

- 要使用您的 IAM Identity Center 用户身份登录，请使用您在创建 IAM Identity Center 用户时发送到您的电子邮件地址的登录网址。

  有关使用 IAM Identity Center 用户登录的帮助，请参阅AWS 登录 用户指南中的登录 AWS 访问门户。

将访问权限分配给其他用户

1. 在 IAM Identity Center 中，创建一个权限集，该权限集遵循应用最低权限的最佳做法。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的 Create a permission set。

2. 将用户分配到一个组，然后为该组分配单点登录访问权限。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的 Add groups。

AWS 大型机现代化概念

AWS 大型机现代化提供了工具和资源，可帮助您迁移、现代化和运行大型机工作负载。 AWS您可以使用此页面来了解 AWS 大型机现代化中的各种概念，包括应用程序、现代化、环境、平台重构、重构和运行时引擎。

主题

应用程序

大型机现代化中正在运行的大型 AWS 机工作负载。应用程序由一组批处理作业、交互式事务（CICS 或 IMS）或其他组件组成。您可以定义其范围。您必须定义和指定工作负载所需的任何组件或资源，例如 CICS 事务或批处理作业。

应用程序定义

大型机现代化中运行的应用程序（大型机工作负载）所需的组件和资源的定义或规范。 AWS 请务必将定义与应用程序本身分开，因为可以对由不同的运行时环境表示的多个阶段（预生产、生产）重复使用相同的定义。

批处理作业

配置为无需用户交互即可运行的计划程序。在 AWS Mainframe Modernization 中，您需要将批处理作业 JCL 文件和批处理作业二进制文件存储在 Amazon S3 存储桶中，并在应用程序定义文件中提供两者的位置。运行批处理作业时， AWS 大型机现代化会报告以下状态值：

正在提交

批处理作业正在提交。

暂停

批处理作业处于暂停状态。

正在分派

批处理作业正在分派。

运行

批处理作业当前正在运行。

正在取消

批处理作业正在被取消。

已取消

批处理作业已取消。

成功

批处理作业已成功完成运行。

失败

批处理作业失败。

成功但出现警告

批处理作业成功完成运行，但报告了一个小错误。作为 GetBatchJobExecution 响应的一部分返回的任务条件代码指出了错误的原因。

配置

环境或应用程序的特征。环境配置包括引擎类型、引擎版本、可用性模式、可选文件系统配置等。

应用程序配置可以是静态的，也可以是动态的。只有当您通过部署新版本来更新应用程序时，静态配置才会发生变化。动态配置（通常是一种操作活动，例如开启或关闭跟踪）会在更新后立即发生变化。

数据集

包含供应用程序使用的数据的文件。

环境

为托管一个或多个应用程序而创建的 AWS 计算资源、运行时引擎和配置详细信息的命名组合。

大型机现代化

将应用程序从传统大型机环境迁移到 AWS的过程。

迁移之旅

传统应用程序的迁移和现代化 end-to-end过程，通常由以下几个阶段组成：评估、移动、迁移和现代化以及运营和优化。

挂载点

文件系统中的一个目录，用于访问存储在该系统中的文件。

自动重构

对遗留应用程序构件进行现代化而使其在现代云环境中运行的过程。该过程可以包括代码和数据转换。有关更多信息，请参阅 AWS Mainframe Modernization 自动重构。

更换平台

将应用程序和应用程序构件从一个计算平台移动到另一个计算平台的过程。有关更多信息，请参阅 AWS Mainframe Modernization 更换平台。

资源

计算机系统中的物理或虚拟组件。

运行时引擎

便于应用程序运行的软件。

现代化方法

迁移很复杂，有很多变量。 AWS 大型机现代化提供了一种进化的方法，它通过提高敏捷性来提供一些短期收益，并有大量机会在以后进行优化和创新。此外， AWS 大型机现代化有助于简化旅程，同时仍然尊重客户公司和业务的细节。 AWS 大型机现代化支持的两种主要方法是自动重构或平台重构。选择哪种方法取决于您的客户的具体情况。

自动重构使用 AWS Blu Age 工具将代码、数据和依赖关系自动转换为现代语言、数据存储和框架，同时保证功能等同于相同的业务功能。

Replatforming 使用 Rocket Software（前身为 Micro Focus）工具将大型机工作负载转换为敏捷 AWS

您可以考虑分阶段实现现代化之旅。第一阶段包括三个阶段：评测、动员以及迁移与现代化。第二阶段包括运行和优化，您可以在其中发现更多的创新机会。

主题

评测阶段

概况来说，评测阶段着眼于您是否准备好迁移。您定义一个业务案例，然后通过 AWS提供的研讨会和沉浸日（演示和实验室）来培训团队成员。研讨会和沉浸日涵盖不同的主题。这些任务是在 AWS 大型机现代化之外执行的。

动员阶段

在动员阶段，首先启动项目，然后执行发现过程，从大型机应用程序中提取数据并将其摄取到迁移工具中。确定要迁移的应用程序，然后选择几个应用程序进行试验。完善业务案例，制定迁移计划，并决定如何处理安全性与合规性、账户治理以及运营模式。与适合的团队成员合作设置一个云卓越中心。运行试点并记录您了解到的内容。完善迁移计划和业务案例。其中许多任务是在 AWS 大型机现代化之外执行的。

迁移与现代化阶段

迁移和现代化阶段适用于每个应用程序，包括多项任务，包括分配人员、进行深入发现、找出正确的应用程序架构 AWS、设置应用程序运行时环境、重构平台或重构代码、与其他系统集成，当然还有测试。在该阶段结束时，将更换平台或重构后的应用程序部署到生产环境中，并切换到 AWS 上的新系统。这些任务中的大多数或全部是在 AWS 大型机现代化、其他 AWS 服务或 AWS 大型机现代化提供访问权限的工具中执行的。

如果要使用自动重构，请参阅 Blu Insights。 AWS Blu Insights 现在可 AWS 管理控制台 通过单点登录获得。您无需再管理单独的 AWS Blu Insights 凭证。你可以直接从中访问 AWS AWS Blu Age Codebase 和 Transformation Center 功能。 AWS 管理控制台

要将数据从大型机迁移到 AWS，我们建议使用 AWS SCT 和。 AWS Database Migration Service有关更多信息，请参阅《AWS Schema Conversion Tool 用户指南》中的什么是 AWS Schema Conversion Tool？ 和《AWS Database Migration Service 用户指南》中的什么是 AWS Database Migration Service？

操作和优化阶段

在操作和优化阶段，您将专注于监控已部署的应用程序、管理资源以及确保安全性和合规性为最新状态。您还可以评测是否有机会优化迁移的工作负载。

开始进行 AWS 大型机现代化改造

您可以通过以下教程开始使用 AWS 大型机现代化，这些教程向您介绍该服务和每个运行时引擎。

主题

要进一步学习更多内容，请参阅以下教程。

- 教程：为 BankDemo 示例应用程序设置 Rocket 软件（前身为 Micro Focus）版本
- 教程：设置 WorkSpaces 应用程序以用于 Rocket Enterprise Analyzer 和 Rocket

教程：为 AWS Blu Age 设置托管运行时

您可以使用本教程中指定的演示应用程序将 AWS Blu Age 现代化应用程序部署到 AWS 大型机现代化运行时环境中。

主题

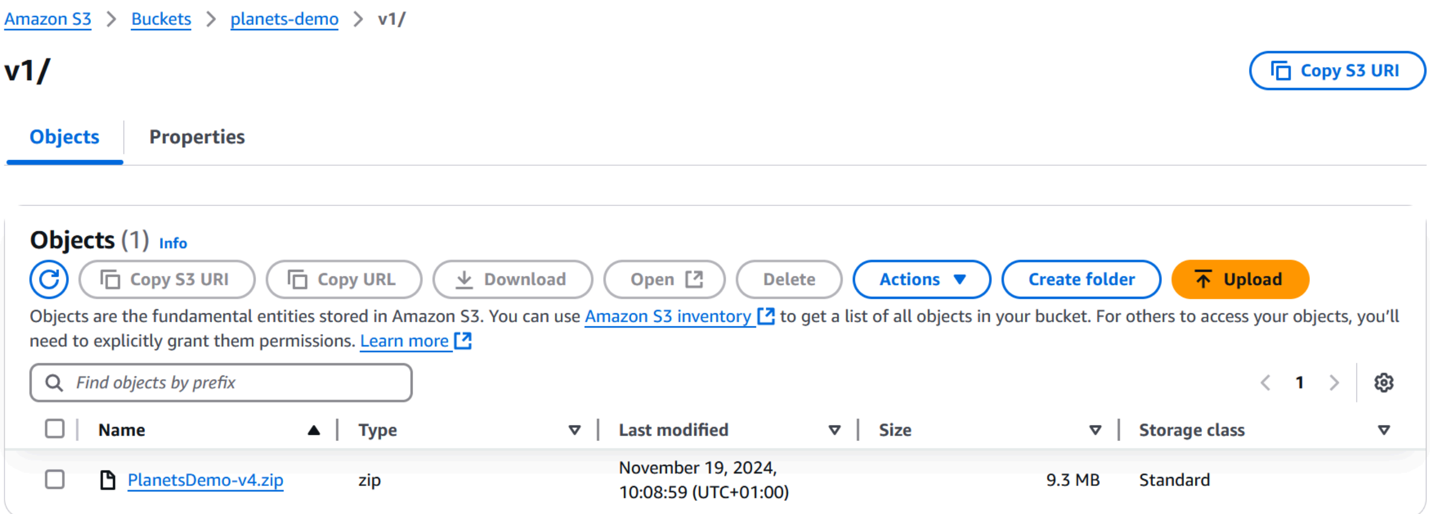
先决条件

要完成本教程，请下载演示应用程序档案 PlanetsDemo-v4.zip。

需要使用现代浏览器才能访问正在运行的演示应用程序。您是在桌面上还是从 Amazon Elastic Compute Cloud 实例（例如，在 VPC 内）运行此浏览器，决定了您的安全设置。

步骤 1：上传演示应用程序

将演示应用程序上传到 Amazon S3 存储桶。请确保此存储桶与您要部署的应用程序位于相同的 AWS 区域 。以下示例显示了一个名为 p lanets-demo 的存储桶，其密钥前缀或文件夹名为 v1，存档名为 v1。`planetsdemo-v4.zip`



> **Note**
>
> 存储桶中的文件夹是必需的。

步骤 2：创建应用程序定义

要将应用程序部署到托管运行时，您需要一个 AWS 大型机现代化应用程序定义。此定义是一个描述应用程序位置和设置的 JSON 文件。以下示例是演示应用程序的应用程序定义：

```
{
    "template-version": "2.0",
    "source-locations": [{
        "source-id": "s3-source",
        "source-type": "s3",
        "properties": {
            "s3-bucket": "planets-demo",
            "s3-key-prefix": "v1"
```

```
        }
    }],
    "definition": {
        "listeners": [{
            "port": 8196,
            "type": "http"
        }],
        "ba-application": {
            "app-location": "${s3-source}/PlanetsDemo-v4.zip"
        }
    }
}
```

将`s3-bucket`条目更改为示例应用程序 zip 文件的名称（例如`planets-demo`），将`app-location`条目更改为存储示例应用程序 zip 文件的 S3 路径（例如`${s3-source}/PlanetsDemo-v4.zip`）。

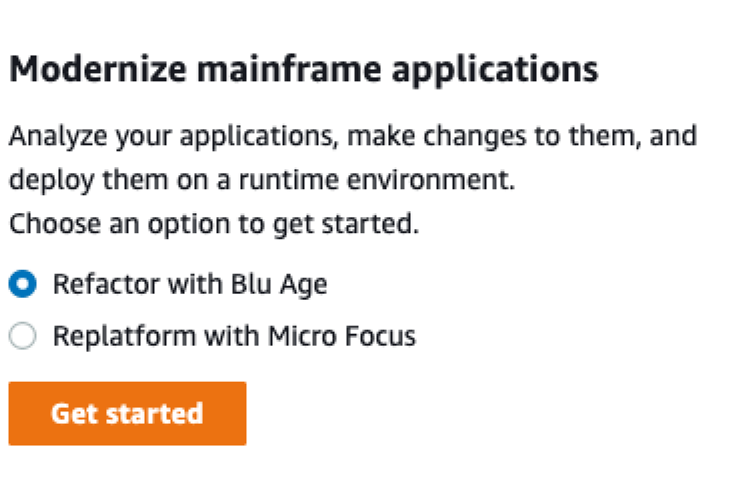
> **Note**
> 请务必在本地以文本文件形式创建应用程序定义文件。

有关应用程序定义的更多信息，请参阅 AWS Blu Age 应用程序定义示例。

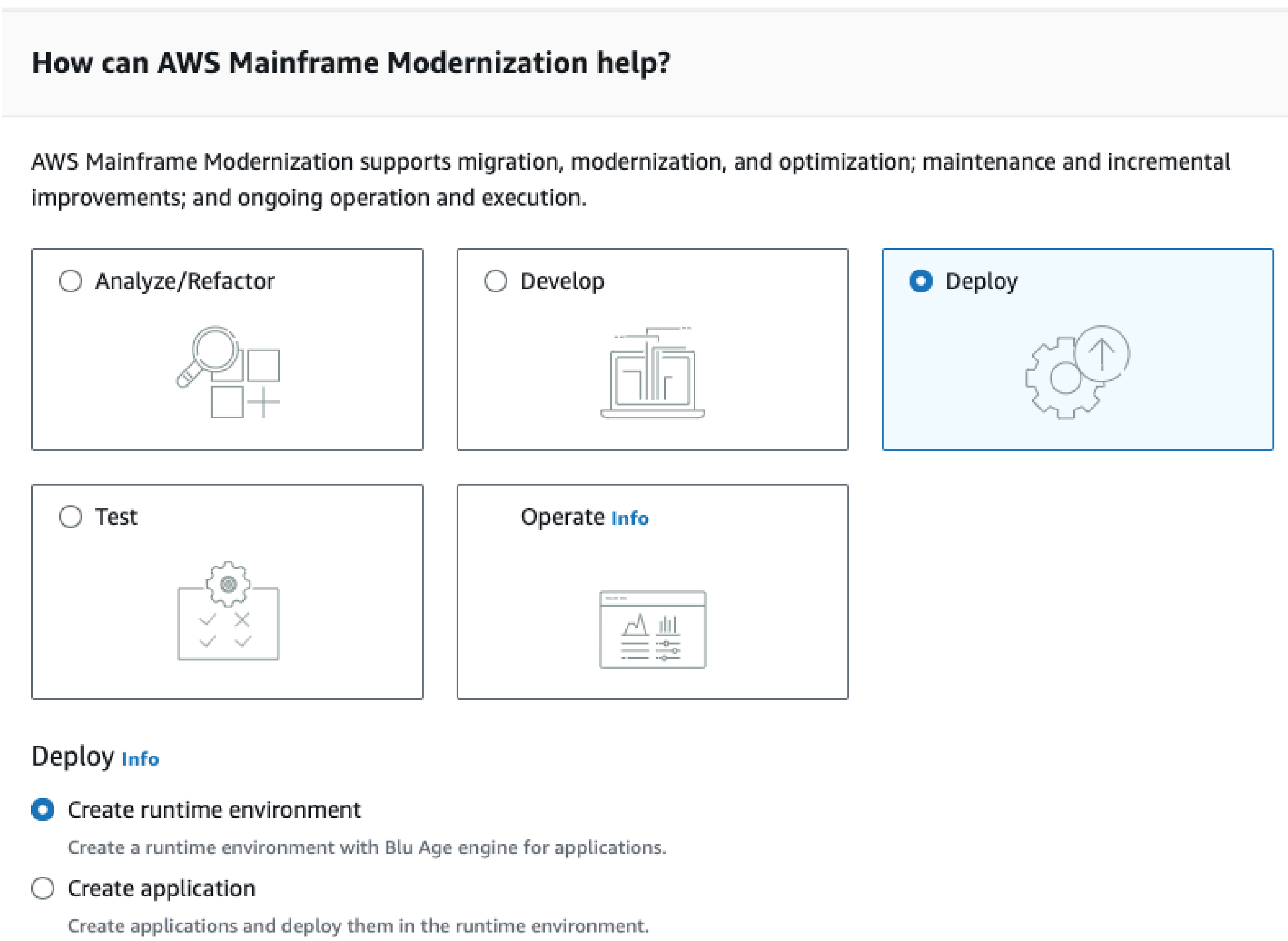
步骤 3：创建运行时环境

要创建 AWS 大型机现代化运行时环境，请执行以下步骤：

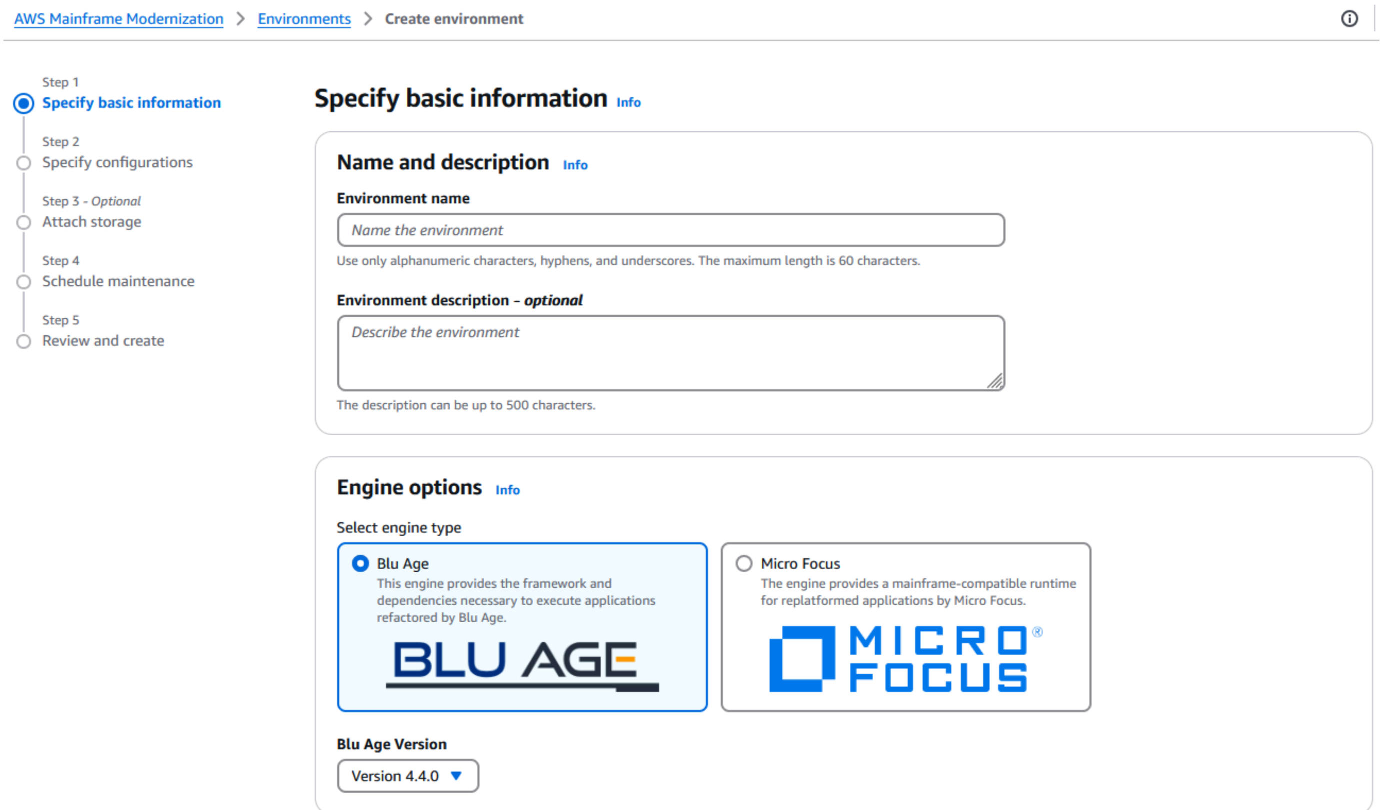
1. 使用 AWS Mainframe Modernization 控制台。
2. 在 AWS 区域 选择器中，选择要在其中创建环境的区域。这 AWS 区域 必须与您在其中创建 S3 存储桶的区域相匹配步骤 1：上传演示应用程序。
3. 在对大型机应用程序进行现代化下，选择使用 Blu Age 重构，然后选择开始。



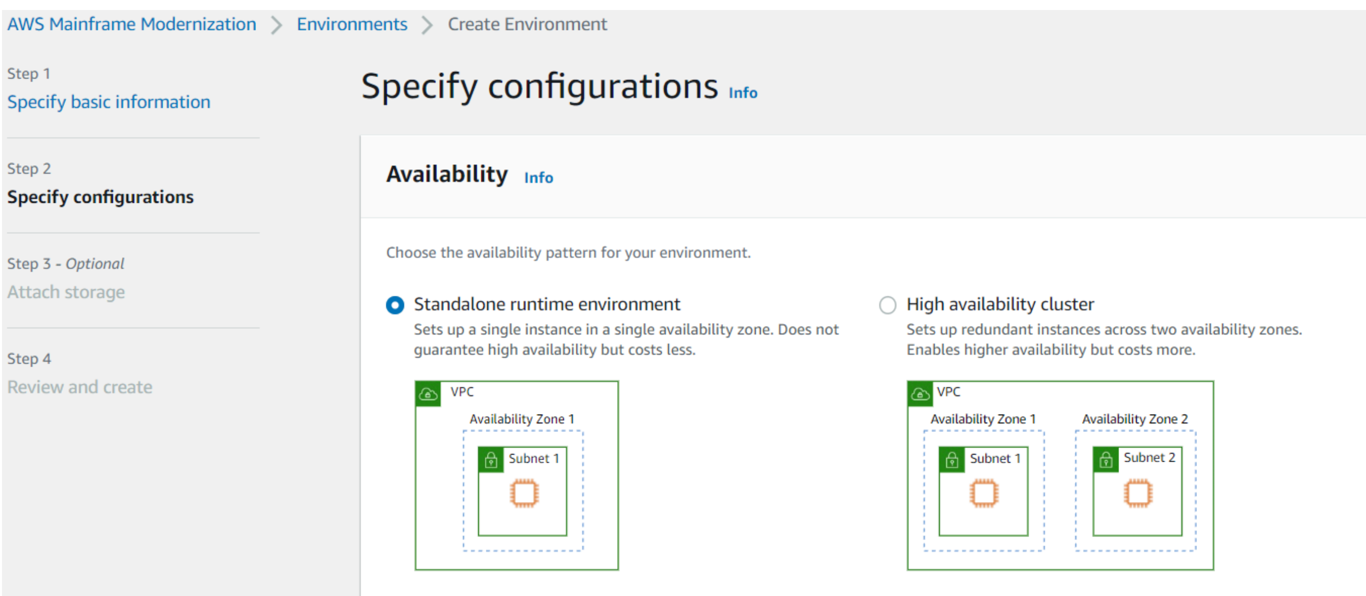
4. 在 AWS Mainframe Modernization 如何帮助您下，选择部署和创建运行时环境。



5. 在左侧导航中，选择环境，然后选择创建环境。在指定基本信息页面上，输入您的环境的名称和描述，然后确保选择 AWS Blu Age 引擎。您可以选择向创建的资源添加标签。然后选择下一步。



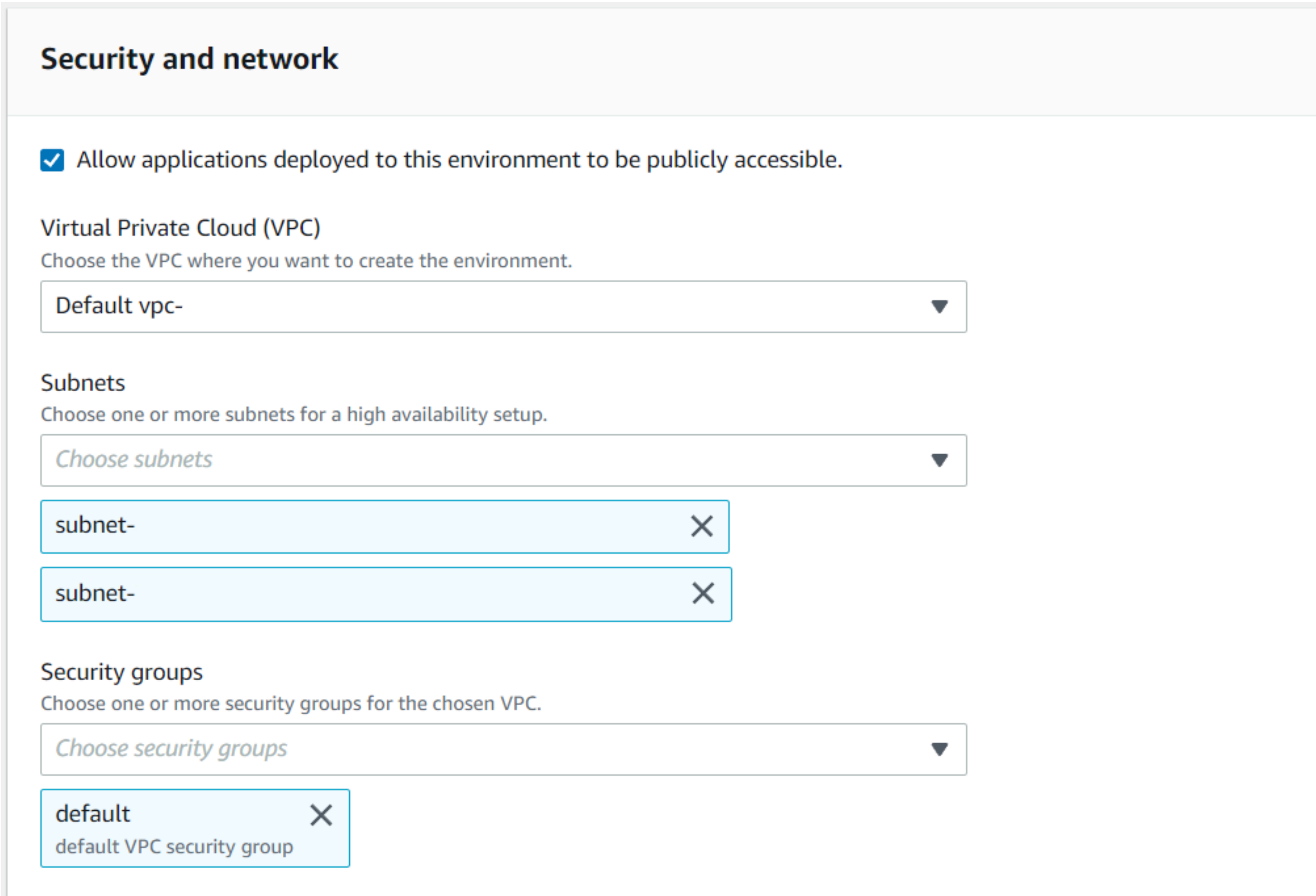
6. 在指定配置页面上，选择独立运行时环境。



7. 在安全和网络下，进行以下更改：

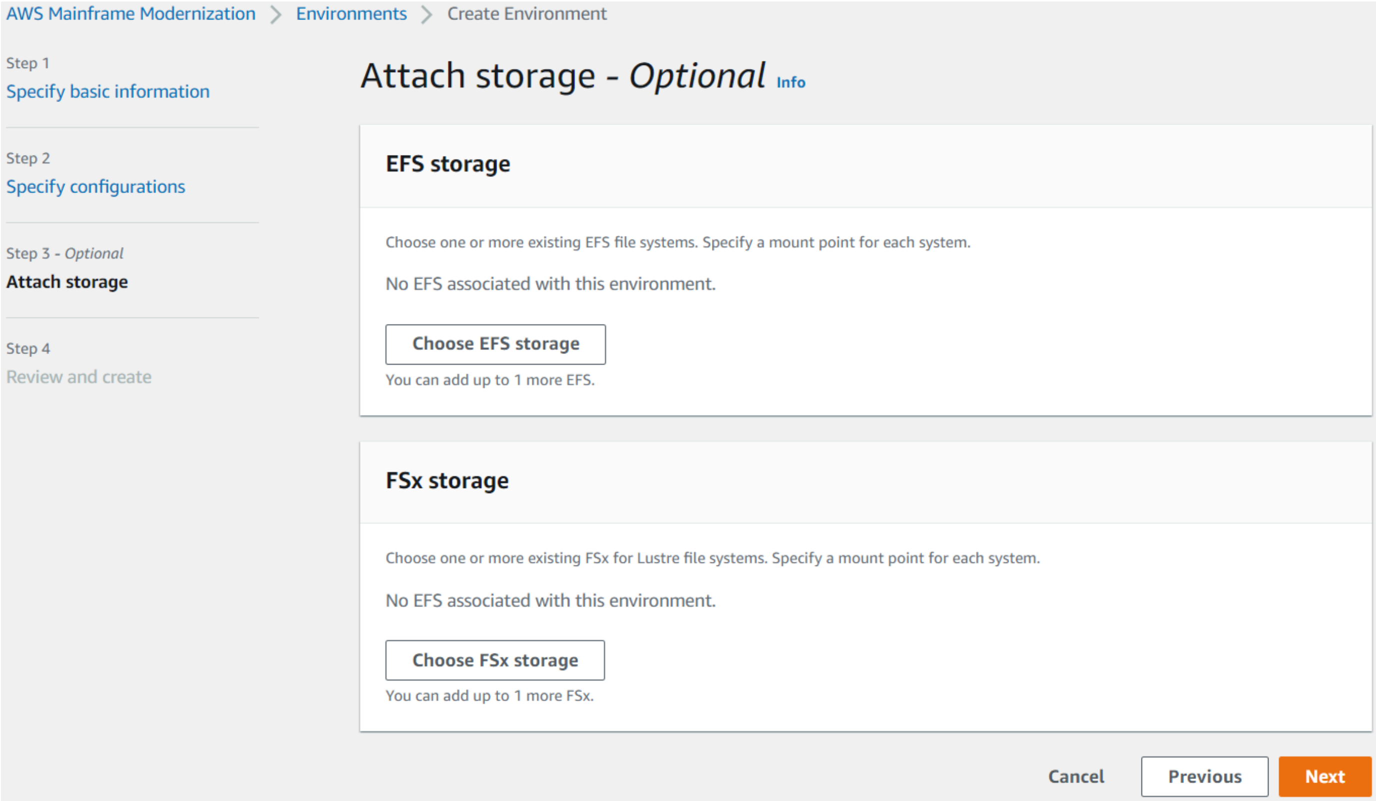
- 选择允许部署到此环境的应用程序可公开访问。此选项为应用程序分配一个公有 IP 地址，以便您可以从桌面访问它。
- 选择 VPC。您可以使用默认值。

- 选择两个子网。确保子网允许分配公有 IP 地址。
- 选择安全组。您可以使用默认值。请确保您所选的安全组允许从浏览器 IP 地址访问您在应用程序定义的 `listener` 属性中指定的端口。有关更多信息，请参阅 步骤 2：创建应用程序定义。



如果要从所选 VPC 的外部访问应用程序，请确保已正确配置该 VPC 的入站规则。有关更多信息，请参阅 错误故障排除：无法访问应用程序 URL。

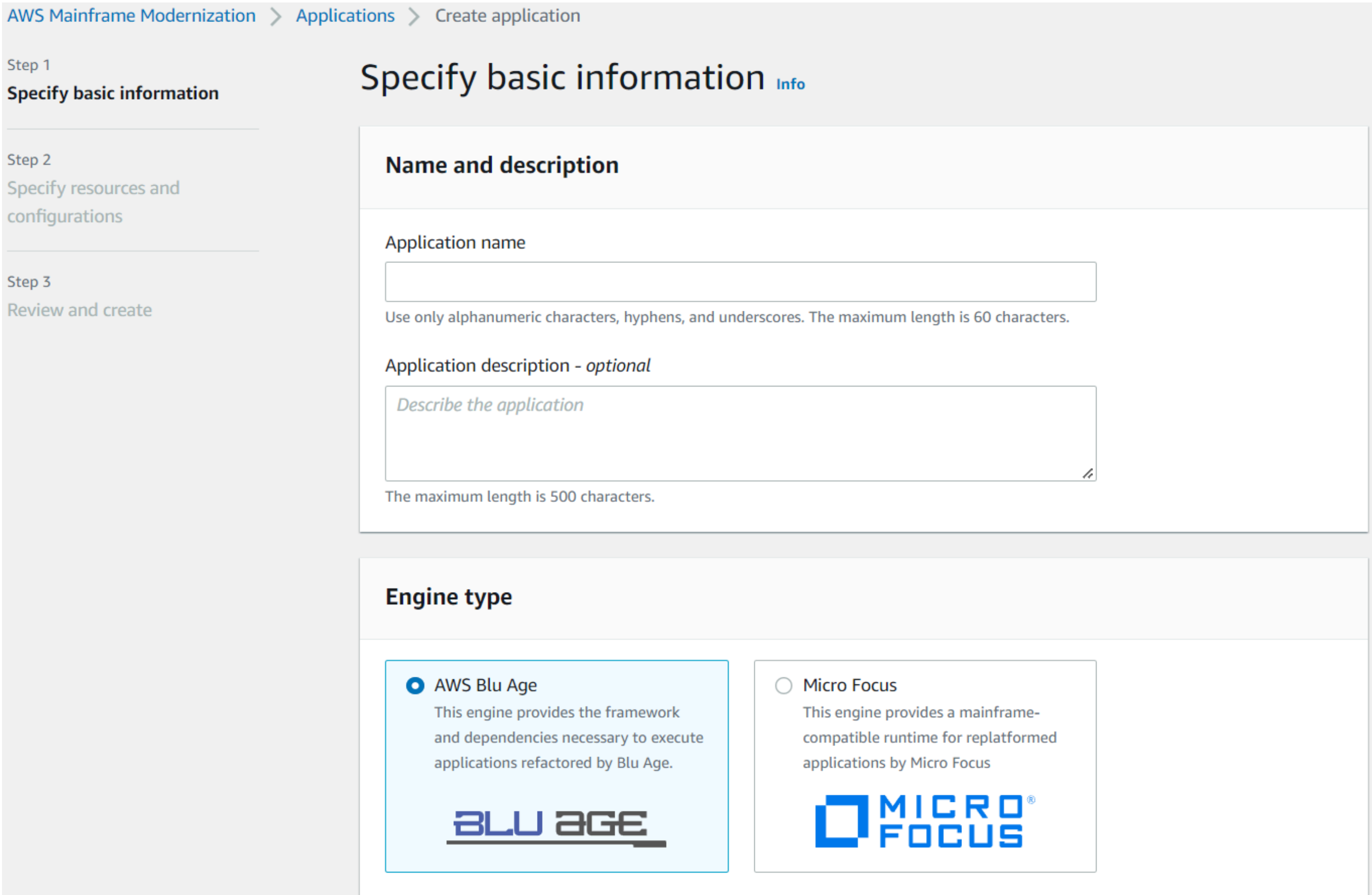
8. 选择下一步。
9. 在附加存储 – 可选中，保留默认选项，然后选择下一步。



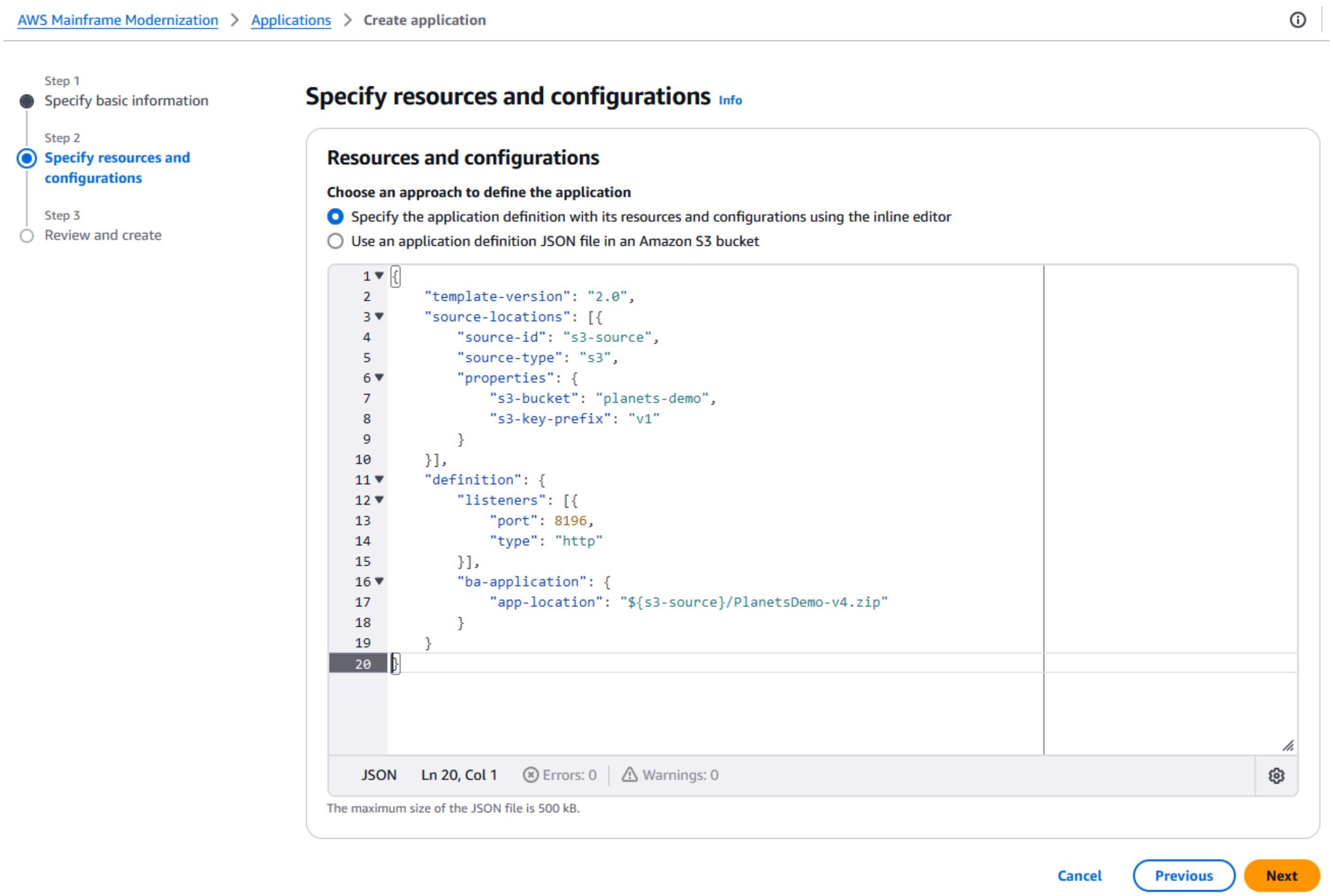
10. 在计划维护中，选择无首选项，然后选择下一步。
11. 在审核并创建中，审核信息，然后选择创建环境。

步骤 4：创建应用程序

1. 在 AWS 管理控制台中导航到 AWS Mainframe Modernization。
2. 在导航窗格中，选择应用程序，然后选择创建应用程序。在指定基本信息页面上，输入应用程序的名称和描述，然后确保选择 AWS Blu Age 引擎。然后选择下一步。



3. 在指定资源和配置页面上，复制并粘贴在the section called “步骤 2：创建应用程序定义”中创建的更新的应用程序定义 JSON。



4. 在审核和创建中，审核您的选择，然后选择创建应用程序。

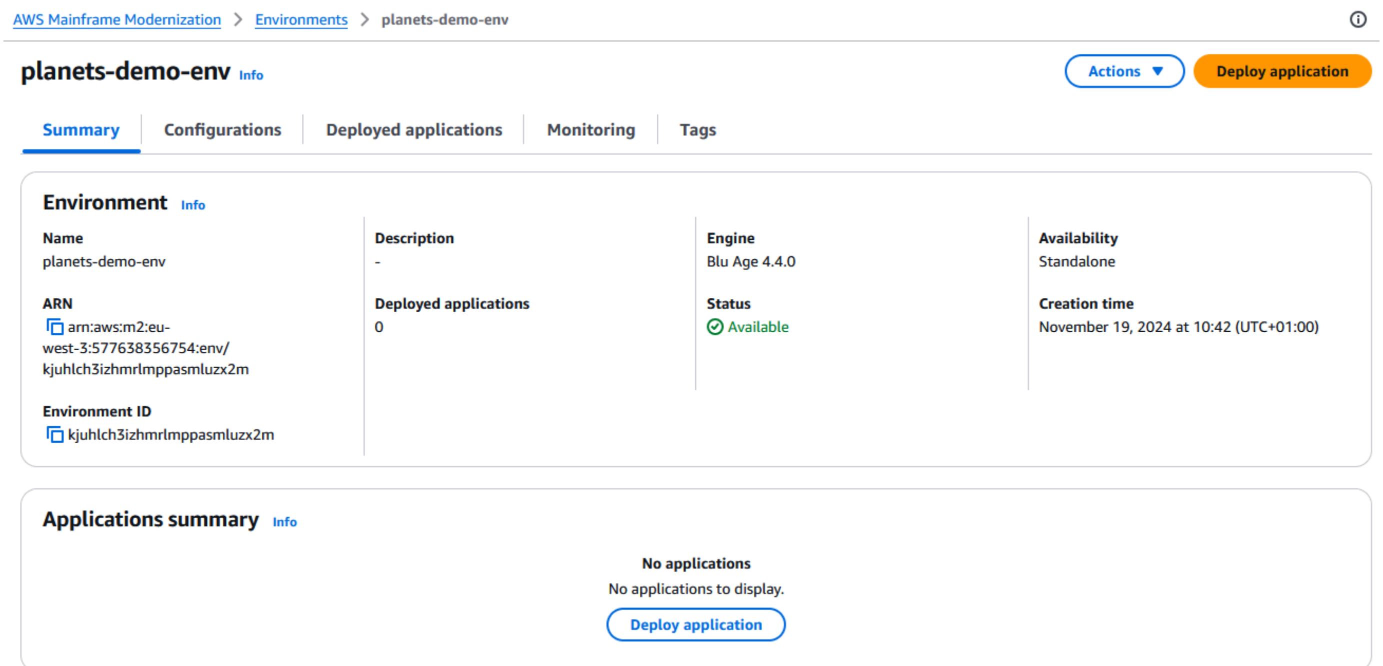
> **Note**
>
> 如果您的应用程序创建失败，请检查您输入的 S3 路径，因为该路径区分大小写。

步骤 5：部署应用程序

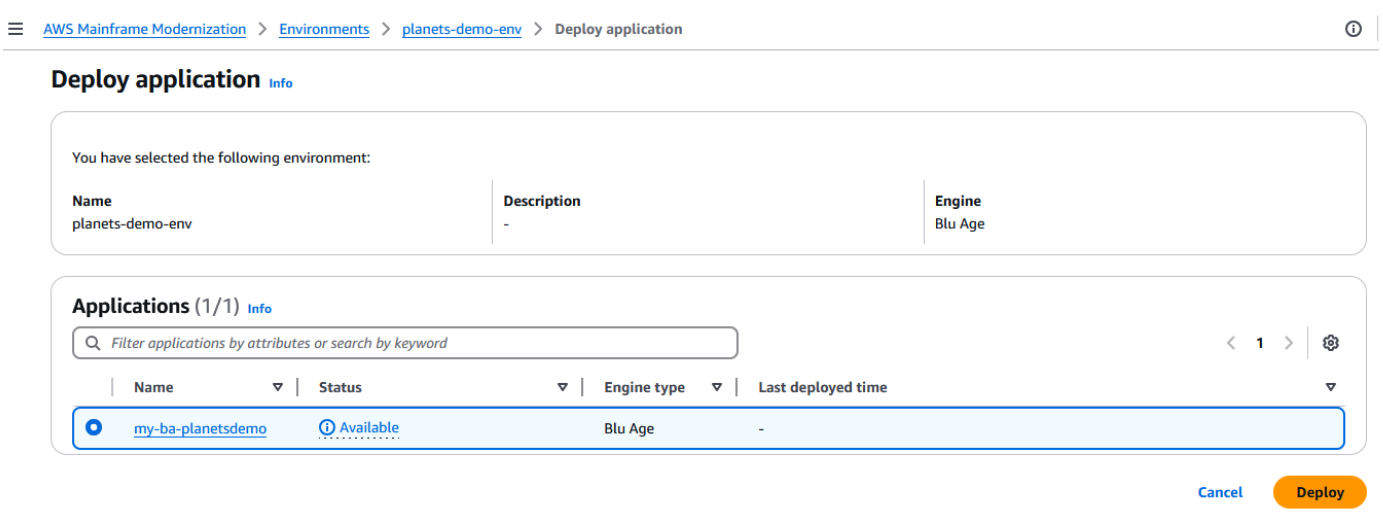
成功创建 AWS 大型机现代化运行时环境和应用程序，并且两者都处于 “可用” 状态后，就可以将应用程序部署到运行时环境中。为此，请完成以下步骤：

1. 在 AWS 管理控制台中导航到 AWS 大型机现代化。在导航窗格中，选择环境。随后显示环境列表页面。

2. 选择之前创建的运行时环境。随后显示环境详细信息页面。
3. 选择部署应用程序。



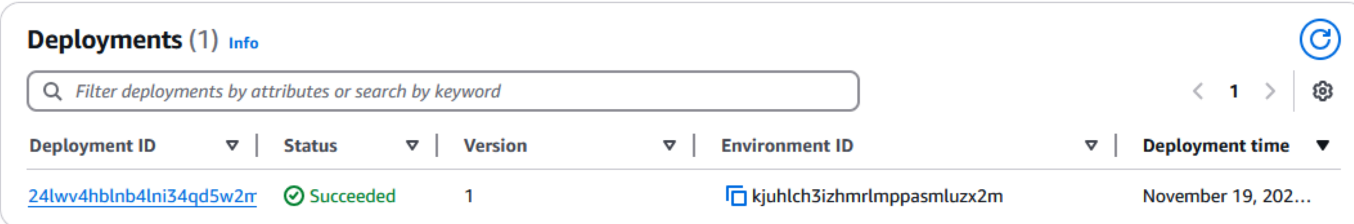
4. 选择前面创建的应用程序，然后选择要将应用程序部署到的版本。然后选择部署。



5. 等待直到应用程序完成部署。您将看到一条横幅，带有消息已成功部署应用程序。

步骤 6：启动应用程序

1. 导航到 “AWS 大型机现代化”， AWS 管理控制台 然后选择 “应用程序”。
2. 选择您的应用程序，然后转到部署。应用程序的状态应为成功。



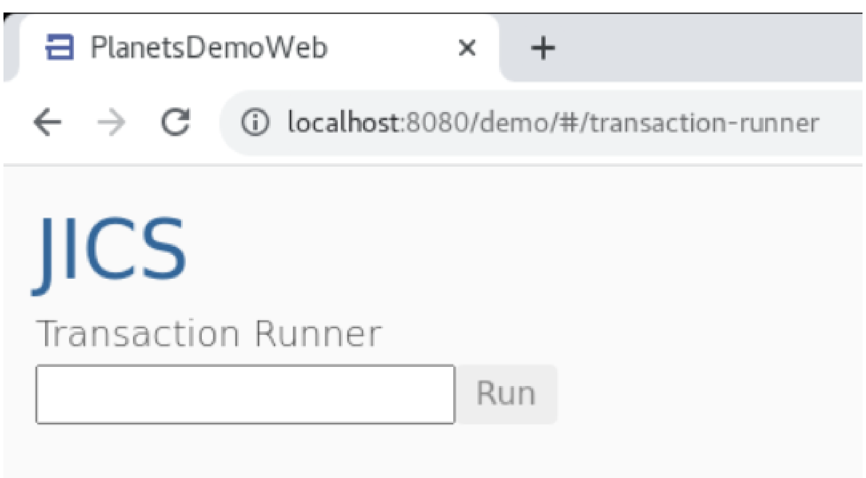
3. 选择操作，然后选择启动应用程序。

步骤 7：访问应用程序

1. 等待直到应用程序处于正在运行状态。您将看到一条横幅，带有消息已成功启动应用程序。
2. 复制应用程序 DNS 主机名。您可以在应用程序的应用程序信息部分中找到此主机名。
3. 在浏览器中，导航到 `http://{hostname}:{portname}/PlanetsDemo-web-1.0.0/`，其中：

   - `hostname` 是之前复制的 DNS 主机名。
   - `portname` 是您在步骤 2：创建应用程序定义中创建的应用程序定义中定义的 Tomcat 端口。

   随后显示 JICS 屏幕。



如果您无法访问应用程序，请参阅错误故障排除：无法访问应用程序 URL。

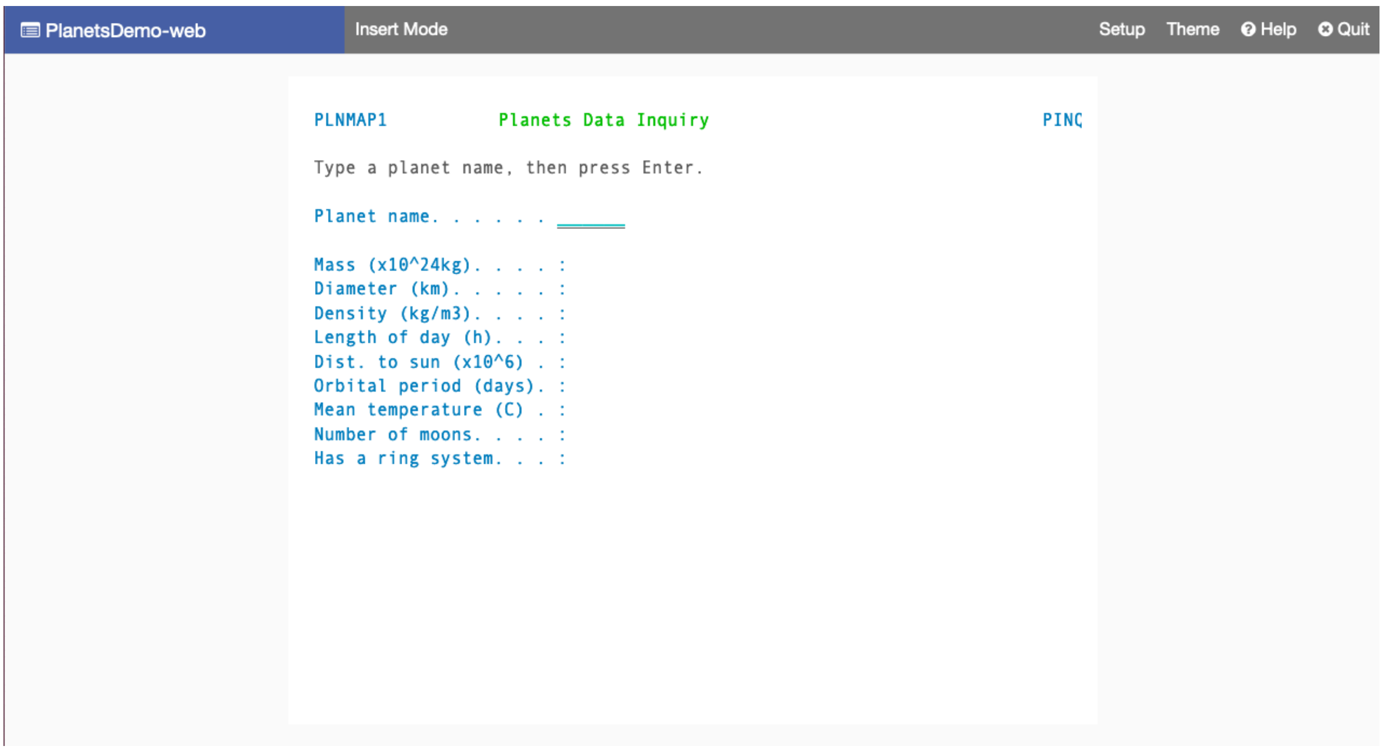
> **Note**
>
> 如果应用程序不可访问，并且端口 8196 上安全组的入站规则选择了“我的 IP”，则指定规则，以允许端口 8196 接受来自 LB i/p 的流量。

步骤 8：测试应用程序

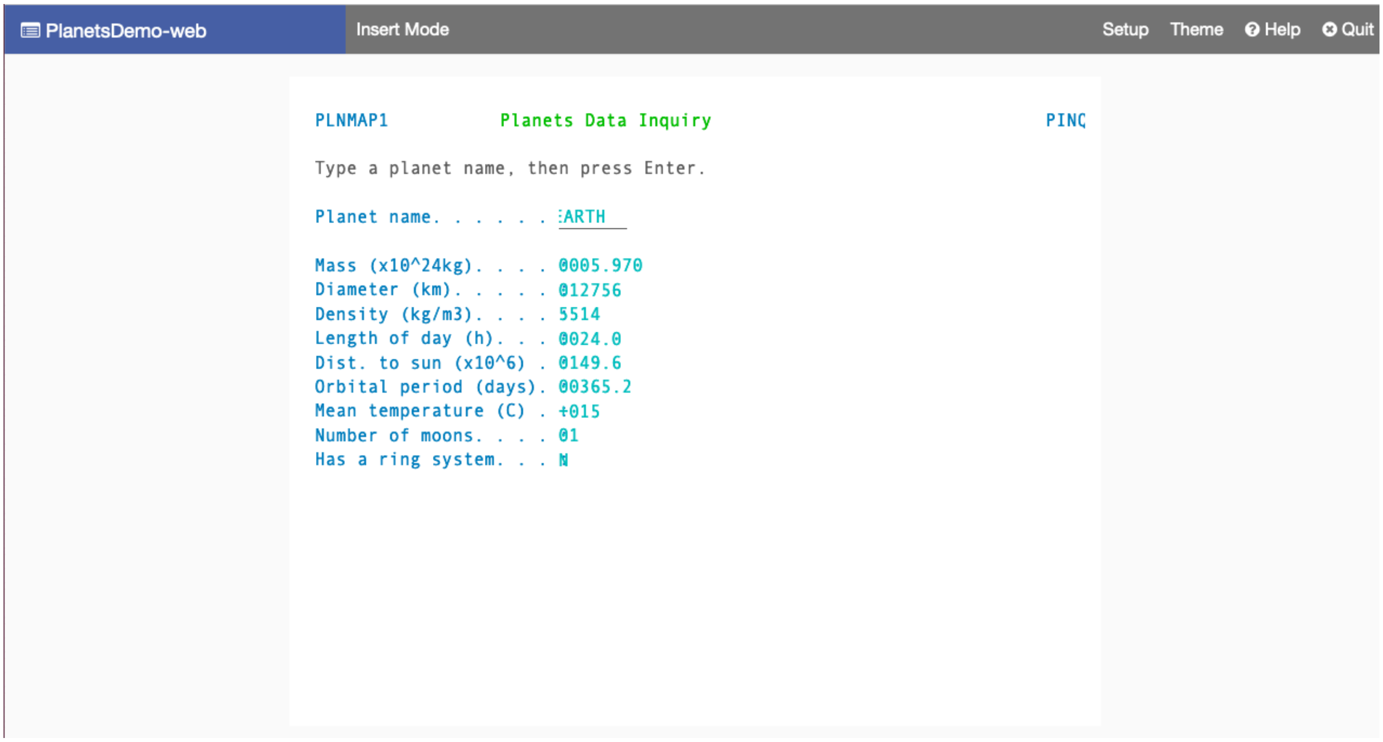
在此步骤中，在迁移的应用程序中运行事务。

1. 在 JICS 屏幕上的输入字段中输入 `PINQ`，然后选择运行（或按 Enter），以便启动应用程序事务。

   随后会显示演示应用程序屏幕。



2. 在相应的字段中键入行星名称，然后按 Enter。



您可以看到有关该行星的详细信息。

清理资源

如果您不再需要为本教程创建的资源，请将其删除，以免产生额外费用。为此，请完成以下步骤：

- 如果 AWS 大型机现代化应用程序仍在运行，请将其停止。
- 删除应用程序。有关更多信息，请参阅 删除 AWS Mainframe Modernization 应用程序。
- 删除运行时环境。有关更多信息，请参阅 删除 AWS 大型机现代化运行时环境。

教程：为 Rocket Software（前身为 Micro Focus）

您可以使用 Rocket Software 运行时引擎在 AWS 大型机现代化托管运行时环境中部署和运行应用程序。本教程介绍如何使用 Rocket Software 运行时引擎在 AWS 大型机现代化托管运行时环境中部署和运行 CardDemo 示例应用程序。 CardDemo 示例应用程序是一个简化的信用卡应用程序，旨在测试 AWS 和展示大型机现代化用例的技术，并与之合作。

在本教程中，您将在 other 中创建资源 AWS 服务。其中包括亚马逊简单存储服务、Amazon Relational Database Service 和 AWS Secrets Manager。 AWS Key Management Service

主题

先决条件

- 确保您可以访问 3270 模拟器以使用 CICS 连接。3270 模拟器可从第三方网站免费试用。或者，你可以启动 AWS 大型机现代化 WorkSpaces 应用程序 Rocket Software 实例并使用 Rumba 3270 模拟器（不免费提供）。

  有关 WorkSpaces 应用程序的信息，请参见the section called “教程：为企业分析器和企业开发人员设置 WorkSpaces 应用程序”。

  > **Note**
  >
  > 创建堆栈时，请选择“Enterprise Developer（ED）”选项，而不是“Enterprise Analyzer（EA）”。

- 下载CardDemo 示例应用程序并将下载的文件解压缩到任意本地目录。该目录将包含一个名为 `CardDemo_runtime` 的子目录。
- 在您的账户中确定一个 VPC，您可以在其中定义本教程中创建的资源。该 VPC 将需要至少两个可用区中的子网。有关 Amazon VPC 的更多信息，请参阅 Amazon VPC 的工作原理。

步骤 1：创建并加载 Amazon S3 存储桶

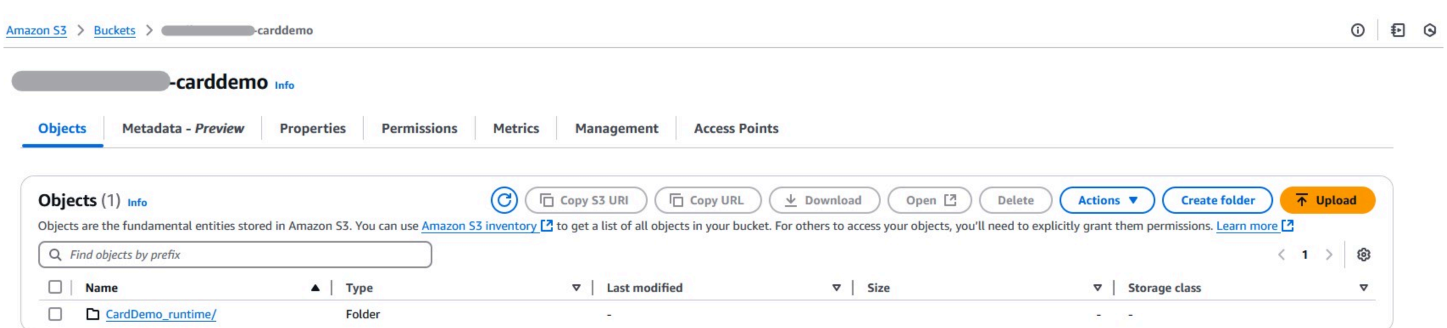
在此步骤中，您将创建一个 Amazon S3 存储桶并将 CardDemo 文件上传到该存储桶。在本教程的后面部分，您将使用这些文件在 AWS 大型机现代化 Rocket 软件托管运行时环境中部署和运行 CardDemo 示例应用程序。

> **Note**
>
> 您不必创建新的 S3 存储桶，但您选择的存储桶必须与本教程中使用的其他资源位于相同的区域中。

创建 Amazon S3 存储桶

1. 打开 Amazon S3 控制台并选择创建存储桶。
2. 在常规配置中，选择要构建 AWS 大型机现代化 Rocket 软件托管运行时的 AWS 区域。
3. 输入存储桶名称，例如 `yourname-aws-region-carddemo`。保留默认设置，然后选择创建存储桶。或者，您也可以从现有 Amazon S3 存储桶复制设置，然后选择创建存储桶。
4. 选择您刚刚创建的存储桶，然后选择上传。
5. 在上传部分中，选择添加文件夹，然后在本地计算机中浏览到 `CardDemo_runtime` 目录。
6. 选择上传以开始上传过程。上传时间因您的连接速度而异。
7. 上传完成后，确认所有文件都已上传成功，然后选择关闭。

您的 Amazon S3 存储桶中现在包含 `CardDemo_runtime` 文件夹。



有关 S3 存储桶的信息，请参阅创建、配置和使用 Amazon S3 存储桶。

步骤 2：创建和配置数据库

在此步骤中，您将在 Amazon Relational Database Service（Amazon RDS）中创建一个 PostgreSQL 数据库。在本教程中，该数据库包含 CardDemo 示例应用程序用于客户执行信用卡交易任务的数据集。

在 Amazon RDS 中创建数据库

1. 打开 [Amazon RDS 控制台](https://console.aws.amazon.com/rds/)。
2. 选择要在其中创建数据库实例的 AWS 区域。
3. 从导航窗格中选择数据库。
4. 选择创建数据库，然后选择标准创建。
5. 对于引擎类型，选择 PostgreSQL。
6. 选择 15 或更高的引擎版本。

   > **Note**
   > 保存该引擎版本，因为本教程后面部分需要用到它。

7. 在模板部分中，选择免费套餐。
8. 将数据库实例标识符更改为有意义的标识符，例如 `MicroFocus-Tutorial`。
9. 请勿在 AWS Secrets Manager中管理主凭证。请输入主密码并确认该密码。

   > **Note**
   > 保存您用于数据库的用户名和密码。在本教程的后续步骤中，您将安全地存储这些信息。

10. 在 “连接” 下，选择要在其中创建 AWS 大型机现代化托管运行时环境的 VPC。
11. 选择创建数据库。

在 Amazon RDS 中创建自定义参数组

1. 在 Amazon RDS 控制台的导航窗格中，选择参数组，然后选择创建参数组。
2. 在创建参数组窗口中，对于参数组系列，选择与您的数据库版本匹配的 Postgres 选项。

> **Note**
>
> 某些 Postgres 版本需要指定类型。如果需要，请选择数据库参数组。为参数组输入组名称和描述。

3. 选择创建。

配置自定义参数组

1. 选择新创建的参数组。
2. 选择操作，然后选择编辑。
3. 按 `max_prepared_transactions` 进行筛选并将参数值更改为 100。
4. 选择保存更改。

将自定义参数组与数据库关联

1. 在 Amazon RDS 控制台的导航窗格中，选择数据库，然后选择要修改的数据库实例。
2. 选择修改。将显示修改数据库实例页面。

> **Note**
>
> 只有在数据库完成创建和备份（这可能需要几分钟时间）之后，修改选项才可用。

3. 在修改数据库实例页面上，导航到其他配置，然后将数据库参数组更改为您的参数组。如果列表中没有您的参数组，请检查该参数组是否使用正确的数据库版本创建。
4. 选择继续，然后查看修改摘要。
5. 选择立即应用以立即应用更改。
6. 选择修改数据库实例以保存更改。

有关参数组的更多信息，请参阅使用参数组。

> **Note**
>
> 您也可以将 Amazon Aurora PostgreSQL 数据库 AWS 与大型机现代化配合使用，但没有免费套餐选项。有关更多信息，请参阅使用 Amazon Aurora postgreSQL。

步骤 3：创建并配置 AWS KMS key

要安全地存储 Amazon RDS 实例的凭证，请先创建一个 AWS KMS key。

要创建 AWS KMS key

1. 打开 Key Management Service 控制台。
2. 选择创建密钥。
3. 对于密钥类型，保留对称默认值，对于密钥用途，保留加密和解密默认值。
4. 选择下一步。
5. 为该密钥指定一个别名（例如 `MicroFocus-Tutorial-RDS-Key`）和一个描述（可选）。
6. 选择下一步。
7. 通过选中您的用户或角色旁边的复选框来分配密钥管理员。
8. 选择下一步。
9. 通过选中您的用户或角色旁边的复选框来分配密钥使用权限。
10. 选择下一步。
11. 在查看屏幕上，编辑密钥策略，然后在现有的“声明”数组中输入以下内容：

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

    此策略使用此特定密钥策略授予 AWS 大型机现代化解密权限。
12. 选择完成以创建密钥。

有关更多信息，请参阅《 AWS Key Management Service 开发人员指南》中的创建密钥。

步骤 4：创建和配置 AWS Secrets Manager 数据库密钥

现在，使用 AWS Secrets Manager 和安全地存储数据库凭据 AWS KMS key。

创建和配置 AWS Secrets Manager 数据库密钥

1. 打开 Secrets Manager 控制台。
2. 在导航窗格中，选择密钥。
3. 在密钥中，选择存储新密钥。
4. 将密钥类型设置为 Amazon RDS 数据库的凭证。
5. 输入您在创建数据库时指定的凭证。
6. 在加密密钥下，选择您在步骤 3 中创建的密钥。
7. 在数据库部分中，选择您为本教程创建的数据库，然后选择下一步。
8. 在密钥名称下，输入名称（例如 `MicroFocus-Tutorial-RDS-Secret`）和描述（可选）。
9. 在资源权限部分，选择编辑权限，然后将内容替换为以下策略：

   JSON

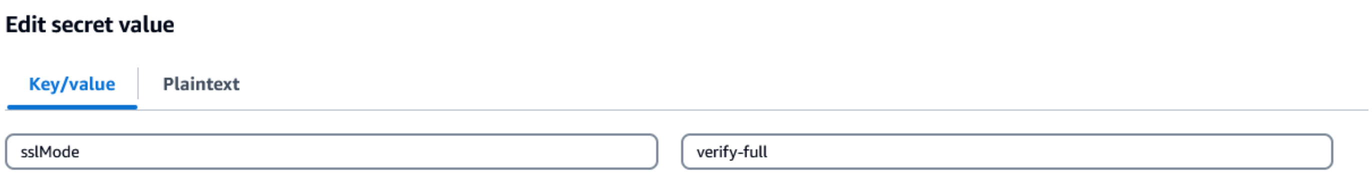
```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect" : "Allow",
            "Principal" : {
                "Service" : "m2.amazonaws.com"
            },
            "Action" : "secretsmanager:GetSecretValue",
            "Resource" : "*"
        }
    ]
}
```

10. 选择保存。
11. 在后续屏幕中选择下一步，然后选择存储。

第 5 步：将 SSLMode 添加到密钥中

将 SSLMode 添加到密钥中

1. 刷新密钥列表以查看新密钥。
2. 在步骤 4 中选择新创建的密钥，并记下，`Secret ARN`因为你需要在本教程的后面部分使用它。
3. 在密钥的概述选项卡中，选择检索密钥值。
4. 选择编辑，然后选择添加行。
5. 添加值为 `verify-full` 的 `sslMode` 密钥：

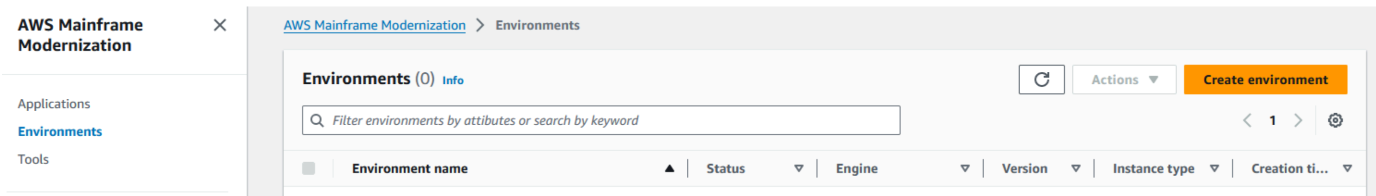


6. 选择保存。

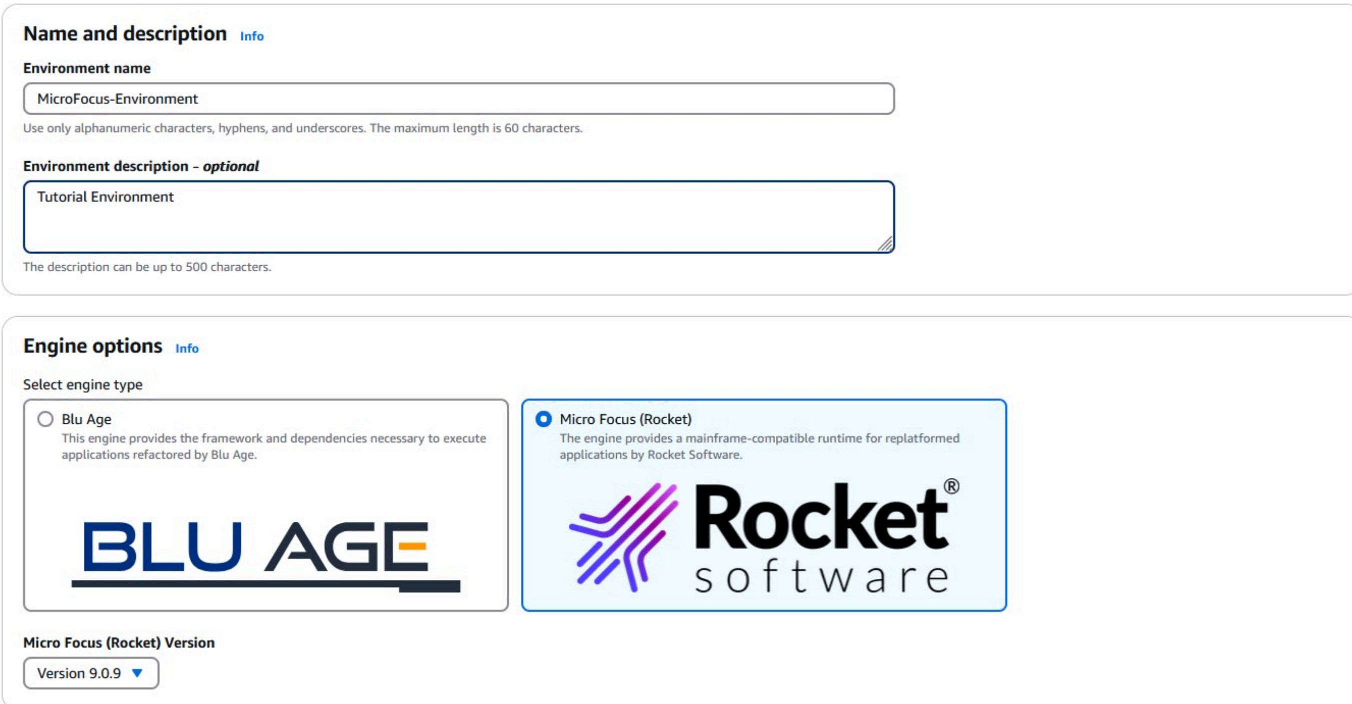
步骤 6：创建运行时环境

创建运行时环境

1. 使用 [AWS Mainframe Modernization 控制台](https://console.aws.amazon.com/m2/)。
2. 在导航窗格中，选择环境。然后，选择创建环境。



3. 在指定基本信息下，
   a. 为“环境名称”输入 `MicroFocus-Environment`。
   b. 在引擎选项下，确保选择了 Micro Focus（火箭）。
   c. 选择最新的 Micro Focus（火箭）版本。
   d. 选择下一步。



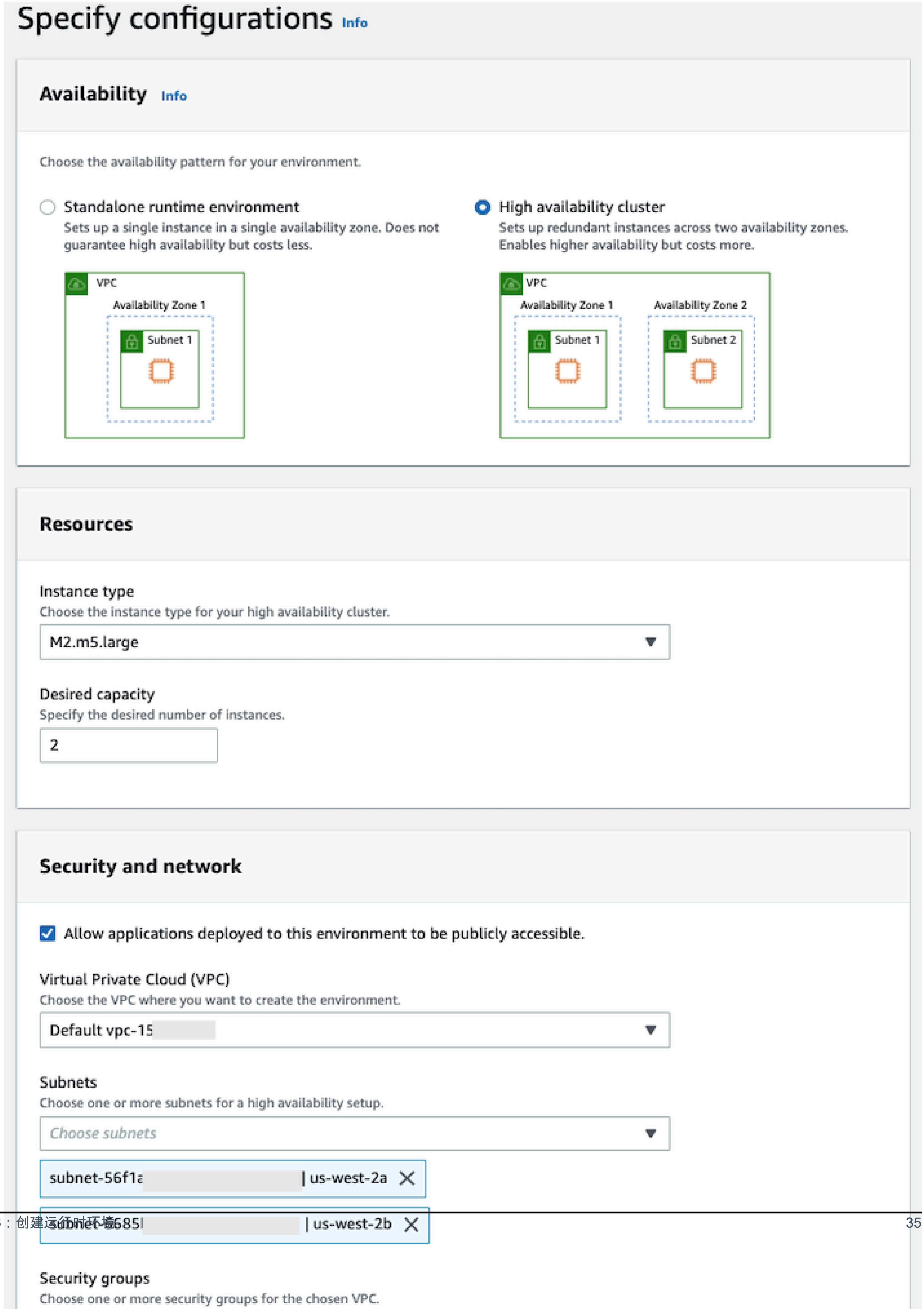
4. 配置环境

   a. 在可用性下，选择高可用性集群。

   b. 在资源下，选择 M2.c5.large 或 M2.m5.large 作为实例类型，并选择所需的实例数量。最多可以指定两个实例。

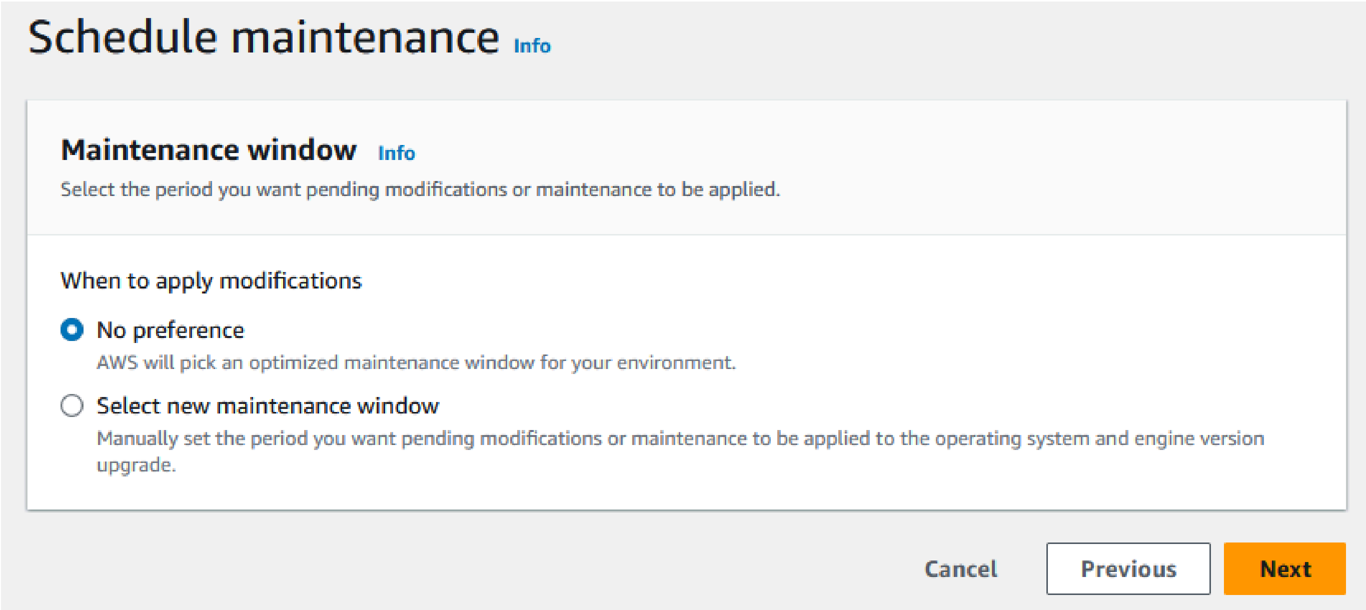
   c. 在安全和网络下，选择支持公开访问部署到此环境的应用程序，并至少选择两个公有子网。

   d. 选择下一步。

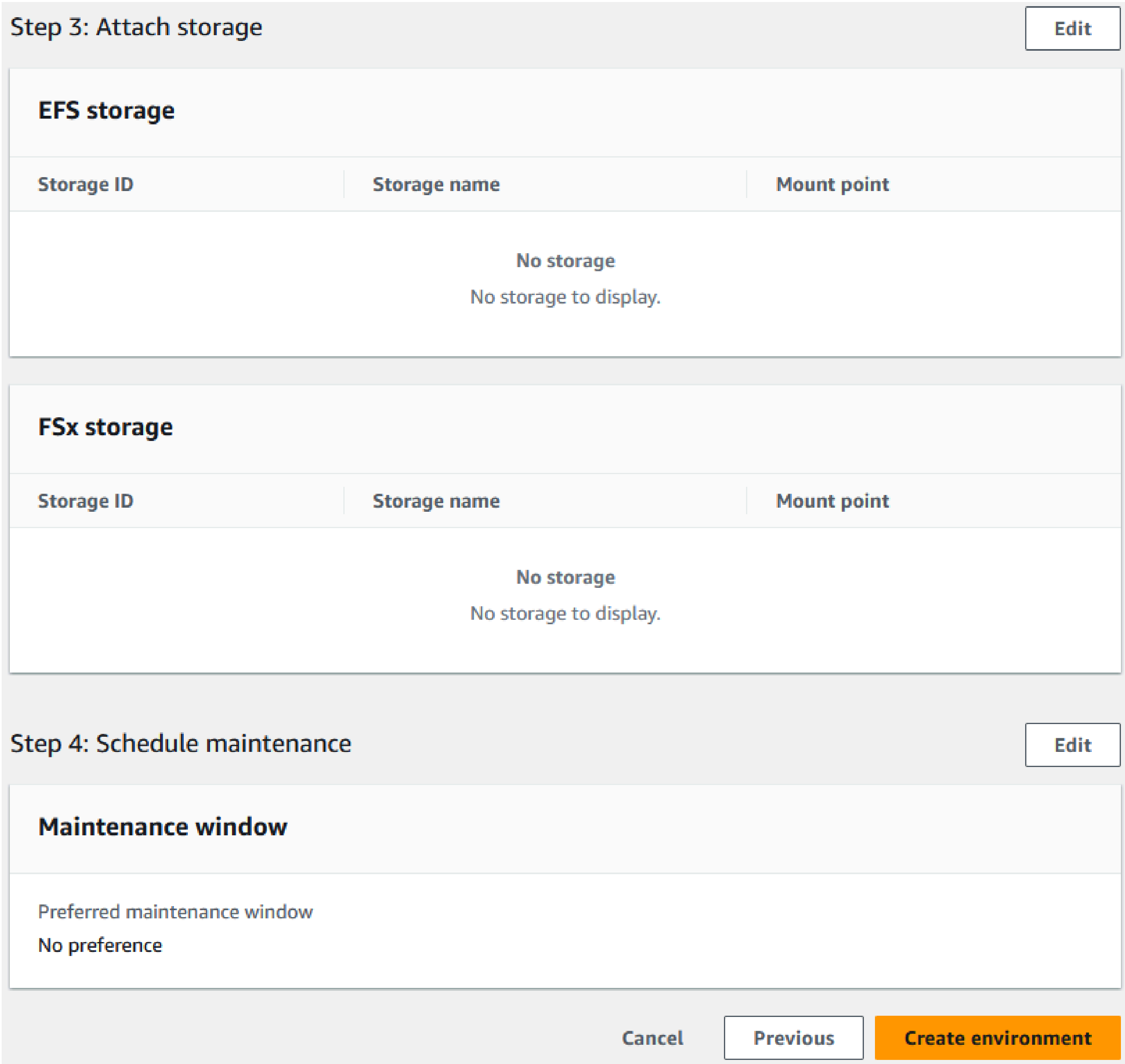


Specify configurations Info
Availability Info
Choose the availability pattern for your environment.
Standalone runtime environment
Sets up a single instance in a single availability zone. Does not guarantee high availability but costs less.
VPC
Availability Zone 1
Subnet 1
High availability cluster
Sets up redundant instances across two availability zones. Enables higher availability but costs more.
VPC
Availability Zone 1
Availability Zone 2
Subnet 1
Subnet 2
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
| us-west-2b
Security groups
Choose one or more security groups for the chosen VPC.

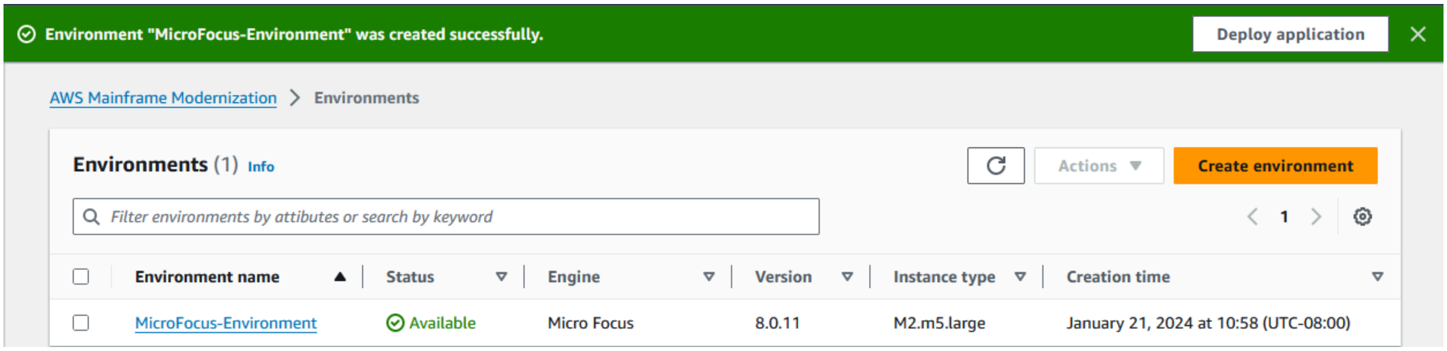
5. 在附加策略页面上，选择下一步。
6. 在计划维护页面上，选择无首选项，然后选择下一步。



7. 在审核和创建页面上，审核您为运行时环境提供的所有配置，然后选择创建环境。



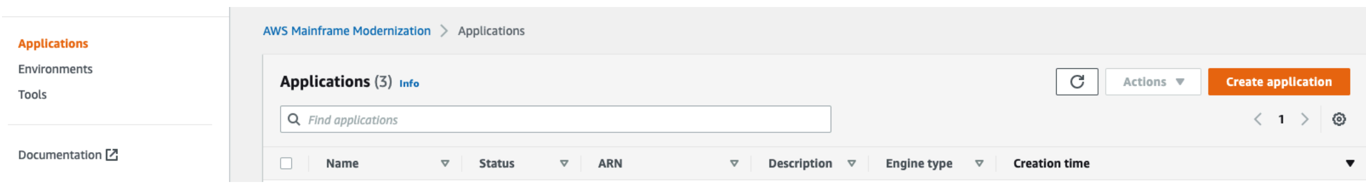
创建环境后，会出现一个横幅，显示`Environment` *name* `was created successfully`，并且状态字段更改为 可用。环境创建过程需要几分钟，但您可以在此过程运行时继续执行后续步骤。



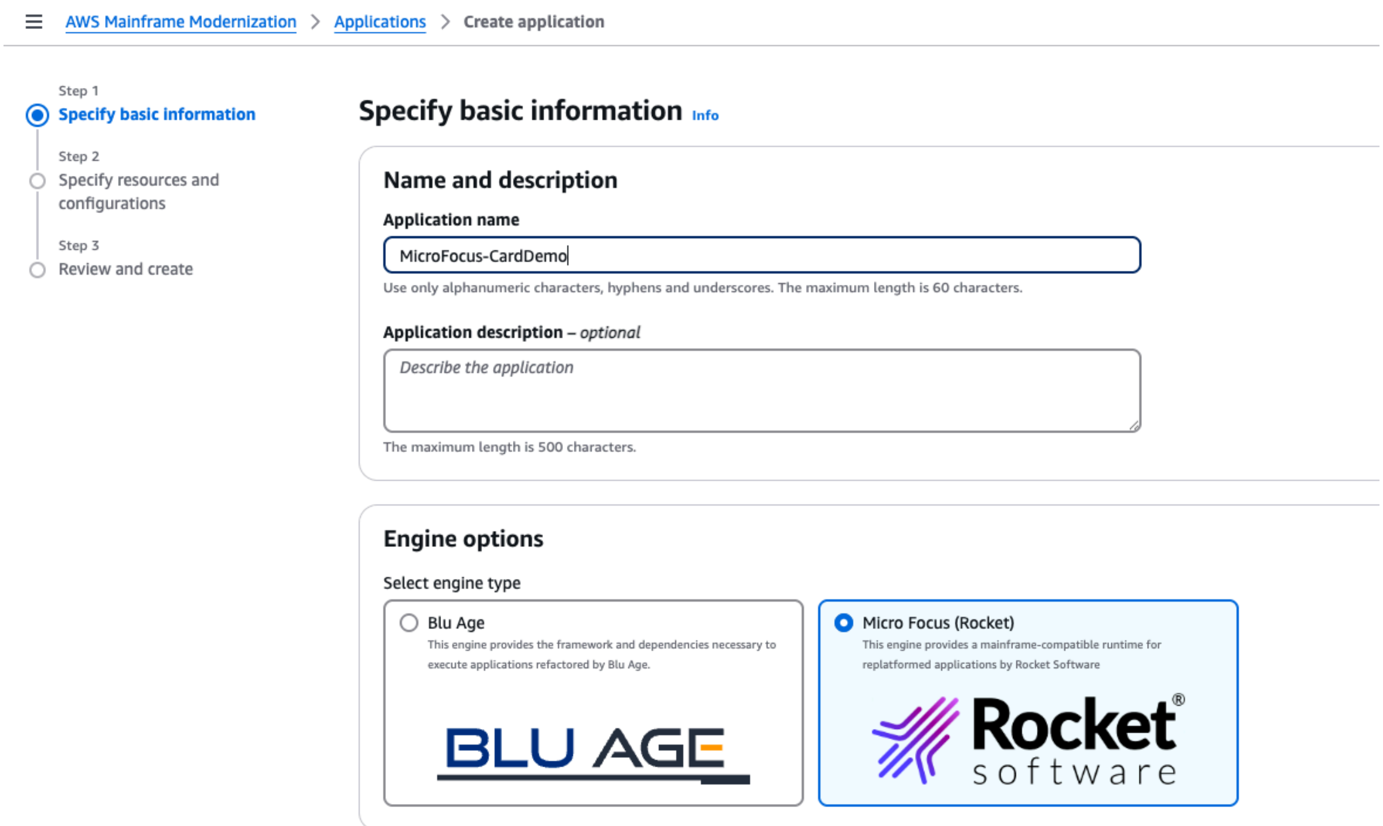
步骤 7：创建应用程序

创建应用程序

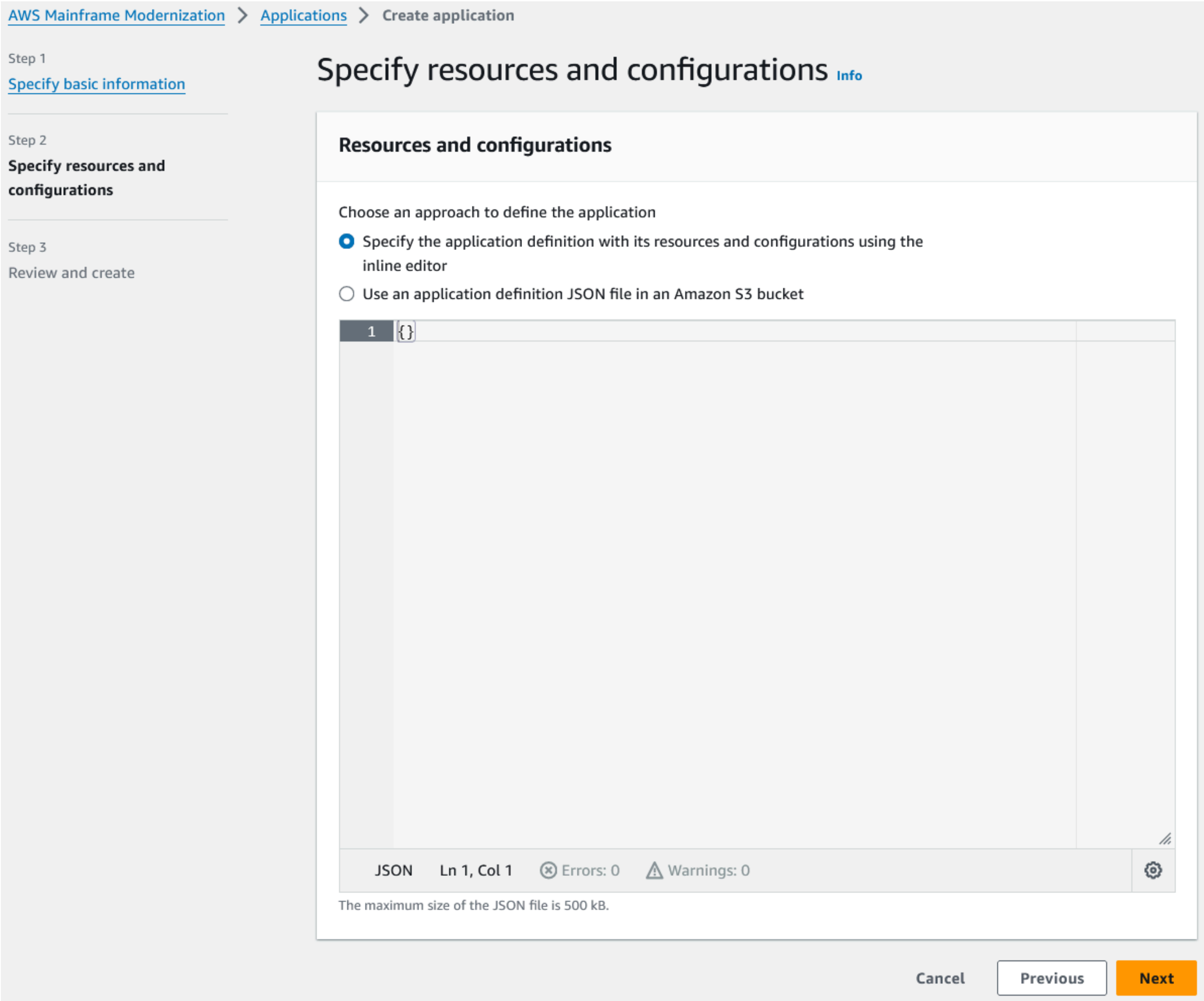
1. 在导航窗格中，选择应用程序。选择创建应用程序。



2. 在 “创建应用程序” 页面的 “指定基本信息” 下，输入应用程序名称，然后在 `MicroFocus-CardDemo` “引擎类型” 下确保选中 Micro Focus（Rocket）。然后选择下一步。



3. 在指定资源和配置下，选择“使用内联编辑器指定应用程序定义及其资源和配置”选项。



在编辑器中输入以下应用程序定义：

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

```
  ],
  "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"
        }
      ]
    }
}
```

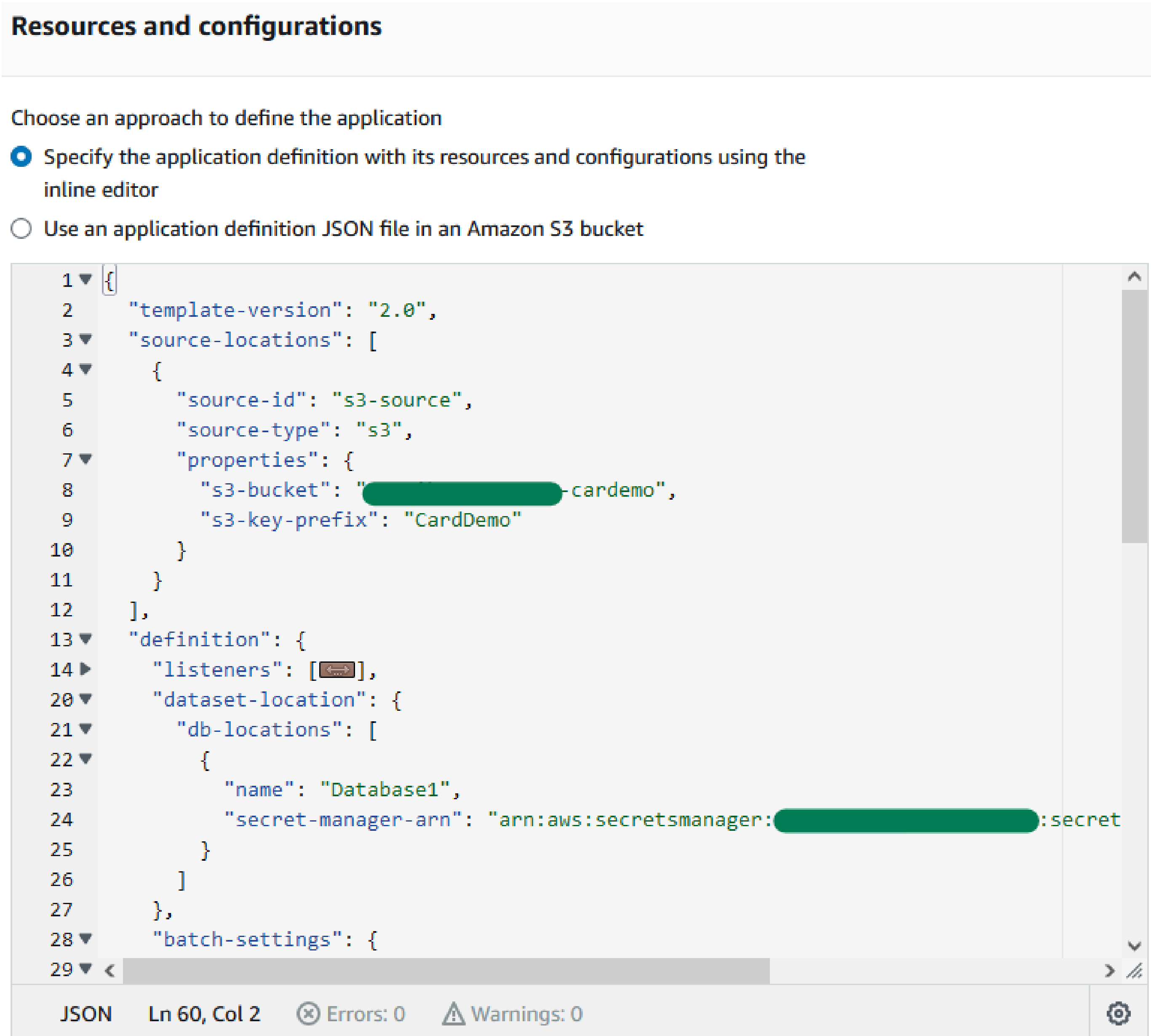
> **Note**
>
> 此文件可能随时更改。

4. 在 source-locations 的 properties 对象中编辑应用程序 JSON，如下所示：

   a. 将 `s3_bucket` 的值替换为在步骤 1 中创建的 Amazon S3 存储桶的名称。

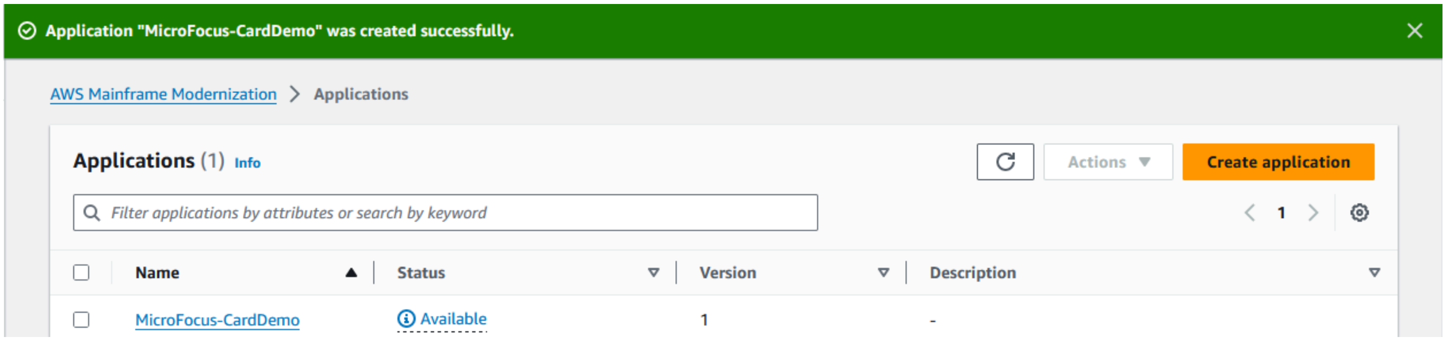
   b. 将的值替换为您上传 CardDemo 示例文件的文件夹（key prefix）。`s3-key-prefix`如果您将 `CardDemo` 目录直接上传到 Amazon S3 存储桶，则无需更改 `s3-key-prefix`。

   c. 将两个 `secret-manager-arn` 值替换为您在步骤 4 中创建的数据库密钥的 ARN。



有关应用程序定义的更多信息，请参阅 [火箭软件（前身为 Micro Focus）应用程序定义](#)。

5. 选择下一步以继续。
6. 在审核和创建页面上，检查您提供的信息，然后选择创建应用程序。

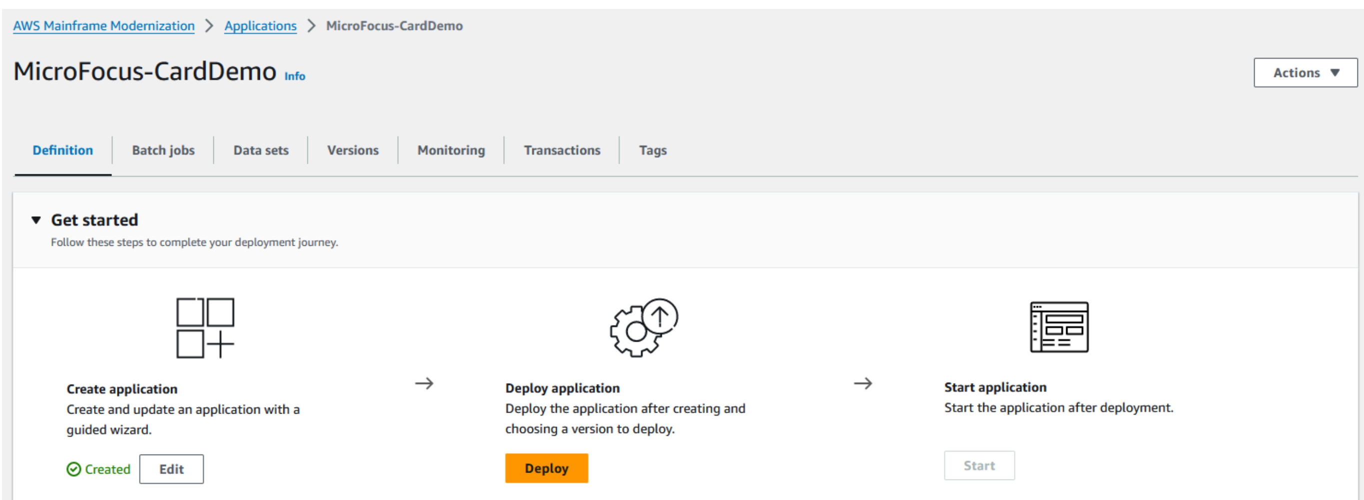


创建应用程序后，会出现一条横幅，显示Application *name* was created successfully。状态字段会变为可用。

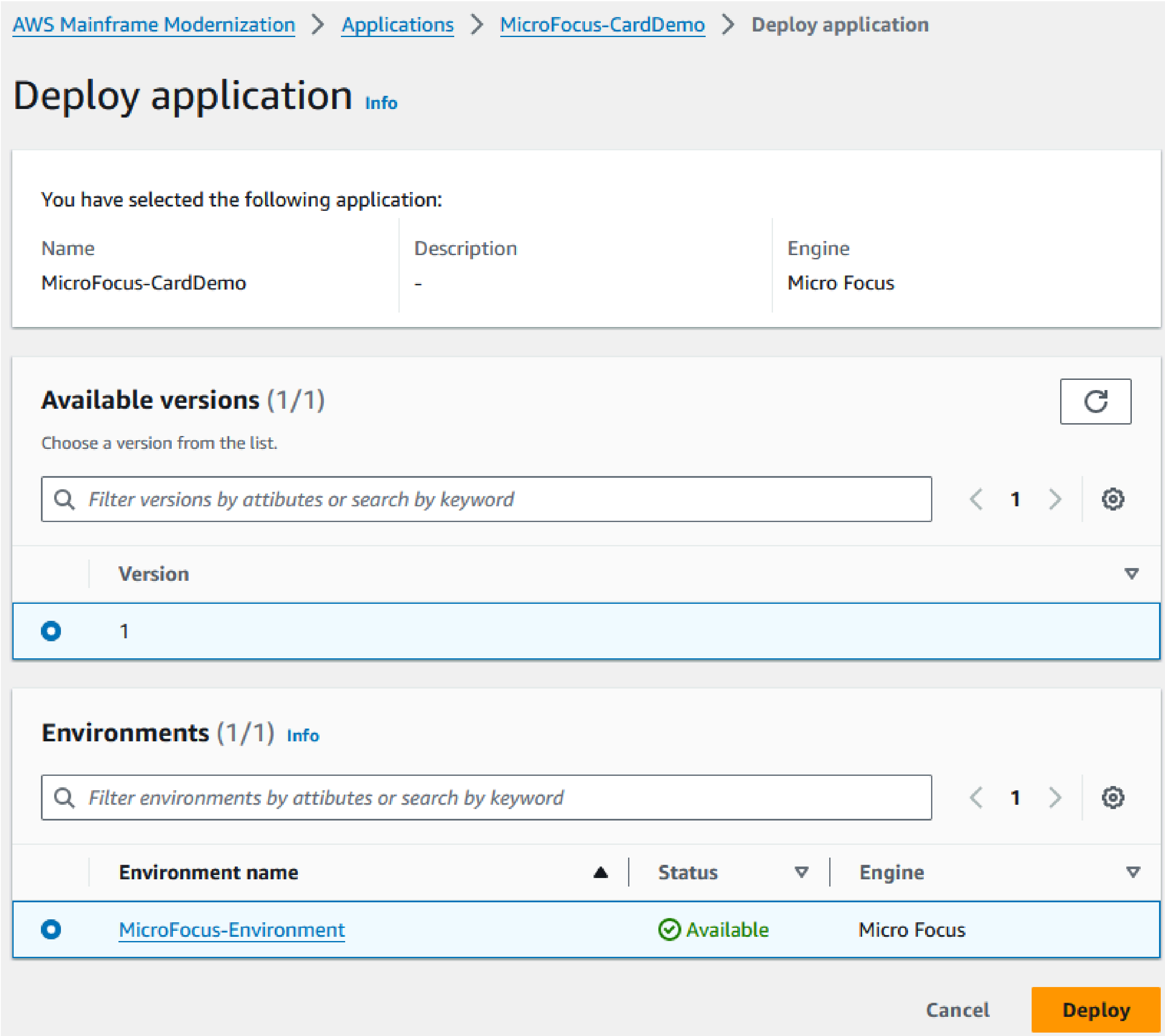
步骤 8：部署应用程序

部署应用程序

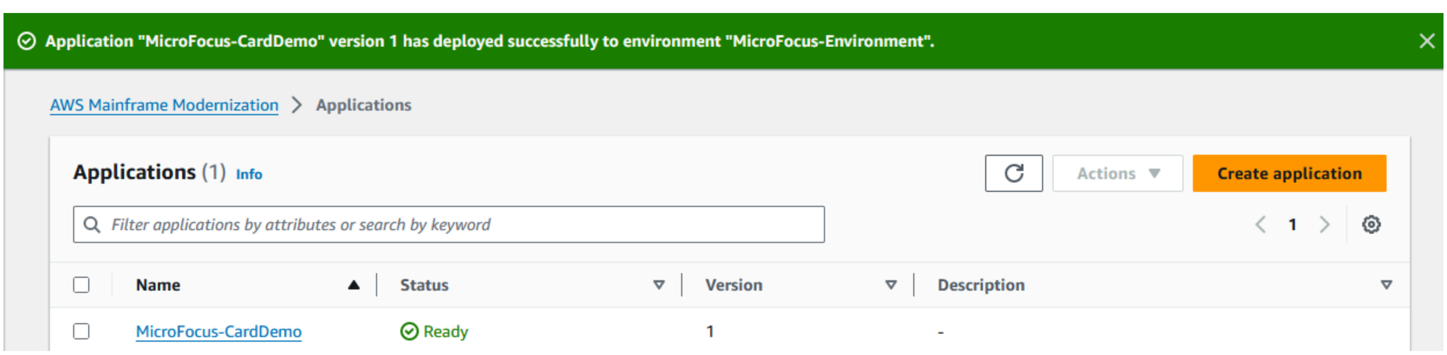
1. 在导航窗格中，选择应用程序，然后选择 MicroFocus-CardDemo。
2. 在部署应用程序下，选择部署。



3. 选择您之前创建的应用程序和环境的最新版本，然后选择部署。



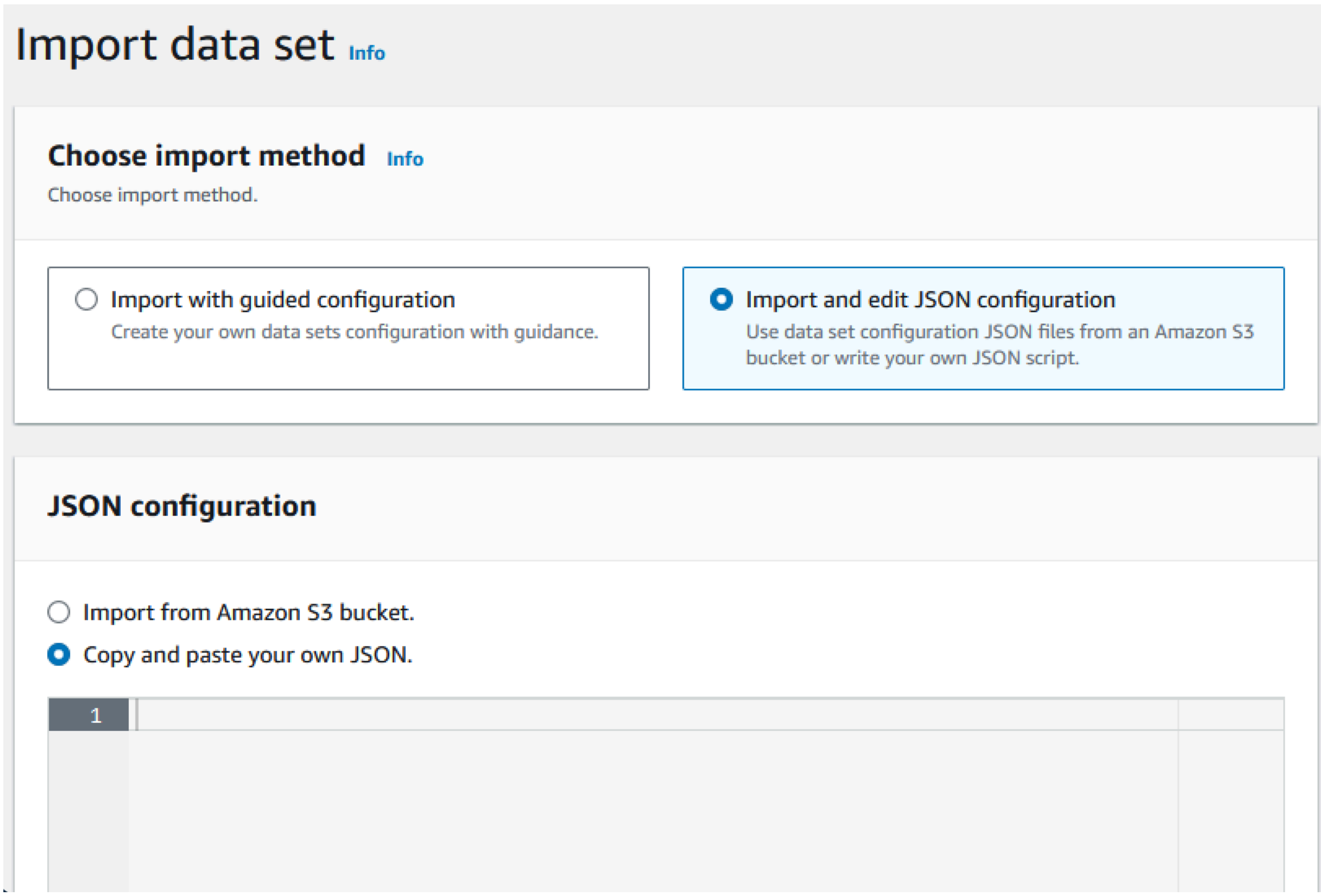
成功部署 CardDemo 应用程序后，状态将更改为 “就绪”。



步骤 9：导入数据集

导入数据集

1. 在导航窗格中，选择应用程序，然后选择应用程序。
2. 选择数据集选项卡。然后选择导入。
3. 选择导入和编辑 JSON 配置，然后选择复制并粘贴您自己的 JSON 选项。



4. 复制并粘贴以下 JSON，但暂不选择“提交”。此 JSON 包含演示应用程序所需的所有数据集，但需要您的 Amazon S3 存储桶详细信息。

```
{
    "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_runtime/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_runtime/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_runtime/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_runtime/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_runtime/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_runtime/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_runtime/catalog/data/
AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS.DAT"
        }
    },
    {
        "dataSet": {
            "storageType": "Database",
            "datasetName": "AWS.M2.CARDDEMO.USRSEC.VSAM.KSDS",
            "relativePath": "DATA",
            "datasetOrg": {
                "vsam": {
```

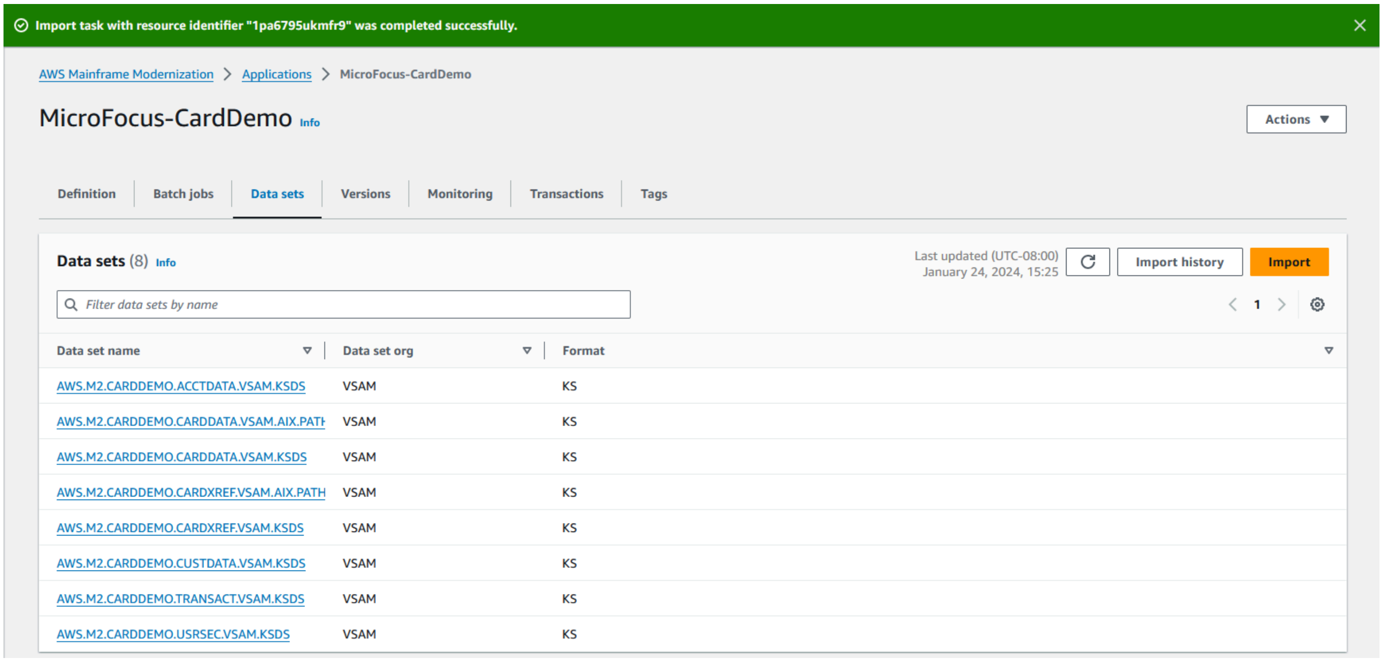
```
                        "format": "KS",
                        "encoding": "A",
                        "primaryKey": {
                            "length": 8,
                            "offset": 0
                        }
                    }
                },
                "recordLength":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

5. 将每次出现的`<s3-bucket-name>`（有八个）替换为包含该 CardDemo 文件夹的 Amazon S3 存储桶的名称，例如`your-name-aws-region-carddemo`。

> **Note**
>
> 要复制 Amazon S3 中该文件夹的 Amazon S3 URI，请选择该文件夹，然后选择复制 Amazon S3 URI。

6. 选择提交。

   导入完成后，将出现一条横幅，显示以下消息：`Import task with resource identifier` *`name`* `was completed successfully.` 将显示导入的数据集的列表。

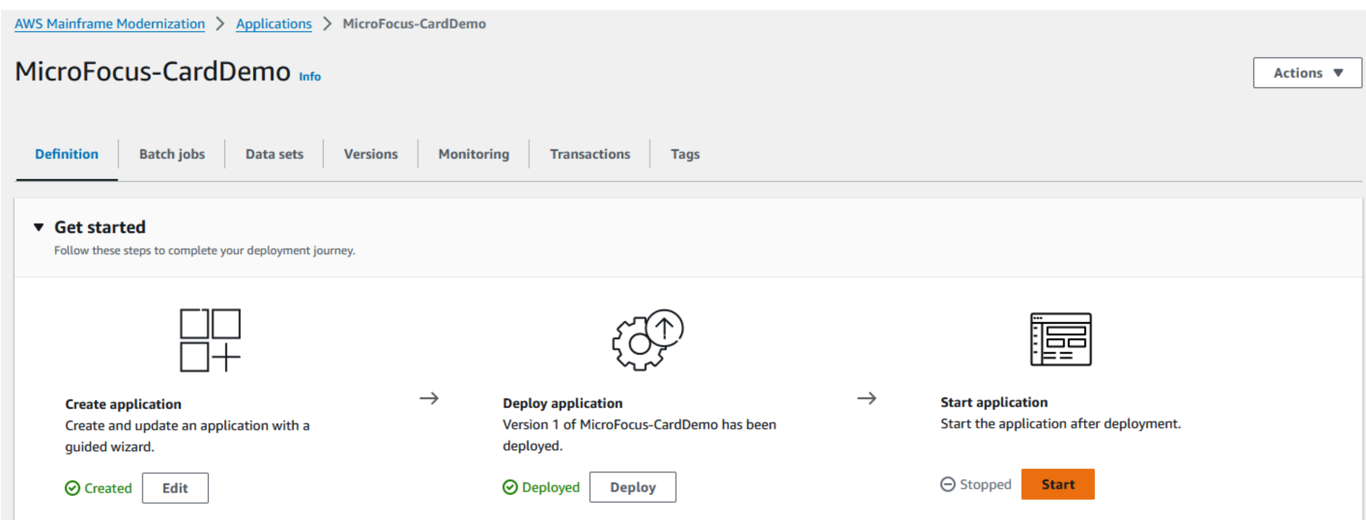


您还可以通过选择数据集选项卡上的导入历史记录来查看所有数据集导入的状态。

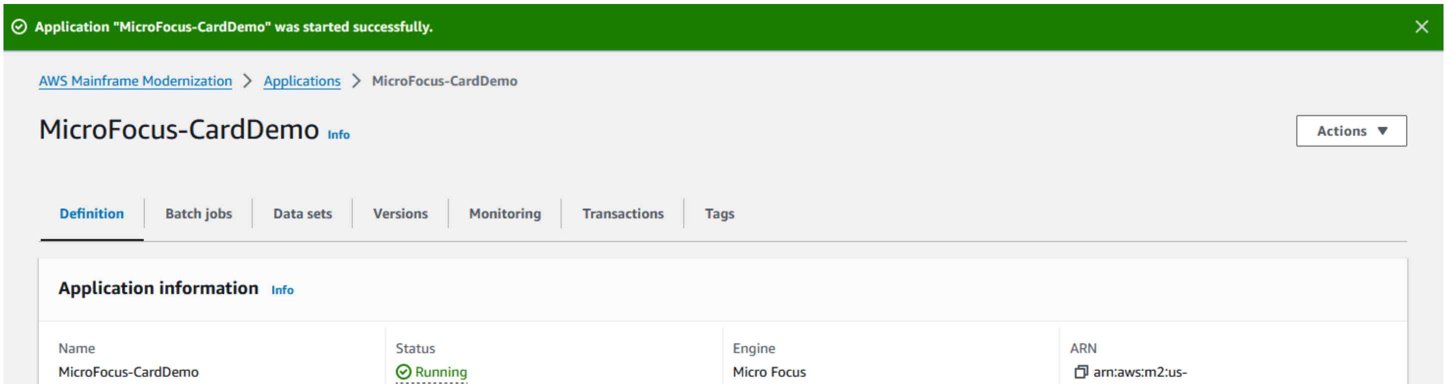
步骤 10：启动应用程序

启动应用程序

1. 在导航窗格中，选择应用程序，然后选择应用程序。
2. 选择启动应用程序。



当 CardDemo 应用程序开始成功运行时，会出现一条横幅，上面写着以下消息：`Application` *name* `was started successfully`。状态字段会变为正在运行。



步骤 11：连接到 CardDemo CICS 应用程序

在连接之前，请确保您为应用程序指定的 VPC 和安全组与应用于您将连接的网络接口的 VPC 和安全组相同。

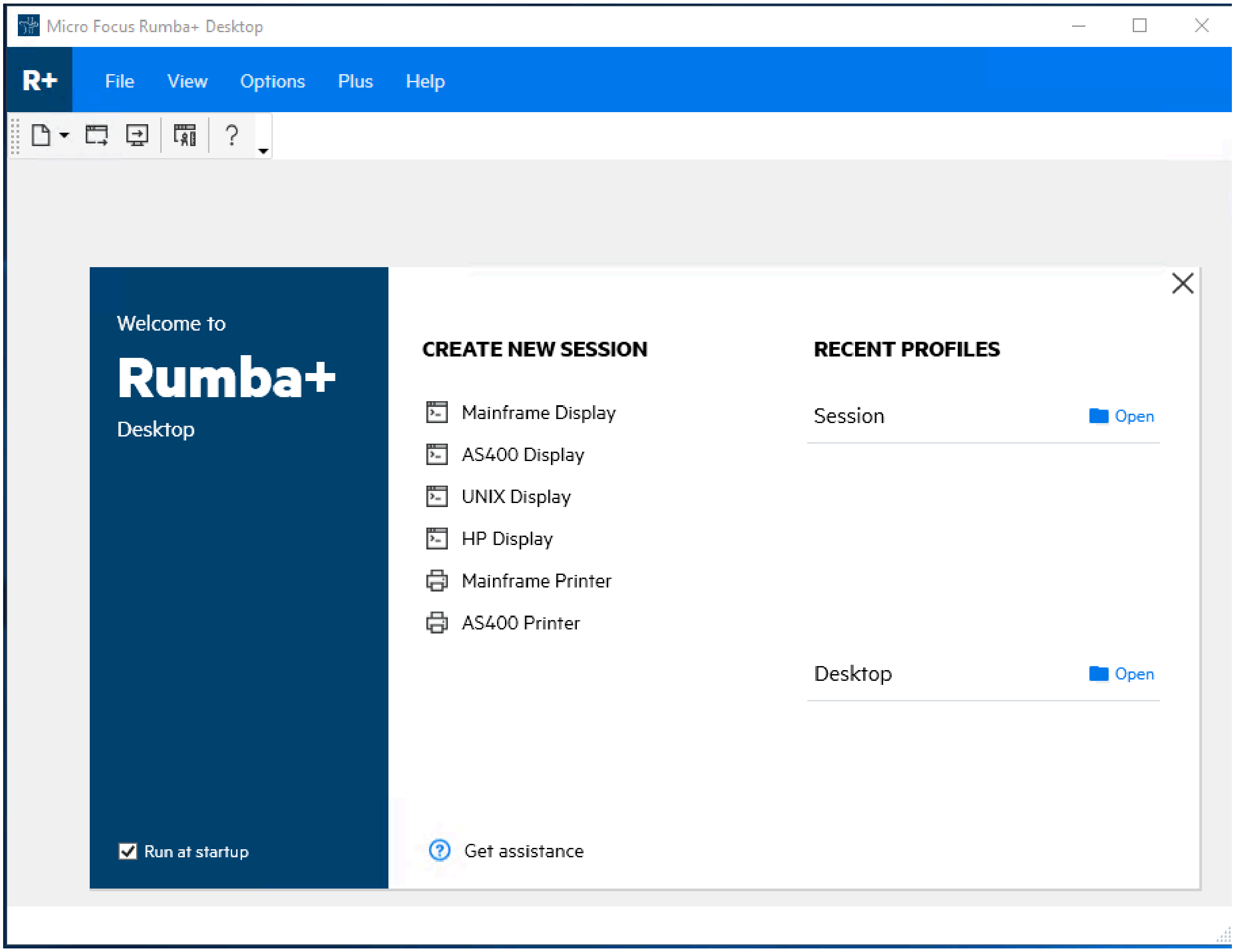
要配置 TN327 0 连接，还需要应用程序的 DNS 主机名和端口。

使用终端模拟器配置应用程序并将其连接到大型机

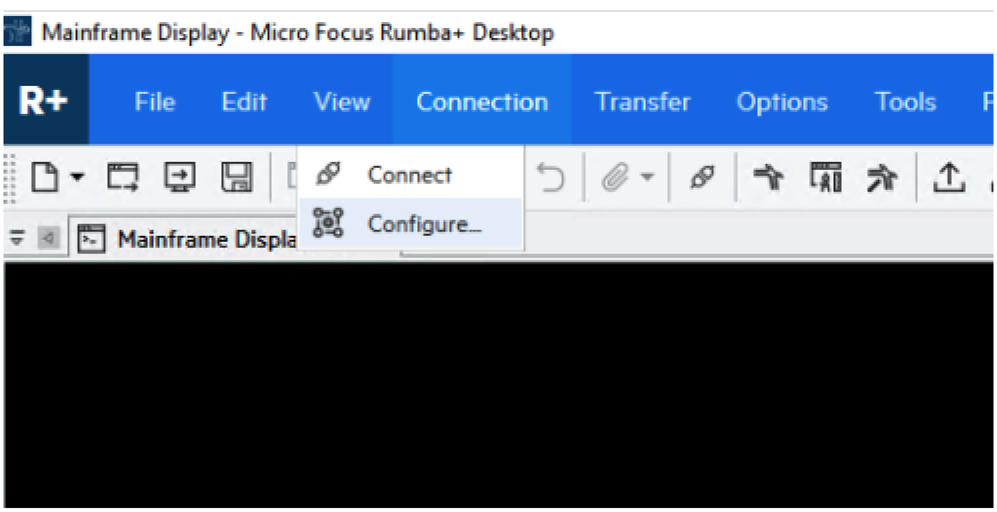
1. 打开 AWS 大型机现代化控制台并选择 “应用程序”，然后选择`MicroFocus-CardDemo`。
2. 选择复制图标来复制 DNS 主机名。另外，请务必记下端口号。
3. 启动终端模拟器。本教程使用 Micro Focus Rumba+。

> **Note**
>
> 配置步骤因模拟器而异。

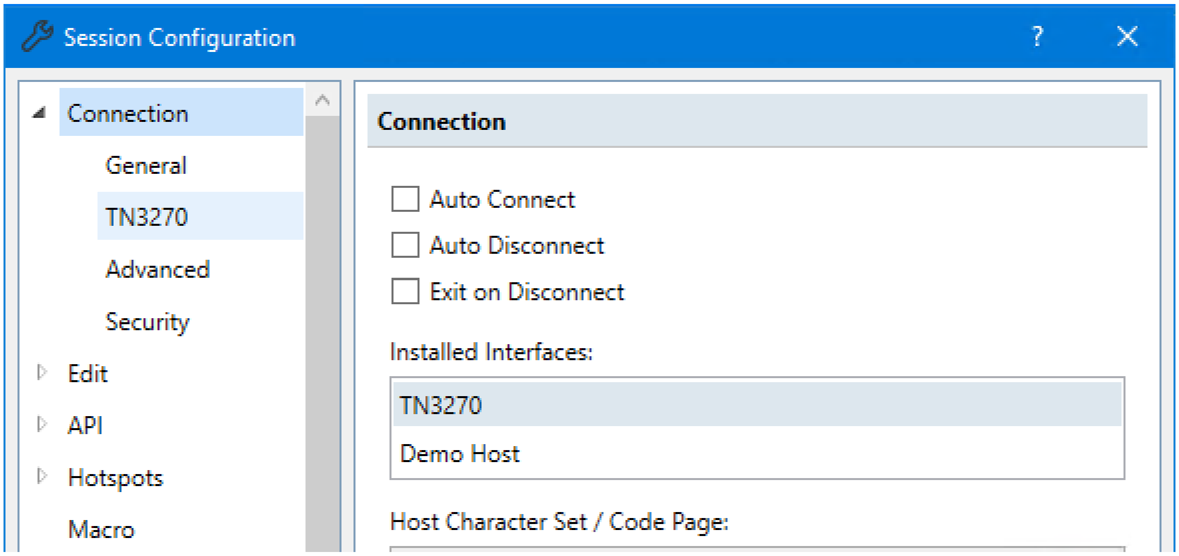
4. 选择大型机显示。



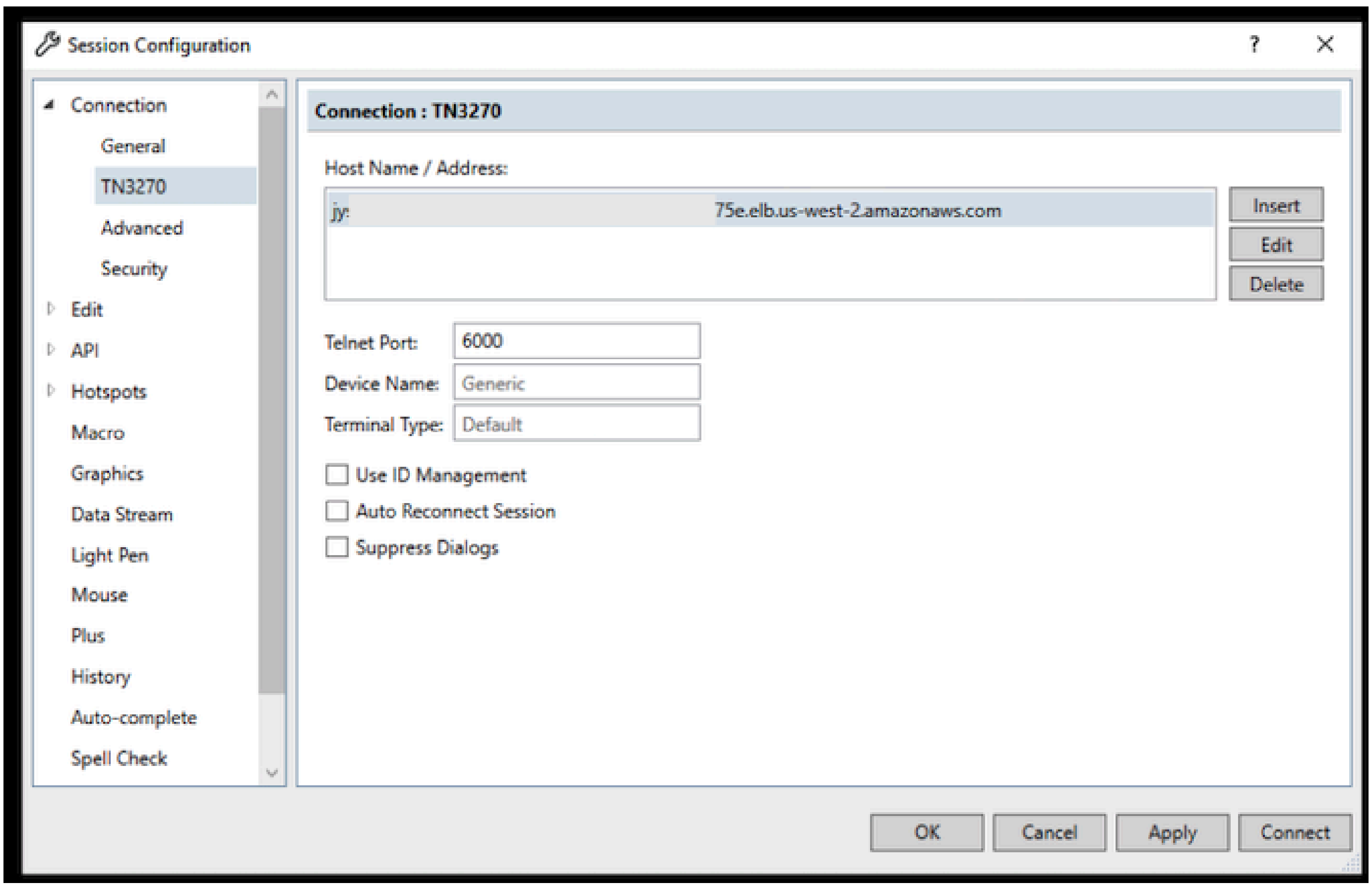
5. 选择连接，然后选择配置。



6. 在已安装的接口下，选择 TN3270，然后在连接菜单下再次选择 TN3270。



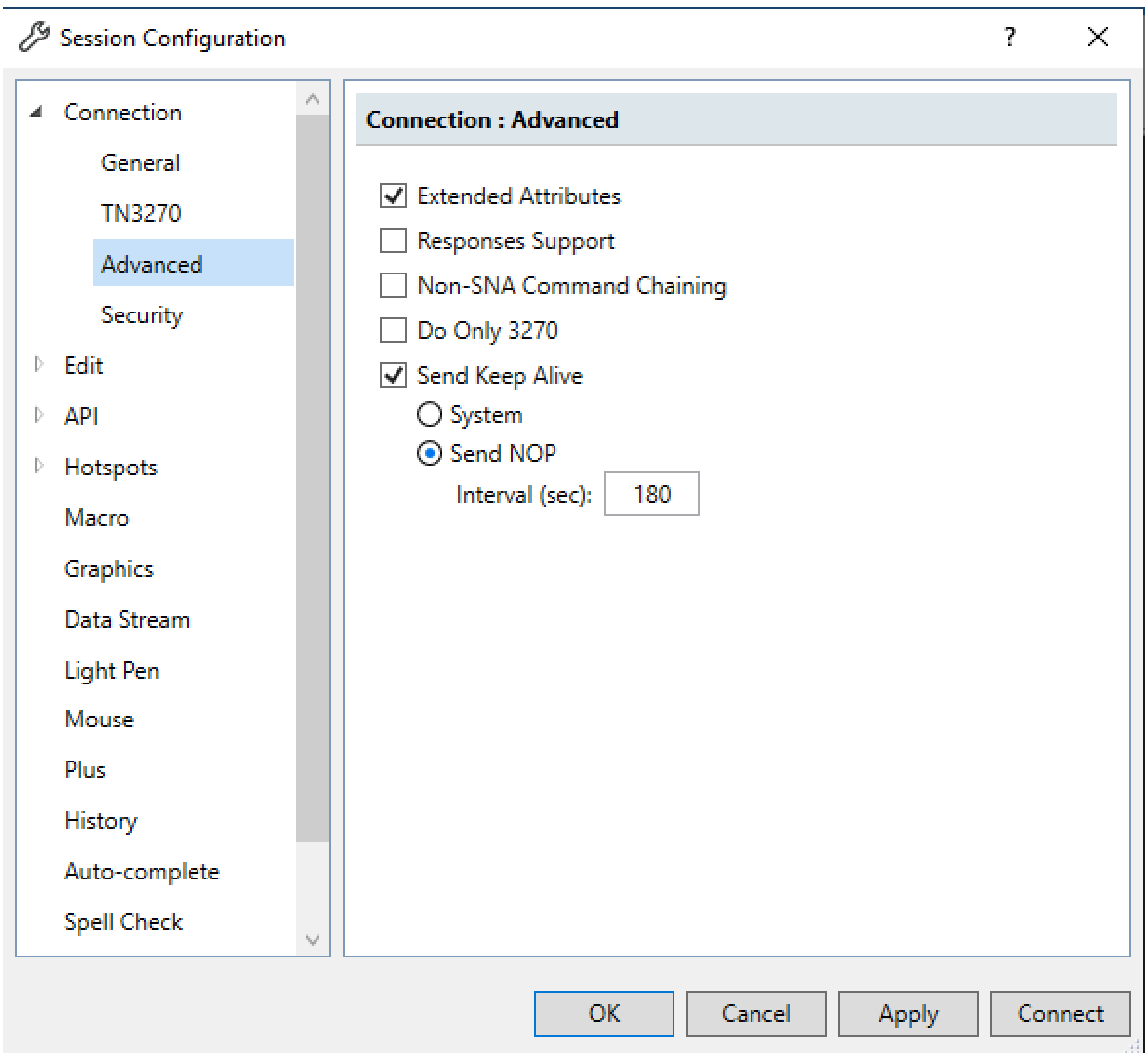
7. 选择插入，然后粘贴应用程序的 `DNS Hostname`。为 Telnet 端口指定 `6000`。



> **Note**
>
> 如果您在浏览器中使用 AWS AppStream 2.0，并且在粘贴值时遇到困难，请参阅 AppStream 2.0 用户问题疑难解答。

8. 在连接下，选择高级，然后选择发送保持活动和发送 NOP，然后为间隔输入 180。

> **Note**
>
> 将 TN327 0 终端上的保持活动状态设置配置为至少 180 秒有助于确保 Network Load Balancer 不会中断您的连接。

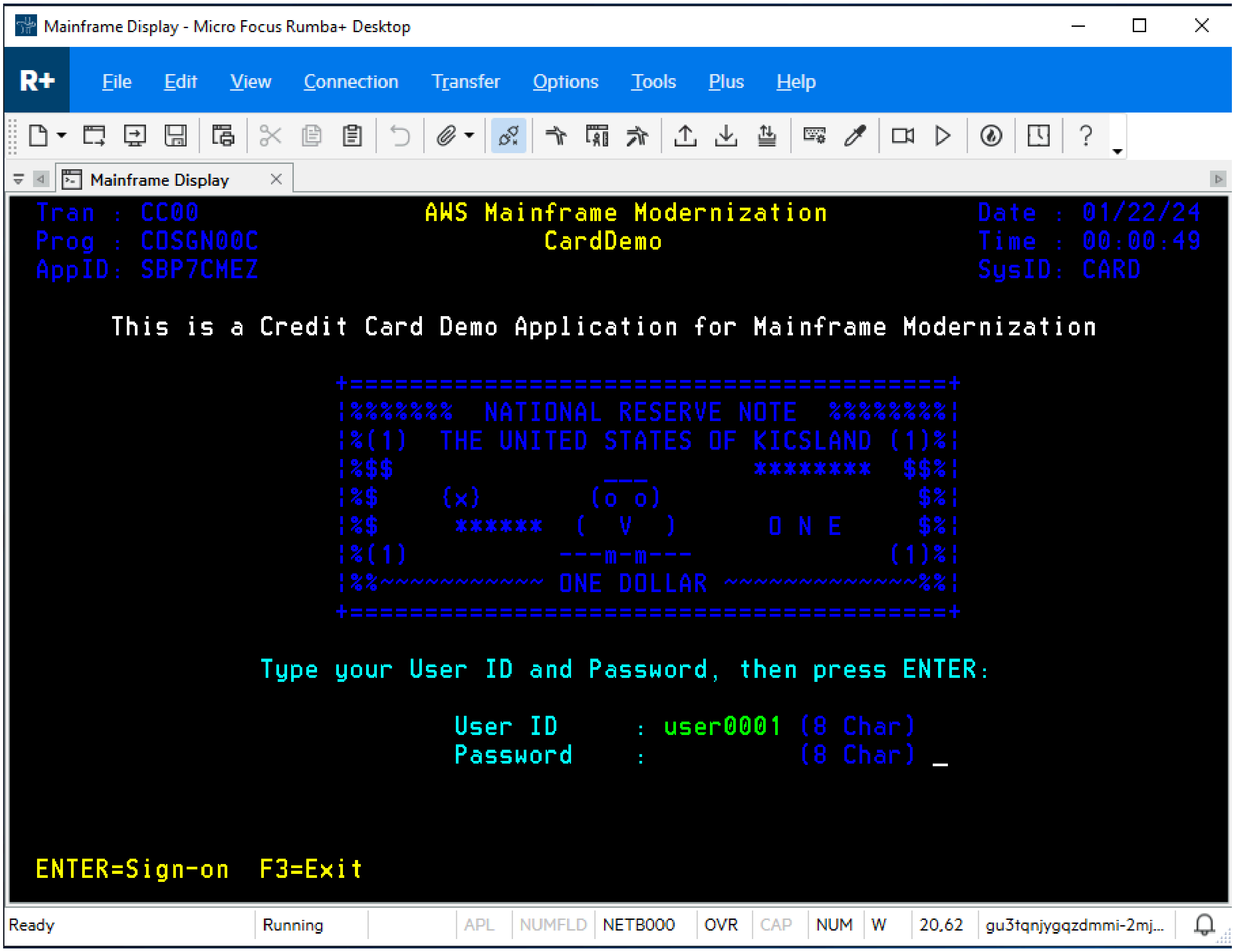


9. 选择连接。

> **Note**
>
> 如果连接失败：
>
> - 如果您使用的是 AppStream 2.0，请确认为应用程序环境指定的 VPC 和安全组与 AppStream 2.0 队列相同。
> - 使用 VPC Reachability Analyzer 分析连接。您可以通过控制台访问 Reachability Analyzer。
> - 作为诊断步骤，请尝试添加或更改应用程序的安全组入站规则，以支持来自任何来源（即 CIDR Block 0.0.0.0/0）的端口 6000 流量。如果连接成功，则表明是安全组阻止了您的流量。将安全组来源更改为更具体的来源。有关安全组的更多信息，请参阅安全组基础知识。

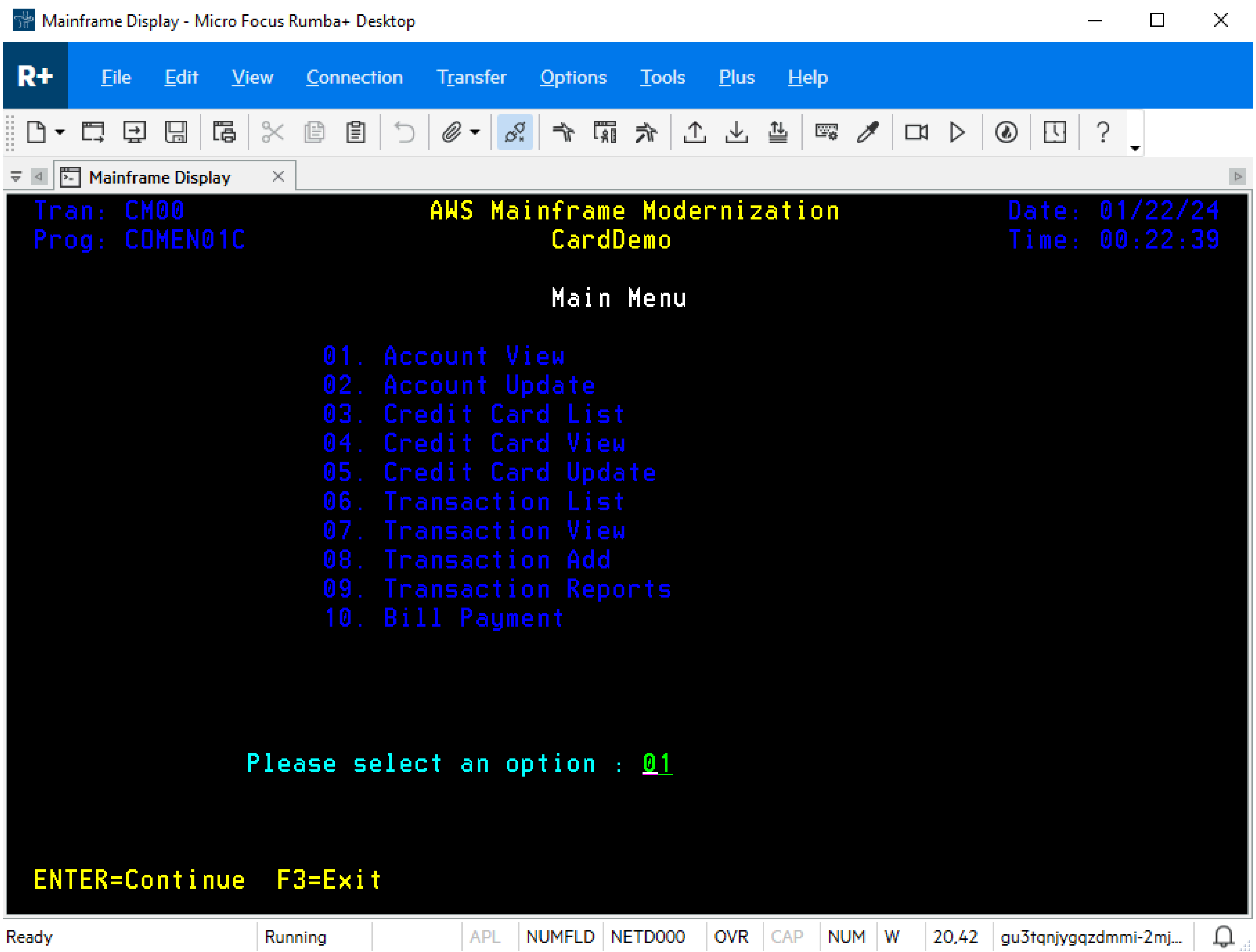
10. 对于用户名，请输入 `USER0001`，对于密码，请输入 `password`。

> **Note**
>
> 在 Rumba 中，“清除” 的默认值为 ctrl-shift-z，“重置” 的默认值为 ctrl-r。



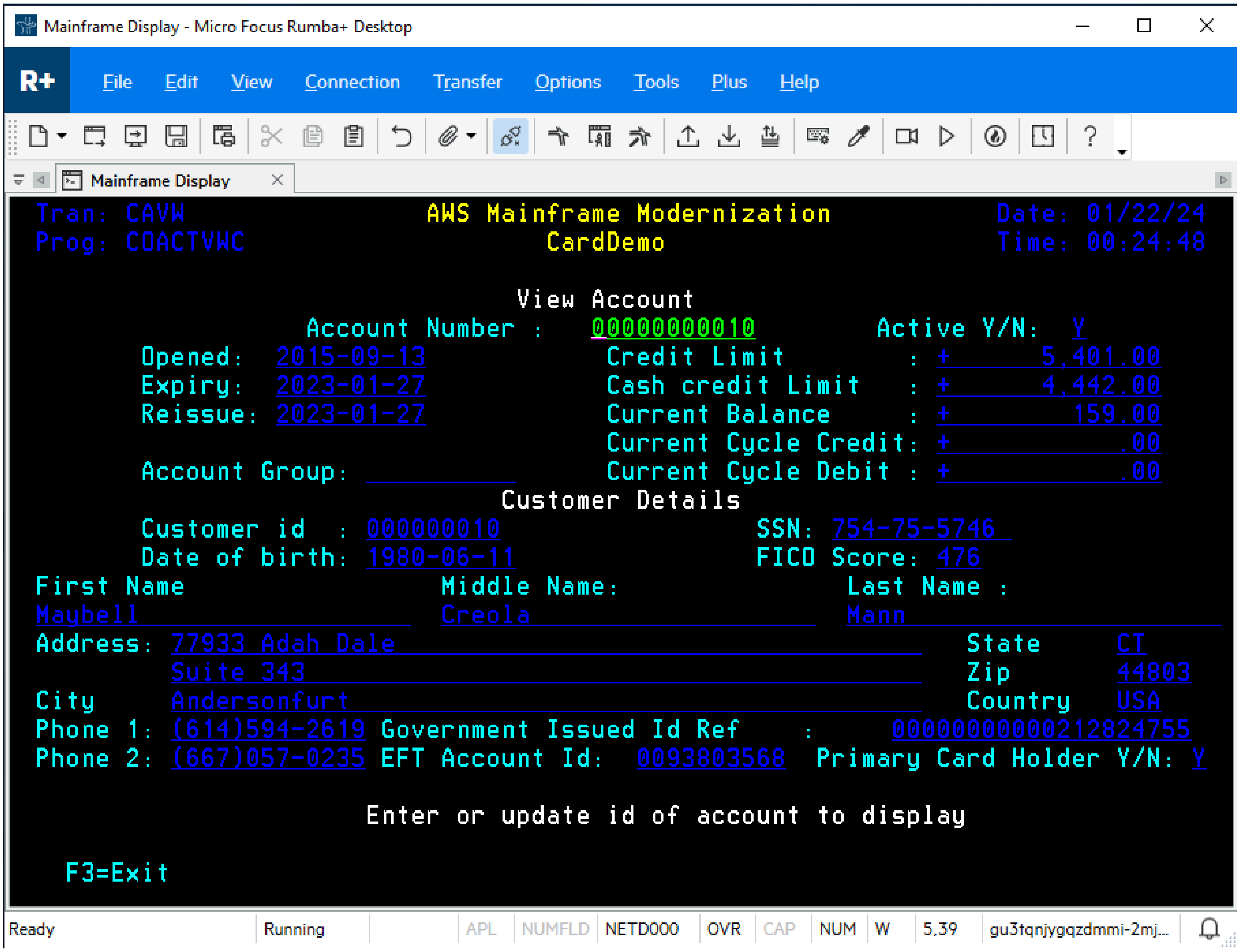
11. 成功登录后，您可以浏览 CardDemo应用程序。

12. 对于“账户视图”，请输入 01。



13. 对于“账户编号”，请输入 00000000010，并按键盘上的 Enter。

> **Note**
>
> 其他有效账户为 0000000011 和 00000000020。



14. 按 F3 可退出菜单，按 F3 可退出交易。

清理 资源

如果您不再需要为本教程创建的资源，请将其删除，以免产生额外费用。为此，请完成以下步骤：

- 如有必要，请停止该应用程序。
- 删除应用程序。有关更多信息，请参阅 删除 AWS Mainframe Modernization 应用程序。
- 删除运行时环境。有关更多信息，请参阅 删除 AWS 大型机现代化运行时环境。
- 删除您为本教程创建的 Amazon S3 存储桶。有关更多信息，请参阅《Amazon S3 用户指南》中的删除存储桶。
- 删除您为本教程创建的 AWS Secrets Manager 密钥。有关更多信息，请参阅删除密钥。

- 删除您为本教程创建的 KMS 密钥。有关更多信息，请参阅删除 AWS KMS 密钥。
- 删除您为本教程创建的 Amazon RDS 数据库。有关更多信息，请参阅 Amazon RDS 用户指南中的删除 EC2 实例和数据库实例。
- 如果您为端口 6000 添加了安全组规则，请删除该规则。

后续步骤

要了解如何为现代化应用程序设置开发环境，请参阅教程：设置 AppStream 2.0 以与 Rocket Enterprise Analyzer 和 Rocket Enterprise Developer 配合使用。

AWS 大型机现代化组件生命周期

AWS 大型机现代化的每个组件都要经历版本升级和开发生命周期。您可以将此页面用作概述，以了解这些组件、其版本升级计划以及 AWS 大型机现代化如何传达这些组件或其版本的发布或弃用。

组件生命周期概述

AWS 大型机现代化生命周期描述了在整个生命周期中发布和支持 AWS 大型机现代化服务组件的方法和时间表。提供可预测且一致的生命周期有助于您规划、测试和部署更新的版本。

根据每个组件的发布日历表，所有 AWS提供 AWS 的大型机现代化组件均受益于 支持 从发布到停用期间提供的产品支持。您可以在比较 支持 计划中了解有关 支持 范围和活动的更多信息。在活跃的现代化改造项目中，我们通常鼓励首先由专业的服务交付团队根据工作说明提供客户支持。

AWS 大型机现代化发布的某些组件的版本来自供应商，这些供应商可以是 AWS 自身、精选 AWS 合作伙伴或社区。对于每个 AWS 大型机现代化组件，一个版本都有一个主版本号和一个次要版本号。每个组件都有其自己的主要和次要版本编号。

对于采取版本控制的组件，我们有以下意图：

- 定期或根据客户需求发布新版本 AWS 的大型机现代化组件。如果需要某个组件的较新版本，但尚未在 AWS 大型机现代化服务中提供，则可以通过 支持 产品功能请求 (PFR) 提出明确请求。
- 使 AWS 大型机现代化组件特定版本的支持终止日期和停用日期与组件供应商的终止支持日期保持一致。
- 在组件的主要版本停用前约一年通知客户。

虽然我们努力遵守这些准则，但在某些情况下，我们可能会在更短的时间内停用特定版本，因而会缩短通知期限。例如，我们可能会迅速停用存在安全问题的版本，从而提供较短的通知期限。如果次要版本存在重大错误或安全问题，而这些问题已在更新的次要版本中得到解决，我们也可能会提前停用次要版本。万一发生此类情况，我们会通知客户并就停用计划和时间表进行沟通。时间表因情况而异。

> **Note**
>
> 组件的重要更新可能会随时发布。例如，出于安全原因或为生产环境提供修复的目的，可能会立即发布新版本。对于通过提出的请求 支持，支持计划规定了流程、严重程度和响应时间。

组件版本停用后， AWS 大型机现代化不会将这些版本分发给客户进行新部署。因此， 支持也不再支持这些版本。在版本停用日期之后运行现有组件部署的客户应意识到这样做的风险。 AWS 不负责为已停用的组件版本提供安全更新、技术支持或修补程序。此外，我们不会自动移除访问权限或删除您环境中的资源。我们强烈建议您每 3 个月检查一次新版本，并将所有 AWS 大型机现代化组件升级到最新支持的版本。

版本升级

适用于大型机重构的 AWS Transform 会定期发布其转换引擎和支持组件的新版本。这些更新提供了错误修复、安全性改进、性能增强、平台支持扩展和新的现代化功能。我们强烈建议您定期升级，以利用最新的修补程序、功能和安全更新。当新版本可用时，您可以控制是否以及何时将其应用于您的转换项目和生成的构件。版本版本遵循 Major.Minor.Patch 编号方案，分为三类：

- 主要版本引入了重要的新功能或结构变化。这些通常包括：
  - Support 支持其他传统平台（例如富士通 GS21）。
  - 新的传统语言支持（例如 Natural）。
  - 新的传统数据库支持（例如 Adabas、Fujitsu NDB）。
  - 目标现代化堆栈的主要版本更新（例如，较新的Spring、Java或PostgreSQL版本）。

  主要版本可能会引入重大更改，需要对现有的现代化应用程序进行调整，尤其是在重新生成代码或适应更新的目标架构定义时。
- 次要版本提供了进化改进、错误修复和维护更新。它们保持支持的旧platform/language/database定义和目标现代化堆栈不变。

  但是，次要版本仍可能包括生成的源代码或出于内部使用目的的运行时类中的重大更改。这可能会影响已经现代化的应用程序，这些应用程序在生成后经过手动维护，并且如果不进行额外的返工，就无法从原始遗留源安全地重新生成。
- 补丁版本可以作为紧急渠道，在保持兼容性的同时，快速解决最新版本中引入的关键回归问题或紧急修复。它们的范围很窄，旨在以最小的风险恢复稳定。

为了最大限度地降低任何升级期间的风险：

- 在将版本更改应用于实际项目或生产环境项目之前，请务必运行全面的非回归测试。
- 利用 DevOps 测试和部署管道（最好是在现代化计划期间构建），在升级后自动验证生成的代码和运行时行为。
- 升级组件 blue/green 时，请考虑运行时环境的灰色部署策略。

有关部署策略和变更管理的更多指导，请参阅 AWS Well-Architected 可靠性支柱。

AWS 使用 AWS Blu Age 进行大型机现代化重构发布概述

在 B AWS lu Age 运行时，该版本遵循一种`Major.Minor.Patch`模式。例如，对于 AWS Blu Age 运行时版本`4.1.0`，主版本为 4，次要版本为 1，补丁版本为 0。

当运行时或其依赖关系发生重大变化时，我们打算发布新的 AWS Blu Age 运行时主要版本。AWS Blu Age 运行时主要版本至少支持 12 个月，除非出现一些常见漏洞和漏洞 (CVEs)。该支持涵盖了我们的文档中提到的运行时特征中的错误。如果运行时依赖关系为 “临界” 和 “高 CVEs ”（Spring、Java、Tomcat 等），除非另有明确说明，否则从修复 CVE 的新运行时版本发布之日起 CVEs，主要版本的支持期限缩短至 6 个月， CVEs 从修复 CVE 的新运行时版本发布之日起缩短至 3 个月。

我们打算每月发布新的 AWS Blu Age 次要版本。客户需要定期升级版本以获得最新的安全修复、错误修复和特征增强。尚未投入生产的活跃项目必须尽快采用可用的最新运行时版本。

针对出现问题的特定主要版本，会在最新的次要版本中提供新的修复。如果您需要新修复，则需要升级到新的次要版本来应用这些修复。

提供受支持版本的补丁版本仅用于解决以前支持的次要版本中不存在的严重运行时缺陷。

Alpha 预发行版是在交付项目期间用于快速迭代的短期版本。由于未为 Alpha 预发行版本提供任何补丁，因此在较新的次要版本中提供了对 alpha 预发行版中发现的问题的修复程序。

您可以在 the section called “AWS Blu Age 发行说明”中找到有关每个运行时版本的发布日期和详细信息。

安全扫描由 Amazon Inspector 执行。

了解中的托管应用程序 AWS Mainframe Modernization

如果您不熟悉 AWS Mainframe Modernization 以下主题，可以开始使用：

- 什么是 AWS 大型机现代化？
- 为 AWS 大型机现代化做好准备
- 教程：为 AWS Blu Age 设置托管运行时
- 教程：为 Rocket Software（前身为 Micro Focus）

中的应用程序 AWS Mainframe Modernization 包含迁移的大型机工作负载。应用程序类似于大型机上的工作负载，并且与运行时环境相关联。您可以向应用程序添加批处理文件和数据集，并在应用程序运行时对其进行监控。您可以为迁移的每个工作负载创建 AWS Mainframe Modernization 应用程序。创建 AWS Mainframe Modernization 应用程序时，需要在创建应用程序时指定运行该应用程序的引擎。如果您使用的是自动重构模式，请选择 AWS Blu Age；如果您使用的是平台重构模式，请选择 Rocket Software（前身为 Micro Focus）。

主题

- 为迁移的应用程序创建 AWS 资源
- 创建 AWS Mainframe Modernization 应用程序
- 部署 AWS Mainframe Modernization 应用程序
- 更新 AWS Mainframe Modernization 应用程序
- 删除 AWS Mainframe Modernization 应用程序
- 提交批处理作业以供 AWS Mainframe Modernization 申请
- 取消 AWS Mainframe Modernization 应用程序的批处理作业
- 为 AWS Mainframe Modernization 应用程序导入数据集
- 导出 AWS Mainframe Modernization 应用程序的数据集
- 管理 AWS Mainframe Modernization 应用程序的事务
- 配置 Rocket 软件（前身为 Micro Focus）托管应用程序
- 配置 AWS Blu Age 托管应用程序
- AWS Mainframe Modernization 应用程序定义参考
- AWS 大型机现代化数据集定义参考

为迁移的应用程序创建 AWS 资源

为了在中运行已迁移的应用程序 AWS，必须与其他 AWS 资源一起创建一些资源 AWS 服务。必须创建的资源包括以下各项：

- 一个 S3 存储桶，用于存放应用程序代码、配置、数据文件和其他必需的构件。
- 一个 Amazon RDS 或 Amazon Aurora 数据库，用于保存应用程序所需的数据。
- AWS KMS key，这是创建和存储密钥 AWS Secrets Manager 所必需的。
- 一个 Secrets Manager 密钥，用于存放数据库凭证

> **Note**
> 每个迁移的应用程序均需要自己的一组上述资源。上述资源组是最低要求。您的应用程序可能还需要额外的资源，例如 Amazon Cognito 密钥或 MQ 队列。

所需的权限

请确保您具有以下权限：

- `s3:CreateBucket`, `s3:PutObject`
- `rds:CreateDBInstance`
- `kms:CreateKey`
- `secretsmanager:CreateSecret`

Amazon S3 存储桶

重构后的应用程序和更换平台后的应用程序都需要一个按如下方式配置的 Amazon S3 存储桶：

```
bucket-name/root-folder-name/application-name
```

bucket-name

符合 Amazon S3 命名约束条件的任何名称。我们建议您将 AWS 区域 名称作为存储桶名称的一部分。请确保在计划部署已迁移应用程序的同一区域中创建存储桶。

root-folder-name

满足应用程序定义中约束条件所需的名称，该定义是作为 AWS Mainframe Modernization 应用程序的一部分创建的。您可以使用 `root-folder-name` 来区分应用程序的不同版本，例如 V1 和 V2。

application-name

迁移后的应用程序的名称，例如， PlanetsDemo 或 BankDemo。

数据库

重构后的应用程序和更换平台后的应用程序都可能需要数据库。必须根据每个运行时引擎的特定要求来创建、配置和管理数据库。 AWS Mainframe Modernization 支持在此数据库上进行传输加密。如果在数据库上启用 SSL，请确保在数据库密钥中指定 `sslMode` 以及数据库的连接详细信息。有关更多信息，请参阅 AWS Secrets Manager 秘密。

如果您使用 AWS Blu Age 重构模式，并且需要Blusam数据库，则 AWS 蓝光时代运行时引擎需要一个 Amazon Aurora PostgreSQL 数据库，您必须创建、配置和管理该数据库。Blusam数据库是可选的。仅在您的应用程序需要时创建此数据库。要创建数据库，请按照《Amazon Aurora 用户指南》中创建 Amazon Aurora 数据库集群中的步骤进行操作。

如果你使用的是 Rocket Software 平台重组模式，你可以创建亚马逊 RDS 或亚马逊 Aurora PostgreSQL 数据库。要创建数据库，请按照《Amazon RDS 用户指南》中创建 Amazon RDS 数据库实例或《Amazon Aurora 用户指南》中创建 Amazon Aurora 数据库集群中的步骤进行操作。

对于这两个运行时引擎，您都必须 AWS Secrets Manager 使用存储数据库凭据 AWS KMS key 来对其进行加密。

AWS Key Management Service 钥匙

您必须将应用程序数据库的凭证安全地存储在 AWS Secrets Manager中。要在 Secrets Manager 中创建密钥，必须创建 AWS KMS key。要创建 KMS 密钥，请参阅《AWS Key Management Service 开发人员指南》中的创建密钥。

创建密钥后，必须更新密钥策略以授予 AWS Mainframe Modernization 解密权限。添加以下策略语句：

```
{
   "Effect" : "Allow",
   "Principal" : {
   "Service" : "m2.amazonaws.com"
```

```
    },
    "Action" : "kms:Decrypt",
    "Resource" : "*"
    }
```

AWS Secrets Manager 秘密

您必须将应用程序数据库的凭证安全地存储在 AWS Secrets Manager中。要创建密钥，请按《AWS Secrets Manager 用户指南》中创建数据库密钥的步骤操作。

AWS Mainframe Modernization 支持在此数据库上传输中的加密。如果在数据库上启用 SSL，请确保在数据库密钥中指定 `sslMode` 以及数据库的连接详细信息。可以为 `sslMode` 指定以下值之一：`verify-full`、`verify-ca` 或 `disable`。

在密钥创建过程中，选择资源权限 – 可选，然后选择编辑权限。在策略编辑器中，添加基于资源的策略（如下所示）以检索加密字段的内容。

```
{
    "Effect" : "Allow",
    "Principal" : {
    "Service" : "m2.amazonaws.com"
    },
    "Action" : "secretsmanager:GetSecretValue",
    "Resource" : "*"
    }
```

创建 AWS Mainframe Modernization 应用程序

使用 AWS Mainframe Modernization 控制台创建 AWS Mainframe Modernization 应用程序。创建应用程序支持您使用迁移后的大型机工作负载执行任务。

以下说明假定您已完成为 AWS 大型机现代化做好准备 中的步骤。

创建 应用程序

创建应用程序

1. 打开 AWS Mainframe Modernization 控制台，网址为https://console.aws.amazon.com/m2/。
2. 在 AWS 区域 选择器中，选择要在其中创建应用程序的区域。
3. 在应用程序页面上，选择创建应用程序。

4. 在指定基本信息页面的名称和描述部分，输入应用程序的名称。
5. （可选）在应用程序描述字段中，输入应用程序的描述。该描述有助于您及其他用户了解该应用程序的用途。
6. 在 “引擎类型” 部分，选择 Blu Age 进行自动重构，或选择 Micro Focus（Rocket）进行平台重构。
7. 如果要使用客户托管密钥，请在 KMS AWS KMS 密钥部分选择自定义加密设置。有关更多信息，请参阅 AWS 大型机现代化服务的静态数据加密。

> **Note**
>
> 默认情况下， AWS Mainframe Modernization 使用为您 AWS Mainframe Modernization 拥有和管理的 AWS KMS 密钥加密您的数据。但是，您可以选择使用客户管理的 AWS KMS 密钥。

8. （可选）按名称或 Amazon 资源名称 (ARN) 选择密 AWS KMS 钥，或者选择创建 AWS KMS 密钥进入 AWS KMS 控制台并创建新 AWS KMS 密钥。
9. （可选）在标签部分中，选择添加新标签以向应用程序添加一个或多个应用程序标签。应用程序标签是一种自定义属性标签，可帮助您组织和管理 AWS 资源）。
10. 选择下一步。
11. 在资源和配置部分，使用内联编辑器输入应用程序定义。或者，选择在 Amazon S3 存储桶中使用应用程序定义 JSON 文件并提供要使用的应用程序定义的位置。有关更多信息，请参阅 AWS Blu Age 应用程序定义示例或 火箭软件（前身为 Micro Focus）应用程序定义。
12. 选择下一步。
13. 在审核和创建页面中，检查您输入的信息，然后选择创建应用程序。

部署 AWS Mainframe Modernization 应用程序

使用 AWS Mainframe Modernization 控制台部署 AWS Mainframe Modernization 应用程序。在执行任务之前，您需要在运行时环境中部署应用程序。

以下说明假定您已完成为 AWS 大型机现代化做好准备 中的步骤。

部署应用程序

要运行 AWS Mainframe Modernization 应用程序，必须先将其部署到运行时环境。应用程序可以包含多个版本。应用程序的每个版本都有各自的应用程序定义。要部署应用程序，必须指定要部署的版本。

一次只能部署给定应用程序的一个版本。如果您部署应用程序的某个版本，然后决定改为部署其他版本，则必须先停止该应用程序（如果该应用程序正在运行）。

部署应用程序

1. 打开 AWS Mainframe Modernization 控制台，网址为https://console.aws.amazon.com/m2/。
2. 在 AWS 区域 选择器中，选择要在其中创建应用程序的区域。
3. 在应用程序页面上，选择要部署的应用程序。
4. 选择部署应用程序。
5. 在可用版本部分中，选择要部署的版本。
6. 在环境部分中，选择要在其中运行应用程序的运行时环境。
7. 选择部署。

部署已部署应用程序的不同版本

1. 打开 AWS Mainframe Modernization 控制台，网址为https://console.aws.amazon.com/m2/。
2. 在 AWS 区域 选择器中，选择要在其中创建应用程序的区域。
3. 在应用程序页面上，选择要部署的应用程序。
4. 从操作菜单中选择停止应用程序。
5. 应用程序停止后，选择部署应用程序。
6. 在可用版本部分中，选择要部署的版本。环境部分中已预先选择已部署该应用程序的环境。
7. 选择部署。

更新 AWS Mainframe Modernization 应用程序

使用 AWS Mainframe Modernization 控制台更新 AWS Mainframe Modernization 应用程序。更新应用程序会创建应用程序的新版本。

以下说明假定您已完成[为 AWS 大型机现代化做好准备](#) 中的步骤。

更新 应用程序

一个 AWS Mainframe Modernization 应用程序可以有多个版本，每个版本都有自己的应用程序定义。要更新应用程序，请提供新的应用程序定义。这样可以创建应用程序的新版本。

更新应用程序

1. 打开 AWS Mainframe Modernization 控制台，网址为https://console.aws.amazon.com/m2/。
2. 在 AWS 区域 选择器中，选择创建要更新的应用程序的区域。
3. 在应用程序页面上，选择要更新的应用程序。
4. 在应用程序详细信息页面的当前定义部分，选择编辑以更新当前的应用程序定义。
5. 在更新应用程序页面上，使用内联编辑器更新当前的应用程序定义。

   或者，选择在 Amazon S3 存储桶中使用应用程序定义 JSON 文件并提供要使用的应用程序定义的位置。有关更多信息，请参阅 AWS Blu Age 应用程序定义示例或 火箭软件（前身为 Micro Focus）应用程序定义。
6. 更新完应用程序定义后，选择更新。

> **Note**
>
> 更新应用程序后，必须将其重新部署。有关更多信息，请参阅 部署 AWS Mainframe Modernization 应用程序。

删除 AWS Mainframe Modernization 应用程序

您可以使用 AWS Mainframe Modernization 控制台从环境中删除 AWS Mainframe Modernization 应用程序。

以下说明假定您已完成为 AWS 大型机现代化做好准备 中的步骤。

删除 应用程序

如果您需要删除正在运行的 AWS Mainframe Modernization 应用程序，请务必先将其停止。您可以在应用程序页面上查看应用程序的状态。

删除 应用程序

1. 打开 AWS Mainframe Modernization 控制台，网址为https://console.aws.amazon.com/m2/。
2. 在 AWS 区域 选择器中，选择要从环境中删除的应用程序的创建区域。
3. 在应用程序页面上，选择要从环境中删除的应用程序，然后选择操作。

4. （可选）如果应用程序的状态为 `Running`，选择停止应用程序。
5. 选择从环境中删除。

删除过程会立即开始。

提交批处理作业以供 AWS Mainframe Modernization 申请

AWS Mainframe Modernization 您可以在其中为应用程序提交批处理作业。您可以提交或取消批处理作业，并查看有关批处理作业执行的详细信息。每次提交批处理作业时， AWS Mainframe Modernization 都会创建一个单独的批处理作业执行。您可以监控此作业的执行情况，也可以按名称搜索批处理作业，并向批处理作业提供 JCL 或脚本文件。

> **Important**
>
> 取消批处理作业不会删除该作业，而是会取消批处理作业的特定运行。您仍然可以使用批处理作业记录查看批处理作业运行的详细信息。

如果您的批处理作业需要访问一个或多个数据集，请使用 AWS Mainframe Modernization 控制台导入数据集。有关更多信息，请参阅 为 AWS Mainframe Modernization 应用程序导入数据集。

以下说明假定您已完成为 AWS 大型机现代化做好准备 和创建 AWS Mainframe Modernization 应用程序中的步骤。

主题

- 提交批处理作业
- 重启批处理作业

提交批处理作业

提交批处理作业

1. 打开 AWS Mainframe Modernization 控制台，网址为https://console.aws.amazon.com/m2/。
2. 在 AWS 区域 选择器中，选择要为其提交批处理作业的应用程序的创建区域。
3. 在应用程序页面上，选择要为其提交批处理作业的应用程序。

> **Note**
> 在向应用程序提交批处理作业之前，必须成功部署该应用程序。

4. 在应用程序详细信息页面上，选择批处理作业。
5. 选择提交作业。
6. 在选择脚本部分，选择一个脚本。您可以按名称搜索所需的脚本。
7. 选择提交作业。

重启批处理作业

重启批处理作业

> **Important**
> 以下引擎版本提供批处理作业重启功能：
>
> - Micro Focus（火箭）环境引擎版本 8.0.6 或更高版本。您还需要将 EFS 或 FSx 文件系统连接到您的环境。
> - AWS Blu Age 环境引擎版本 4.3.0 或更高版本。如果是 HA 环境，则还需要连接 EFS 或 FSx文件系统。

1. 打开 AWS Mainframe Modernization 控制台，网址为https://console.aws.amazon.com/m2/。
2. 在 AWS 区域 选择器中，选择创建应用程序和批处理作业的区域。
3. 在应用程序页面上，选择要重启批处理作业的应用程序。
4. 在应用程序详细信息页面上，选择批处理作业。
5. 从生成的列表中选择要重启的批处理作业。导航至操作菜单，然后选择重新启动作业。
6. 指定要如何重启批处理作业。你可以对 Micro Focus（Rocket）环境引擎和 AWS Blu Age 环境引擎执行以下操作：
   - 对于 Micro Focus（Rocket）环境引擎，你可以选择从头开始重启，也可以选择使用步骤或过程步骤重启。
     - “从头重新启动” 选项允许您从头开始重新启动批处理作业的所有步骤。

  - 使用 steps 或 procsteps 重启选项允许您选择要重新启动的特定步骤或过程步骤，也可以选择要结束的步骤或过程步骤。

> **Note**
>
> 结束步骤或过程步骤的编号必须大于或等于起始步骤或过程步骤的编号。

- 对于 AWS Blu Age 环境引擎，您可以从之前失败的 JCL/PROC 步骤重新启动批处理作业的最新执行，也可以通过绕过之前成功的步骤来执行延迟重启。
  - 您可以选择要重新启动的特定步骤名称。
  - 或者，您可以使用 “跳过” 步骤来绕过所选步骤，然后从工作流程的下一个步骤重新启动。

7. 选择提交作业。

取消 AWS Mainframe Modernization 应用程序的批处理作业

在中， AWS Mainframe Modernization 您可以取消应用程序的批处理作业。您可以查看有关批处理作业执行的详细信息。每次提交批处理作业时， AWS Mainframe Modernization 都会创建一个单独的批处理作业执行。您可以监控此作业的执行情况，也可以按名称搜索批处理作业，并向批处理作业提供 JCL 或脚本文件。

> **Important**
>
> 取消批处理作业不会删除该作业，而是会取消批处理作业的特定运行。您仍然可以使用批处理作业记录查看批处理作业运行的详细信息。

取消批处理作业

取消批处理作业时，不会删除批处理作业，而是会删除该批处理作业正在运行的任务。您仍然可以查看批处理作业的详细信息。

取消批处理作业

1. 打开 AWS Mainframe Modernization 控制台，网址为https://console.aws.amazon.com/m2/。
2. 在 AWS 区域 选择器中，选择包含批处理作业应用程序的区域。

3. 从批处理作业列表中查找并选择要取消的批处理作业。
4. 选择操作，然后选择取消作业。
5. 选择取消批处理作业。

这将取消已列入运行计划的所有批处理作业任务。

为 AWS Mainframe Modernization 应用程序导入数据集

AWS Mainframe Modernization 您可以通过导入数据集以用于您的应用程序。您可以在 Amazon S3 存储桶中存储的 JSON 文件中指定要导入的数据集，也可以单独指定数据集配置值。导入数据集后，可以查看导入任务的详细信息，以确认所需的数据集已导入。控制台中会一并列出应用程序的所有已编目数据集。

使用 AWS Mainframe Modernization 控制台导入 AWS Mainframe Modernization 应用程序的数据集。

以下说明假定您已完成为 AWS 大型机现代化做好准备 和创建 AWS Mainframe Modernization 应用程序中的步骤。

导入数据集

导入数据集

1. 打开 AWS Mainframe Modernization 控制台，网址为https://console.aws.amazon.com/m2/。
2. 在 AWS 区域 选择器中，选择要为其导入数据集的应用程序的创建区域。
3. 在应用程序页面上，选择要为其导入数据集的应用程序。
4. 在应用程序详细信息页面上，选择数据集。
5. 选择导入。
6. 请执行以下操作之一：
    - 选择使用 Amazon S3 存储桶中的数据集配置 JSON 文件，并提供数据集配置的位置。
    - 选择单独指定数据集配置值，并按引导配置操作。有关特定定义的详细信息，请参阅 the section called “数据集定义参考”。

      输入每个数据集配置值的名称、数据集组织（VSAM、GDG、PO、PS）、位置和外部 Amazon S3 位置以及参数设置。在引导配置中，您还可以选择生成 JSON 以查看您输入的 JSON 配置。

7. 选择提交。

导出 AWS Mainframe Modernization 应用程序的数据集

AWS Mainframe Modernization 您可以使用导出数据集以用于您的应用程序。您可以在 Amazon S3 存储桶中存储的 JSON 文件中指定要导出的数据集，也可以单独指定数据集配置值。导出数据集后，您可以查看导出任务的详细信息，以确认所需的数据集已导出。

使用 AWS Mainframe Modernization 控制台导出 AWS Mainframe Modernization 应用程序的数据集。

以下说明假定您已完成为 AWS 大型机现代化做好准备 和创建 AWS Mainframe Modernization 应用程序中的步骤。

导出数据集

导出数据集

1. 打开 AWS Mainframe Modernization 控制台，网址为https://console.aws.amazon.com/m2/。
2. 在 AWS 区域 选择器中，选择要为其导入数据集的应用程序的创建区域。
3. 在应用程序页面上，选择要为其导出数据集的应用程序。
4. 在应用程序详细信息页面上，选择数据集。
5. 选择导出。
6. 请执行以下操作之一：
   - 选择使用 Amazon S3 存储桶中的数据集配置 JSON 文件，并提供数据集配置的位置。
   - 选择单独指定数据集配置值，并按引导配置操作。有关更多信息，请参阅 the section called “数据集定义参考”。

     为每个数据集配置值输入数据集名称、外部 Amazon S3 位置和参数设置。在引导配置中，您还可以选择生成 JSON 以查看您输入的 JSON 配置。
7. 选择提交。

管理 AWS Mainframe Modernization 应用程序的事务

使用后， AWS Mainframe Modernization 您可以根据请求与许多其他用户同时运行应用程序，这些用户提交了使用相同文件和程序运行同一应用程序的请求。单个事务由一个或多个执行所需处理的应用程序组成。

以下说明假定您已完成为 AWS 大型机现代化做好准备 和创建 AWS Mainframe Modernization 应用程序中的步骤。

管理 应用程序的事务

管理应用程序事务

1. 打开 AWS Mainframe Modernization 控制台，网址为https://console.aws.amazon.com/m2/。
2. 在 AWS 区域 选择器中，选择要运行的应用程序的创建区域。
3. 在应用程序页面上，选择要为其管理事务的应用程序。
4. 在事务选项卡的事务资源下，从下拉列表中选择资源的显示方式。您可以根据事务资源、组、列表或来显示资源 SITs。
    - 事务资源允许您根据文件定义、事务定义、程序定义或临时数据队列定义来选择资源类型。

    > **Note**
    > 该 AWS Mainframe Modernization 服务支持其他资源类型来管理应用程序的事务，并且可以在控制台中进行访问。

    - 组是事务资源的集合。您可以选择要与事务资源关联的组。
    - 列表是按顺序排列的组集合。您可以在列表视图中查看所有事务资源和组。启动列表决定在服务器初始化时加载哪些资源。
        - 使用 AWS Blu Age 重构引擎，您可以指定要在启动时包含的列表。对列表的数量没有限制。
        - 使用 Rocket Software 平台重定向引擎，你可以在一个 SIT 中指定最多四个列表。
    - SIT（系统初始化表）显示所有可用的事务配置。您可以 SITs 根据属性（名称、描述和启动列表）进行查找。也可以选择列表来与您所选的 SIT 关联。

    > **Note**
    > SITs 仅适用于 Rocket 软件平台重定向引擎。

5. 选择事务资源以显示所有资源信息。您还可以查看与事务资源关联的所有属性。

配置 Rocket 软件（前身为 Micro Focus）托管应用程序

您可以使用 Rocket Software 运行时引擎配置应用程序，以自定义其他属性，包括集成。

主题

Rocket Software 支持的第三方集成

要使用第三方集成，您的 AWS 大型机现代化托管环境必须使用支持此类配置的 Rocket Software 引擎版本。支持带有后缀 R 的引擎版本（例如，版本 9.0.9.R）。这意味着，引擎版本 9.0.9.R 包含对第三方集成的客户端安装支持，但 9.0.9 不包含。

打印机

打印机资源是通过 Rocket Software 应用程序定义配置的，如the section called “打印机 - 可选”本节所述。

打印机定义可以为打印机定义自定义或服务提供的退出模块。一些可能的退出模块配置示例如下：

1. 加载服务提供的二进制文件示例。

```
...
{
    "name": "p1",
    "classes": [
        "AB"
    ],
    "description": "Using service managed LRS Queue exit module",
    "exit-module": {
        "name": "lrsprte6"
    }
},
...
```

2. 提供来自 S3 的二进制文件的示例。

```
"exit-module": {
        "name": "s3Exit",
        "module": "${s3-source}/3pa/s3Exit.so"
}
```

3. 提供来自 EFS 的二进制文件的示例。

> **Note**
>
> 要使用 EFS 挂载，必须在创建环境期间连接该挂载，同时还需要设置一些其他值，例如 `program-path`。

```
...
"batch-settings": {
    "jes-printers": [
        {
            "name": "p3",
            "classes": [
                "EF"
            ],
            "description": "Using binary from customer provided exit module on EFS
 Mount",
            "exit-module": {
                "name": "efsExit"
            }
        }
    ],
    "program-path": "$EFS_MOUNT/path/to/directory/containing/binaries/"
},
"runtime-settings": {
    "environment-variables": {
        "EFS_MOUNT": "/m2/mount/efs"
    }
}
...
```

LRS 队列 - 可选

要使用 LRS 队列，您必须使用支持第三方工件（即以结尾的引擎`.R`）的 Rocket Software 引擎。除了设置打印机，其退出模块指向`lrsprte6`退出模块的条目名称外，LRS 队列还需要一个额外的环境变量，该变量在 Rocket Software 应用程序定义中预先存在的 “运行时设置” 块中定义。

LRSQ_ADDRESS

（必需）指定 LRSQ 打印退出模块要发送到的 LRS 服务器地址。

LRS 打印机：配置 LRS 打印机需要 jes 打印机定义，如the section called “打印机 - 可选”部分中所指定。

此外，还必须在应用程序定义中的 `runtime-settings` 字段中指定 LRSQ_ADDRESS。

```
"runtime-settings": {
    "environment-variables": {
        "LRSQ_ADDRESS": "<lrsq-address>"
    }
}
```

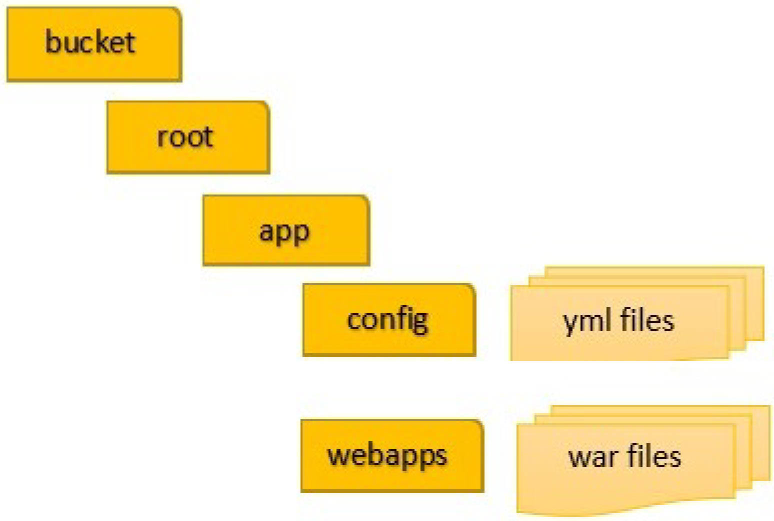
配置 AWS Blu Age 托管应用程序

您可以配置应用程序以包含对遗留实用程序的访问权限，也可以自定义其他属性。要了解您可以配置什么以及在何处配置，请参阅the section called “AWS Blu Age 托管应用程序的结构”本节以了解 AWS Blu Age 现代化应用程序的整体结构。

主题

AWS Blu Age 托管应用程序的结构

如果您使用 AWS Blu Age 重构模式，则 AWS Blu Age 运行时引擎希望应用程序定义中指定的文件夹`app-location`中包含以下结构：



config

包含您的项目的 YAML 文件。这些是特定于您的应用程序的 YAML 文件，通常命名类似`application-planetsdemo.yaml`于 AWS 大型机现代化为您自动提供和设置的`application-main.yaml`文件。

webapps

包含您的应用程序的 `war` 文件。这些文件是在现代化过程中产生的。

应用程序还可以具有以下可选文件夹：

jics/sql

包含用于为应用程序初始化 JICS 数据库的 `initJics.sql` 脚本。

scripts

包含应用程序脚本，您也可以直接在 `war` 文件中提供这些脚本。

sql

包含应用程序 SQL 文件，您也可以直接在 `war` 文件中提供这些脚本。

lnk

包含应用程序 LNK 文件，您也可以直接在 `war` 文件中提供这些脚本。

extra

包含可以为经过现代化改造的应用程序提供额外功能的 jar。

管理应用程序的 Java 选项

要管理应用程序的某些 Java 选项，请在指定的文件夹中添加名`tomcat.properties`为的属性文件`app-location`。此文件可包含三个属性：`xms`（指定最小 Java 内存消耗）、`xmx`（指定最大 Java 内存消耗）和 `dnscachettl`（管理 dns 解析的缓存持续时间）。以下是一个有效的 `tomcat.properties` 文件的内容示例。

```
xms=512M
xmx=1G
dnscachettl=5
```

您为前两个属性指定的值可以采用以下任何单位：

- 字节：不需指定单位。
- 千字节：在值后面加一个 K。
- 兆字节：在值后面加一个 M。
- 千兆字节：在值后面加一个 G。

第三个属性的值表示以秒为单位的缓存持续时间，其值可以为 -1（永久缓存），也可以为从 0（从不缓存）到 999。在托管应用程序部署情境中，默认值为 -1。

为托管应用程序配置对实用程序的访问权限

使用 AWS Blu Age 重构大型机应用程序时，如果您的应用程序依赖于各种传统平台实用程序，例如 IDCAMS、INFUTILB、SORT 等，则可能需要为它们提供支持。 AWS Blu Age 重构通过与现代化应用程序一起部署的专用 Web 应用程序为这种访问提供了这种访问权限。此 Web 应用程序需要您提供配置文件 `application-utility-pgm.yml`。如果您未提供此配置文件，则 Web 应用程序无法与您的应用程序一并部署，并且不可用。

主题

- [配置属性](#)

本主题介绍了您可以在 `application-utility-pgm.yml` 配置文件中指定的所有可能属性及其默认值，包括必需属性和可选属性。以下是一个完整的配置文件示例，其中的属性按我们推荐的顺序列出。您可以将此示例为起点，设置自己的配置文件。

```
# 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
```

配置属性

您可以在配置文件中指定以下属性。

spring.jta.enabled

（可选）控制是否启用 JTA 支持。对于实用程序，我们建议您将此值设置为 `false`。

```
spring.jta.enabled : false
```

logging.config

（必需）指定专用记录器配置文件的路径。建议使用名称 `logback-utility.xml` 并在经过现代化改造的应用程序中使用此文件。组织这些文件的常用方法是将所有记录器配置文件放在同一个位置，通常放在子文件夹 `/config/logback` 中，其中 `/config` 是包含 YAML 配置文件的文件夹。有关更多信息，请参阅 Logback 文档中的第三章：Logback 配置。

```
logging.config : classpath:logback-utility.xml
```

encoding

（必需）指定实用程序使用的字符集。在大多数情况下，从 z/OS 平台迁移时，此字符集是 EBCDIC 变体，应与为经过现代化改造的应用程序配置的字符集相匹配。如果未设置，默认值为 `ASCII`。

```
encoding : cp1047
```

sysPunchEncoding

（可选）指定 INFUTILB 和 DSNUTILB 用于生成和读取 SYSPUNCH 文件的字符集。如果您照原样使用来自遗留平台的 SYSPUNCH 文件，则此值应该是 EBCDIC 变体。如果未设置，默认值为 `ASCII`。

```
sysPunchEncoding : cp1047
```

数据来源配置

某些与数据库相关的实用程序（例如 LOAD 和 UNLOAD）需要通过数据来源访问目标数据库。与 AWS 大型机现代化中的其他数据源定义一样，此访问权限需要您使用 AWS Secrets Manager。指向 Secrets Manager 中正确密钥的属性如下所示：

主数据来源

这是主要的业务应用程序数据库。

spring.aws.client.datasources.primary.secret

（可选）指定 Secrets Manager 中包含数据来源属性的密钥。

```
spring.aws.client.datasources.primary.secret: datasource-secret-ARN
```

spring.aws.client.datasources.primary.dbname

（可选）如果数据库密钥中未直接提供数据库名称，则使用 dbname 属性指定目标数据库名称。

```
spring.aws.client.datasources.primary.dbname: target-database-name
```

spring.aws.client.datasources.primary.type

（可选）指定要使用的连接池实现的完全限定名称。默认值为 `com.zaxxer.hikari.HikariDataSource`。

```
spring.aws.client.datasources.primary.type: target-datasource-type
```

如果主数据来源的类型为 `com.zaxxer.hikari.HikariDataSource`，则可以按如下方式指定其他属性：

spring.datasource.primary.[property_name]

（可选）您可以使用此格式来指定用于配置主数据来源连接池实现的额外属性。

以下是 `com.zaxxer.hikari.HikariDataSource` 类型的主数据来源的示例。

```
spring:
   datasource:
       primary:
         autoCommit: XXXX
         maximumPoolSize: XXXX
         keepaliveTime: XXXX
         minimumIdle: XXXX
         idleTimeout: XXXX
         connectionTimeout: XXXX
         maxLifetime: XXXX
```

其他实用程序数据来源

除了主数据来源外，还可以提供其他实用程序数据来源。

spring.aws.client.utility.pgm.datasources.names

（可选）指定实用程序数据来源名称列表。

```
spring.aws.client.utility.pgm.datasources.names: dsname1, dsname2, dsname3
```

spring.aws.client.utility.pgm.datasources.[dsname].secret

（可选）指定托管数据来源属性的 SSM 中的密钥 ARN。在 `spring.aws.client.utility.pgm.datasources.names` 中指定的名称列表中提供 [dsname]。

```
spring.aws.client.utility.pgm.datasources.dsname1.secret: datasource-secret-ARN
```

spring.aws.client.utility.pgm.datasources.[dsname].dbname

（可选）如果数据库密钥中未直接提供数据库名称，则使用 `dbname` 属性指定目标数据库名称。在 `spring.aws.client.utility.pgm.datasources.names` 中指定的名称列表中提供 [dsname]。

```
spring.aws.client.utility.pgm.datasources.dsname1.dbname: target-database-name
```

spring.aws.client.utility.pgm.datasources.[dsname].type

（可选）指定要使用的连接池实现的完全限定名称。默认值为 `com.zaxxer.hikari.HikariDataSource`。在 `spring.aws.client.utility.pgm.datasources.names` 中指定的名称列表中提供 [dsname]。

```
spring.aws.client.utility.pgm.datasources.dsname1.type: target-datasource-type
```

如果实用程序数据来源类型为 `com.zaxxer.hikari.HikariDataSource`，则可以按如下方式提供其他属性：

spring.datasource.[dsname].[property_name]

（可选）指定一组用于配置实用程序数据来源连接池实现的额外属性。在 `spring.aws.client.utility.pgm.datasources.names` 中指定的名称列表中提供 [dsname]。采用以下格式指定这些属性：`property_name : value`

以下是 `com.zaxxer.hikari.HikariDataSource` 类型的额外实用程序数据来源的示例：

```
spring:
   datasource:
       dsname1:
         connectionTimeout: XXXX
         maxLifetime: XXXX
       dsname2:
         connectionTimeout: XXXX
         maxLifetime: XXXX
       dsname3:
         connectionTimeout: XXXX
         maxLifetime: XXXX
```

treatLargeNumberAsInteger

（可选）与 Oracle 数据库引擎细节和 DSNTEP2/DSNTEP4 实用程序用法有关。如果将此标志设置为 true，则来自 Oracle 数据库的大数字 (NUMBER (38,0)) 将被作为整数处理。默认值：`false`

```
treatLargeNumberAsInteger : false
```

zonedMode

（可选）设置分区模式，以对分区数据类型进行编码或解码。此设置会影响有符号数字的表示方式。有效值如下：

- EBCDIC_STRICT：默认值。对符号处理使用严格的定义。根据字符集是 EBCDIC 还是 ASCII，有符号数字表示使用以下字符：
  - 与字节 (Cn+Dn) 对应的 EBCDIC 字符表示正数和负数范围（`+0` 至 `+9`、`-0` 至 `-9`）。字符显示为 `{`,A 至 `I`, `}`、`J` 至 `R`
  - 与字节 (3n+7n) 对应的 ASCII 字符表示正数和负数范围（`+0` 至 `+9`、`-0` 至 `-9`）。字符显示为 `0` 至 9、 p 至 y。
- EBCDIC_MODIFIED：实用修改后的定义进行符号处理。对于 EBDIC 和 ASCII，相同的字符列表表示有符号数字，即 `+0` 至 `+9` 映射为 `{` + `A` 至 `I` 并且 `-0` 至 `-9` 映射为 `}` + `J` 至 R.\
- AS400：用于来自 iSeries (AS400) 平台的现代化传统资产。

```
zonedMode:EBCDIC_STRICT
```

jcl.type

（可选）表示经过现代化改造的 JCL 脚本的遗留类型。如果调用的 JCL 类型为 `vse`，IDCAMS 实用程序将使用此设置来定制返回码。有效值如下所示：

- `mvs`（默认值）
- `vse`

```
jcl.type : mvs
```

数据库卸载实用程序相关属性

使用这些属性来配置用于将数据库表卸载到数据集的实用程序。以下所有属性均为可选属性。

此示例显示了所有可能的卸载属性。

```
# 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
```

sqlCodePoint移动

（可选）指定一个整数值，该值表示数据上使用的 SQL 代码点转换。默认值是 0。这表示不进行代码点转换。将此设置与用于现代化应用程序的 SQL 代码点转换参数保持一致。当使用码点转换时，此参数的常用值为 384。

```
unload.sqlCodePointShift: 0
```

nbi

（可选）指定一个空的指示符字节，以十六进制值（作为字符串）形式添加到数据值右侧的。两个可能的值如下所示：

- whenNull：当数据值为空时，添加十六进制值。默认值为 6`。有时会改用高值 FF。

  ```
  unload.nbi.whenNull: "6F"
  ```

- whenNotNull：当数据值不为空但列可以为空时，添加十六进制值。默认值为 `00`（低值）。

  ```
  unload.nbi.whenNotNull: "00"
  ```

useDatabaseConfiguration

（可选）指定日期和时间格式化属性，用于处理 UNLOAD 查询中的日期/时间对象。默认值为 `false`。

- 如果设置为 `true`，则使用主配置文件 (`application-main.yml`) 中的 `pgmDateFormat`、`pgmTimeFormat` 和 `pgmTimestampFormat` 属性。
- 如果设置为 `false`，则使用以下日期和时间格式化属性：
  - `unload.format.date`：指定日期格式化模式。默认值为 `MM/dd/yyyy`。
  - `unload.format.time`：指定时间格式化模式。默认值为 `HH.mm.ss`。

- `unload.format.timestamp`：指定时间戳格式化模式。默认值为 `yyyy-MM-dd-HH.mm.ss.SSSSSS`。

chunkSize

（可选）指定用于创建 SYSREC 数据集的数据块的大小。这些数据集是数据集卸载操作的目标，可用于并行操作。默认值为 `0`（无数据块）。

```
unload.chunkSize:0
```

fetchSize

（可选）指定数据提取大小。该值是使用数据块策略时一次要提取的记录数。默认值：`0`。

```
unload.fetchSize:0
```

varCharIs空

（可选）指定如何处理不可为空的 varchar 列内容为空的情况。默认值为 `false`。

如果将此值设置为 `true`，则出于卸载目的，列内容将被作为空字符串（而不是单个空格字符串）进行处理。仅针对 Oracle 数据库引擎将此标志设置为 `true`。

```
unload.varCharIsNull: false
```

columnFiller

（可选）指定用于填充 varchar 列中已卸载列的值。可能的值为 space 或低值。默认值为 space。

```
unload.columnFiller: space
```

与数据库加载相关的属性

使用以下属性来配置将数据集记录加载到目标数据库（例如 DSNUTILB）的实用程序。以下所有属性均为可选属性。

此示例显示了所有可能的加载属性。

```
# 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
```

sqlCodePoint移动

（可选）指定一个整数值，该值表示数据上使用的 SQL 代码点转换。默认值为 0，这表示应用程序没有代码点转换。将此设置与用于经过现代化改造的应用程序的 SQL 代码点转换参数保持一致。当使用代码点转换时，此参数的常用值为 384。

```
load.sqlCodePointShift : 384
```

batchSize

（可选）指定一个整数值，该值表示在向数据库发送实际批处理语句之前要处理的记录数。默认值为 0。

```
load.batchSize: 500
```

format

（可选）指定在数据库加载操作期间用于日期/时间转换的日期和时间格式化模式。

- `load.format.localDate`：本地日期格式化模式。默认值为 `dd.MM.yyyy|dd/MM/yyyy|yyyy-MM-dd`。
- `load.format.dbDate`：数据库日期格式化模式。默认值为 `yyyy-MM-dd`。
- `load.format.localTime`：本地时间格式化模式。默认值为 `HH:mm:ss|HH.mm.ss`。
- `load.format.dbTime`：数据库时间格式化模式。默认值为 `HH:mm:ss`。

table-mappings

（可选）指定客户提供的遗留表名和现代表名之间的映射集合。DSNUTILB 实用程序会使用这些映射。

按以下格式指定值：MODERN_TABLE_NAME : LEGACY_TABLE_NAME

示例如下：

```
table-mappings:
 TABLE_1_NAME : LEGACY_TABLE_1_NAME
 TABLE_2_NAME : LEGACY_TABLE_2_NAME
 ...
 TABLE_*N*_NAME : LEGACY_TABLE_*N*_NAME
```

> **Note**
>
> 实用程序应用程序启动时，会显式记录所有提供的映射。

使用 AWS Blu Age 引擎为托管应用程序添加配置属性

你可以在重构后的应用程序的`config`文件夹中添加一个文件，这样你就可以访问 AWS Blu Age 运行时引擎中的新功能。您必须将此文件命名为 `user-properties.yml`。此文件不会替换应用程序定义，而是对其进行了扩展。本主题介绍可以在 `user-properties.yml` 文件中包含的属性。

> **Note**
>
> 您无法更改某些参数，因为它们要么由 AWS 大型机现代化控制，要么由应用程序定义控制。在应用程序定义中为应用程序定义的所有参数均比您在 `user-properties.yml` 中指定的参数优先级高。

有关重构应用程序结构的更多信息，请参阅[AWS Blu Age 托管应用程序的结构](#)。

下图显示了在 AWS Blu Age 示例应用程序结构中该`user-properties.yml`文件的位置 PlanetsDemo。

```
PlanetsDemo-v1/
   ## config/
   #  ## application-PlanetsDemo.yml
   #  ## user-properties.yml
   ## jics/
   ## webapps/
```

配置属性参考

以下是可用属性的列表。所有参数都是可选的。

主题

Gapwalk 应用程序属性

bluesam.fileLoading.commitInterval

可选。Blusam提交间隔。

类型：数字

默认值：100000

card.encoding

可选。卡片编码：与 `useControlMVariable` 一起使用。

类型：字符串

默认： CP1145

checkinputfilesize

可选。指定在文件大小为记录大小倍数的情况下是否要进行检查。

类型：布尔值

默认值：false

database.cursor.overflow.allowed

可选。指定是否允许游标溢出。设置为 `true` 时，无论游标位置如何，均可在游标处执行下一次调用。设置为 `false` 时，会在游标处执行下一次调用之前检查游标是否位于最后一个位置。仅在游标可滚动（敏感或不敏感）的情况下将其启用

类型：布尔值

默认：True

数据简化器。 onInvalidNumeric数据

可选。解码无效的数字数据时如何应对。允许的值包括：`reject`、`toleratespaces`、`toleratespaceslowvalues`、`toleratemost`。

类型：字符串

默认：reject

defaultKeepExisting档案

可选。指定是否设置数据集的默认先前值。

类型：布尔值

默认：false

disposition.checkexistence

可选。指定是否针对配置 DISP SHR 或 OLD 的数据集检查文件是否存在。

类型：布尔值

默认：false

externalSort.threshold

可选。排序阈值：确定系统何时应从内存中切换到外部合并排序（使用文件系统上的临时文件）。如果未指定 (default=`null`)，则阈值将设置为每个新排序任务中可用内存的一半。为了估计可用内存，需要调用 system.gc ()。

类型：数据大小

默认值：null

```
externalSort.threshold: 12MB
```

blockSizeDefault

可选。用于 BDW 字节的默认区块大小。

类型：数字

默认值：32760

```
blockSizeDefault: 32760
```

forceHR

可选。指定是否在控制台或文件输出上使用人类可读的 SYSPRINT。

类型：布尔值

默认：false

forcedDate

可选。在数据库中强制指定特定日期和时间。仅在开发和测试期间使用。

默认值：null

```
forcedDate: 2022-08-26T12:59:58.123456+01:57
```

frozenDate

可选。冻结数据库中的日期和时间。仅在开发和测试期间使用。

默认：false

```
frozenDate: false
```

ims.messages.extendedSize

可选。指定是否在 ims 消息上设置 extendedSize。

类型：布尔值

默认：false

lockTimeout

可选。无法在指定时间范围内获取锁时，事务的超时时间（单位为毫秒）。

类型：数字

默认值：500

mapTransfo.prefixes

可选。转换 controlM 变量时要使用的前缀列表。每一个前缀均用逗号隔开。

类型：字符串

默认值：&,@,%%

查询。 useConcatCondition

可选。指定键条件是否通过键连接构建。

类型：布尔值

默认：false

rollbackOnRTE

可选。指定是否在运行时异常时回滚隐式运行单元事务。

类型：布尔值

默认：false

sctThreadLimit

可选。用于触发脚本的线程限值。

类型：数字

默认：5

sqlCodePoint移动

可选。sql 代码点转换。对在将遗留 rdbms 数据迁移到现代 rdbms 时可能遇到的控制字符的代码点进行转换。例如，您可以指定 `384` 来匹配 unicode 字符 `\u0180`。

类型：数字

默认值：0

sqlIntegerOverflow已允许

可选。指定是否允许 SQL 整数溢出，即是否允许在主机变量中放置更大的值。

类型：布尔值

默认：false

stepFailWhenAbend

可选。指定在步骤失败或完成执行时是否引发异常中止。

类型：布尔值

默认：True

stopExecutionWhenProgNotFound

可选。指定在找不到程序时是否停止运行。如果设置为 `true`，则在找不到程序时中断运行。

类型：布尔值

默认：True

系统.date.format

可选。指定系统日期格式。

类型：字符串

默认：MDY

uppercaseUserInput

可选。指定用户输入是否必须采用大写形式。

类型：布尔值

默认：True

useControl MVariable

可选。指定是否使用 control-M 规范进行变量替换。

类型：布尔值

默认：false

jcl.checkpoint.expireTimeout

可选。指定在持久性提供程序或内存中的注册表中保留 JCL 检查点的持续时间。

类型：数字

默认值：-1

jcl.checkpoint expireTimeoutUnit

可选。指定 `jcl.checkpoint.expireTimeout` 属性的持续时间单位。支持的枚举常量值：`java.util.concurrent.TimeUnit`。

类型：字符串

默认值：SECONDS

Gapwalk 批处理脚本属性

encoding

可选。批处理脚本项目中使用的编码（不是 groovy）。预期的有效编码包含 `CP1047`、`IBM930`、`ASCII`、`UTF-8`...

类型：字符串

默认值：ASCII

Gapwalk Blugen 属性

managers.trancode

可选。对话管理器转码映射。允许您将 JICS 事务代码映射到对话管理器。预期格式为 `trancode1:dialogManager1;trancode2:dialogManager2;`。

类型：字符串

默认值：null

```
managers.trancode: OR12:MYDIALOG1
```

Gapwalk CL 命令属性

commands-off

可选。要关闭的命令列表，以逗号分隔。允许的值包括：PGM_BASIC、RCVMSG、SNDRCVF、CHGVAR、QCLRDTAQ、RTVJOBA、ADDLFM、ADDPFM、RCVF、C
当你想禁用或覆盖现有程序时很有用。 PGM_BASIC是专为调试目的而设计的特定的 AWS Blu Age 运行时程序。

类型：字符串

默认值：null

spring.datasource.primary.jndi-name

可选。主要 Java 命名和目录接口（JNDI）数据源。

类型：字符串

默认值：jdbc/primary

zonedMode

可选。对分区数据类型进行编码或解码的模式。允许的值包括：EBCDIC_STRICT、EBCDIC_MODIFIED、AS400。

类型：字符串

默认值：EBCDIC_STRICT

Gapwalk CL 运行程序属性

cl.configuration.context.encoding

可选。CL 文件的编码。预期的有效编码包含 CP1047、IBM930、ASCII、UTF-8...

类型：字符串

默认： CP297

cl.zonedMode

可选。对控制语言 (CL) 命令进行编码或解码的模式。允许的值包括：EBCDIC_STRICT、EBCDIC_MODIFIED、AS400。

类型：字符串

默认值：EBCDIC_STRICT

Gapwalk JHDB 属性

ims.programs

可选。要使用的 IMS 程序列表。用分号（;）分隔每个参数，用逗号（,）分隔每个事务。例如：`ims.programs: PCP008,PCT008;PCP054,PCT054;PCP066,PCT066;PCP068,PCT068;`

类型：字符串

默认值：null

jhdb.checkpointPath

可选。如果 `jhdb.checkpointPersistence` 不是 none，则此参数允许您设置检查点持久性路径（checkpoint.dat 文件存储位置），注册表中包含的所有检查点数据都将序列化并备份到所提供文件夹的文件 (checkpoint.dat) 中。请注意，此备份仅涉及检查点数据（scriptId、stepId、数据库位置和检查点区域）。

类型：字符串

默认值：file:./setup/

jhdb.checkpointPersistence

可选。检查点持久性模式。允许的值包括：`none`、`add`、`end`。使用 `add` 可在创建新检查点并将其添加到注册表后保留该检查点。使用 `end` 可在服务器关闭时保留检查点。任何其他值都会禁用持久性。请注意，每次向注册表中添加新的检查点时，所有现有的检查点都将被序列化并且文件会被擦除，而不是添加到文件中现有数据中。因此，根据检查点的数量，可能会对性能产生一些影响。

类型：字符串

默认值：none

jhdb.configuration.context.encoding

可选。Java 分层数据库 (JHDB) 编码。预期的有效编码字符串包含 `CP1047`、`IBM930`、`ASCII`、`UTF-8`...

类型：字符串

默认： CP297

jhdb。 identificationCardData

可选。用于将某些“操作员识别卡数据”硬编码到 CARD 参数指定的 MID 字段。

类型：字符串

默认值：""

jhdb.lterm

可选。允许您在进行 IMS 仿真的情况下强制使用通用的逻辑终端 ID。如果未设置，则使用会话 ID。

类型：字符串

默认值：null

jhdb.metadata.extrapath

用于为 psbs 和 dbds 文件夹指定一个特定于运行时的额外根文件夹的配置参数。

类型：字符串

默认值：file:./setup/

> **Note**
>
> 目前，对于部署限制，您必须将 dbds 和 psbs 目录复制到应用程序的配置目录或配置目录的子目录中：例如 config/setup
>
> ```
> config
> |- setup
>    |- dbds
>    |- psbs
> ```
>
> 并在 application-jhdb.yml 中设置 `jhdb.metadata.extrapath: file: ./config/setup/`

jhdb.navigation.cachenexts

可选。RDBMS 分层导航中使用的缓存持续时间（以毫秒为单位）。

类型：数字

默认值：5000

jhdb.query。 limitJoinUsage

可选。指定是否在 RDBMS 图形上使用限制联接使用参数。

类型：布尔值

默认：True

jhdb。 use-db-prefix

可选。指定是否在 RDBMS 的分层导航中启用数据库前缀。

类型：布尔值

默认：True

Gapwalk JICS 属性

jics.data。 dataJsonInit地点

可选。分析器在解析 CSD 时准备的 json 文件的位置，该文件用于初始化 jics 数据库。

类型：字符串

默认值：""

jics.db。 dataScriptLocation

可选。initJics.sql 脚本的位置，该脚本由 Analyzer 在解析大型机的 CSD 导出时准备。

类型：字符串

默认值：""

jics.db。 dataTestQuery地点

可选。包含单个 sql 查询的 sql 脚本的位置，该查询应返回对象计数（例如：计算 jics 程序表中的记录数）。如果计数等于 0，则将使用 `jics.db.dataScriptLocation` 脚本加载数据库，否则将跳过数据库加载。

类型：字符串

默认值：""

jics.db。 ddlScriptLocation

可选。Jics ddl 脚本位置。支持您使用 .sql 脚本启动 jics 数据库架构。

类型：字符串

默认值：""

```
jics.db.ddlScriptLocation: ./jics/sql/jics.sql
```

jics.db。 schemaTestQuery地点

可选。sql 文件的位置，该文件应包含唯一查询，且该查询返回 jics 架构中对象的数量（如果有）。

类型：字符串

默认值：""

jics。 runUnitLauncher池. 启用

可选。指定是否在 JICS 中激活运行单元启动器池。

类型：布尔值

默认：false

jics。 runUnitLauncher泳池。大小

可选。JICS 中的运行单元启动器池大小。

类型：数字

默认值：20

jics。 runUnitLauncherpool. 验证间隔

可选：JICS 中运行单元启动器池的验证间隔，以毫秒为单位。

类型：数字

默认值：1000

jics.queues.sqs.region

可选。 AWS 区域 适用于 Amazon SQS 的，在 JICS 中使用。为了提高性能，建议将该属性设置为与已部署应用程序相同的区域，但并不强制。

类型：字符串

默认值：eu-west-1

jics.xa.agent.timeout

可选。定义负责管理分布式事务的 xa 代理用于完成其操作的最长持续时间。

类型：数字

默认值：null

mq.queues.sqs.region

可选。 AWS 区域 适用于亚马逊 SQS MQ 服务。

类型：字符串

默认值：eu-west-3

任务执行器。 allowCoreThreadTimeOut

可选。指定是否允许核心线程在 JCIS 中超时。即使与非零队列结合使用，也可以实现动态增长和缩小（因为只有在队列已满后，池最大大小才会增长）。

类型：布尔值

默认：false

任务执行器。 corePoolSize

可选。当通过 groovy 脚本启动终端中的事务时，会创建一个新线程。可以使用此参数设置核心池大小。

类型：数字

默认：5

任务执行器。 maxPoolSize

可选。当通过 groovy 脚本启动终端中的事务时，会创建一个新线程。使用此参数设置池的最大大小（并行线程的最大数量）。

类型：数字

默认值：10

taskExecutor.queueCapacity

可选。当通过 groovy 脚本启动终端中的事务时，会创建一个新线程。使用此参数设置队列大小。（= 达到 `taskExecutor.maxPoolSize` 时待处理事务的最大数量）

类型：数字

默认值：50

Gapwalk 运行时属性

cacheMetadata

可选。指定是否缓存数据库元数据。

类型：布尔值

默认：True

check-groovy-file

可选。指定是否在注册前检查 groovy 文件内容。

类型：布尔值

默认：True

databaseStatistics

可选。指定是否允许 SQL 生成器收集和显示统计信息。

类型：布尔值

默认：false

dateTimeFormat

可选。 dateTimeFormat 描述了如何将数据库日期时间戳类型泄漏到数据简化器实体中。允许的值包括： `ISO`、`EUR`、`USA`、`LOCAL`

类型：字符串

默认值：ISO

dbDateFormat

可选。数据库目标日期格式。

类型：字符串

默认： yyyy-MM-dd

dbTimeFormat

可选。数据库目标时间格式。

类型：字符串

默认值：HH:mm:ss

dbTimestampFormat

可选。数据库目标时间戳格式。

类型：字符串

默认： yyyy-MM-ddhh: mm: ss.sssssss

fetchSize

可选。游标的 fetchSize 值。在 utils 使用区块获取数据时使用。 load/unload

类型：数字

默认值：10

强制禁用 SQLTrim StringType

可选。指定是否禁用所有 sql 字符串参数的 TRIM。

类型：布尔值

默认：false

localDateFormat

可选。本地日期格式列表。使用 | 分隔每种格式。

类型：字符串

localTimeFormat

可选。本地时间格式列表。使用 | 分隔每种格式。

类型：字符串

localTimestampFormat

可选。本地时间戳格式列表。使用 | 分隔每种格式。

类型：字符串

默认值：

pgmDateFormat

可选。程序中使用的日期时间格式。

类型：字符串

默认： yyyy-MM-dd

pgmTimeFormat

可选。程序 (pgm) 执行的时间格式。

类型：字符串

默认值：HH.mm.ss

pgmTimestampFormat

可选。时间戳格式

类型：字符串

默认： yyyy-MM-dd-HH.mm.ss.ssssss

Gapwalk 实用程序属性

jcl.type

可选。`.jcl` 文件类型。允许的值包括：`jcl`、`vse`。对于非 vse jcl，如果文件为空，IDCAMS 实用程序 PRINT/REPRO 命令将返回 4。

类型：字符串

默认值：mvs

listcat.variablelengthpreprocessor.enabled

可选。指定是否为 LISTCAT 命令启用可变长度预处理器。

类型：布尔值

默认：false

listcat.variablelengthpreprocessor.type

可选。listcat 文件中包含的对象的类型（如果启用 `listcat.variablelengthpreprocessor.enabled`）。允许的值包括：`rdw`、`bdw`。

类型：字符串

默认为：rdw

load.batchSize

可选。加载实用程序批次大小。

类型：数字

默认值：0

load.format.dbDate

可选。要使用的加载实用程序数据库格式。

类型：字符串

默认： yyyy-MM-dd

load.format.dbTime

可选。要使用的加载实用程序数据库时间。

类型：字符串

默认值：HH:mm:ss

load.format.localDate

可选。要使用的加载实用程序本地日期格式。

类型：字符串

默认：dd.mm.yyyy| |yyyy-mm-dd dd/MM/yyyy

load.format.localTime

可选。要使用的加载实用程序本地时间格式。

类型：字符串

默认值：HH:mm:ss|HH.mm.ss

加载。 sqlCodePoint移动

可选。加载实用程序的 SQL 代码点转换。运行字符转换进程。如果您的目标数据库来自 DB2 于 Postgresql，则为必填项。

类型：数字

默认值：0

sysPunchEncoding

可选。syspunch 编码字符集。支持的值为 `Cp1047`、`ASCII`。

类型：字符串

默认值：ASCII

treatLargeNumberAsInteger

可选。指定是否将大数字作为 `Integer` 处理。默认情况下，大数字作为 `BigDecimal` 处理。

类型：布尔值

默认：false

unload.chunkSize

可选。卸载实用程序使用的区块大小。

类型：数字

默认值：0

unload.columnFiller

可选。卸载实用程序列填充内容。

类型：字符串

默认值：space

unload.fetchSize

可选。允许您在卸载实用程序中处理游标时调整提取大小。

类型：数字

默认值：0

unload.format.date

可选。卸载实用程序中使用的日期格式（如果启用 `unload.useDatabaseConfiguration`）。

类型：字符串

默认：MM/dd/yyyy

unload.format.time

可选。卸载实用程序中使用的时间格式（如果启用 `unload.useDatabaseConfiguration`）。

类型：字符串

默认值：HH.mm.ss

unload.format.timestamp

可选。卸载实用程序中使用的时间戳格式（如果启用 `unload.useDatabaseConfiguration`）。

类型：字符串

默认： yyyy-MM-dd-HH.mm.ss.ssssss

unload.nbi。 whenNotNull

可选。数据库中的值不为空时，要添加的空字节指示符 (nbi) 值。

类型：十六进制

默认值：00

unload.nbi.whenNull

可选。数据库中的值为空时，要添加的空字节指示符 (nbi) 值。

类型：十六进制

默认值：6F

unload.nbi。 writeNullIndicator

可选。指定是否在卸载输出文件中写出空指示符。

类型：布尔值

默认：false

卸载。 sqlCodePoint移动

可选。卸载实用程序的 SQL 代码点转换。运行字符转换进程。如果您的目标数据库来自 DB2 于 Postgresql，则为必填项。

类型：数字

默认值：0

卸载。 useDatabaseConfiguration

可选。指定是否在卸载实用程序中使用 application-main.yml 中的日期或时间配置。

类型：布尔值

默认：false

卸载。 varCharIs空

可选。在 INFTILB 程序中使用此参数，如果设置为 `true`，则所有具有空（空格）值的不可为空的字段都将返回一个空字符串。

类型：布尔值

默认：false

其他属性

qtemp.cleanup.threshold.hours

可选。指定何时启用 `qtemp.dblog`。数据库分区的生命周期（以小时为单位）。

类型：数字

默认值：0

qtemp.dblog

可选。是否启用 QTEMP 数据库日志记录。

类型：布尔值

默认：false

qtemp.uuid.length

可选。QTEMP 的唯一 ID 长度。

类型：数字

默认值：9

quartz.scheduler。 stand-by-if-error

可选。指定当作业调度程序处于待机模式时是否触发作业执行。如果设置为 true，则启用时不会触发作业执行。

类型：布尔值

默认：false

warmUpCache

可选。指定是否在服务器启动时将所有数据通信表数据加载到预热缓存中。

类型：布尔值

默认：false

AWS Mainframe Modernization 应用程序定义参考

在中 AWS Mainframe Modernization，您可以在应用程序定义 JSON 文件中配置迁移的大型机应用程序，该文件特定于您选择的运行时引擎。应用程序定义既包含一般信息，也包含特定于引擎的信息。本主题介绍了 AWS Blu Age 和 Rocket Software（前身为 Micro Focus）应用程序的定义，并确定了所有必需和可选元素。

目录

一般头部区段

每个应用程序定义都首先指定有关模板版本和源位置的一般信息。应用程序定义的当前版本为 2.0。

使用以下结构来指定模板版本和源位置。

```
"template-version": "2.0",
    "source-locations": [
        {
            "source-id": "s3-source",
            "source-type": "s3",
            "properties": {
                "s3-bucket": "mainframe-deployment-bucket",
                "s3-key-prefix": "v1"
            }
        }
    ]
```

> Note
>
> 如果要将 S3 ARN 作为 s3 存储桶输入，则可以使用以下语法：

```
"template-version": "2.0",
  "source-locations": [
      {
          "source-id": "s3-source",
          "source-type": "s3",
          "properties": {
              "s3-bucket": "arn:aws:s3:::mainframe-deployment-bucket",
              "s3-key-prefix": "v1"
          }
      }
  ]
```

template-version

（必需）指定应用程序定义文件的版本。请勿更改此值。目前唯一支持的值为 2.0。使用字符串指定 `template-version`。

source-locations

指定应用程序在运行时所需的文件和其他资源的位置。

source-id

指定位置的名称。此名称用于根据需要在应用程序定义 JSON 中引用源位置。

source-type

指定源类型。目前唯一支持的值为 s3。

properties

提供源位置的详细信息。每个属性均使用字符串指定。

- `s3-bucket` – 必需。存储文件的 Amazon S3 存储桶名称。
- `s3-key-prefix` – 必需。指定存储文件的 Amazon S3 存储桶的文件夹的名称。

定义区段概述

指定应用程序运行所需的服务、设置、数据和其他典型资源的资源定义。更新应用程序定义时， AWS Mainframe Modernization 会通过比较应用程序定义 JSON 文件的先前版本和当前版本中的 `source-locations` 和 `definition` 列表来检测是否发生更改。

定义区段是特定于引擎的，可能会发生变化。以下各节针对两个引擎显示了特定于引擎的应用程序定义示例。

AWS Blu Age 应用程序定义示例

```
{
 "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 Blu Age 定义详情

侦听器 – 必需

指定用于通过 AWS Mainframe Modernization创建的 Elastic Load Balancing 访问应用程序的端口。请使用以下结构：

```
"listeners": [{
            "port": 8194,
            "type": "http"
}],
```

端口

（必需）除了公认端口 0 到 1023 之外，您可以使用任何可用端口。我们建议使用的端口范围为 8192 至 8199。请确保没有其他侦听器或应用程序使用此端口。

类型

（必需）目前仅支持 `http`。

AWS Blu Age 应用程序-必填

使用以下结构指定引擎获取应用程序映像文件的位置。

```
"ba-application": {
            "app-location": "${s3-source}/murachs-v6/",
            "files-directory": "/m2/mount/myfolder",
            "enable-jics": <true|false>,
            "enable-batch-restart": <true|false>,
            "shared-app-location": "${s3-source}/shared/"
},
```

app-location

Amazon S3 中存储应用程序映像文件的特定位置。

files-directory

（可选）批处理 input/output 文件的位置。必须是环境级别的 Amazon EFS 或 Amazon FSx 挂载点设置的子文件夹。该子文件夹必须归合适的用户所有，以供 AWS Mainframe Modernization内部运

行的 Blu Age 应用程序使用。为此，在将驱动器连接到 Linux Amazon EC2 实例时，`3001`必须创建一个带有 ID 的群组`101`和一个 ID 为 ID 的用户，并且所需的文件夹必须归该用户所有。例如，通过这种方式，Blu Age Manag ed 就可以使用该*`testclient`*文件 AWS Mainframe Modernization 夹。

```
groupadd -g 101 mygroup
useradd -M -g mygroup -p mypassword -u 3001 myuser
mkdir testclient
chown myuser:mygroup testclient
```

enable-jics

（可选）指定是否启用 JICS。默认值为 true。将其设置为 false 可防止生成 JICS 数据库。

enable-batch-restart

（可选）指定是否为批处理作业启用重启功能。默认值为 false。有关批量重启配置的更多信息，请参阅带有 AWS Blu Age 引擎的托管应用程序的配置属性`jcl.checkpoint`中前缀的 AWS Blu Age 引擎属性。

shared-app-location

（可选）Amazon S3 中存储共享应用程序元素的进一步位置，其中可以包含与 app-location 中相同的应用程序结构。

Blusam- 可选

使用以下结构指定Blusam数据库和 Redis 缓存。

```
"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

指定与应用程序搭配使用的数据库的属性。该数据库必须为 Aurora PostgreSQL 数据库。您可以指定以下属性：

- `nb-threads`-（可选）指定Blusam引擎所依赖的写入机制使用多少专用线程。默认值为 8。
- `batch-size`：（可选）指定回写机制用于启动批量存储操作的阈值。该阈值表示已修改记录的数量，这些记录将启动批量存储操作，以确保修改后的记录得以保存。触发器本身基于所耗时间（一秒钟）和批量大小（以先达到者为准）的组合。默认值为 10000。
- `name`：（可选）指定数据库的名称。
- `secret-manager-arn` – 指定包含数据库凭证的密钥的 Amazon 资源名称 (ARN)。有关更多信息，请参阅 步骤 4：创建和配置 AWS Secrets Manager 数据库密钥。

Redis

指定 Redis 缓存的属性，应用程序使用该缓存将临时数据存储在中央位置以提高性能。我们建议您同时对 Redis 缓存进行加密和实施密码保护。

- `hostname`：指定 Redis 缓存的位置。
- `port`：指定 Redis 缓存发送和接收数据的通信端口，通常为 6379。
- `useSsl`：指定 Redis 缓存是否已加密。如果缓存未加密，则将 `useSsl` 设置为 false。
- `secret-manager-arn`：指定包含 Redis 缓存密码的密钥的 Amazon 资源名称（ARN）。如果 Redis 缓存没有密码保护，请勿指定 `secret-manager-arn`。有关更多信息，请参阅 步骤 4：创建和配置 AWS Secrets Manager 数据库密钥。

AWS Blu Age 消息队列——可选

指定 Blu Age 应用程序的 JMS-MQ 连接详细信息。 AWS

```
"message-queues": [
  {
      "product-type": "JMS-MQ",
      "queue-manager": "QMgr1",
      "channel": "mqChannel1",
```

```
        "hostname": "mqserver-host1",
        "port": 1414,
        "user-id": "app-user1",
        "ssl-cipher": "*TLS12ORHIGHER",
        "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",
        "ssl-cipher": "*TLS12ORHIGHER",
        "secret-manager-arn": "arn:aws:secretsmanager:us-
west-2:123456789012:secret:sample/mq/test-279PTa"
    }
]
```

product-type

（必需）指定产品类型。目前，这只能是 Blu Age 应用程序的 “JMS-MQ”。 AWS

queue-manager

（必需）指定队列管理器的名称。

渠道

（必需）指定服务器连接通道的名称。

hostname

（必需）指定消息队列服务器的主机名。

端口

（必需）指定服务器正在监听的侦听器端口号。

user-id

（可选）指定支持在指定通道上执行消息队列操作的用户账户 ID。

ssl-cipher

（可选）指定连接的 SSL 密码规范。

secret-manager-arn

（可选）指定 Secrets Manager 的 Amazon 资源名称（ARN），Secrets Manager 提供指定用户的密码。

AWS Blu Age 应用程序存储 EFS 配置-可选

使用以下结构指定应用程序存储 EFS 接入点详细信息。

```
"ba-application": {
            "file-permission-mask": "UMASK002"
},
"efs-configs": [
           {
            "file-system-id": "fs-01376dfsvfvrsvsr",
            "mount-point": "/m2/mount/efs-ap2",
            "access-point-id": fsap-0eaesefvrefrewgv8"
           }
]
```

file-system-id

（必需）应用接入点的 EFS 文件系统的 ID。模式：“fs-([0-9a-f]{8,40}){1,128}$”

mount-point

（必需）应用程序级文件系统的挂载点。必须不同于环境级存储挂载点。

access-point-id

（必需）接入点 ID，由 Amazon EFS 分配。模式：“^fsap-([0-9a-f]{8,40}){1,128}$”

file-permission-mask

（可选）为应用程序进程创建的文件定义文件创建掩码。例如，当该值设置为 `UMASK006` 时，所有文件都将具有 660 权限。这意味着只有文件所有者和文件组才具有读写权限，而其他用户则没有任何权限。

> **Note**
> 只有在使用应用程序级 EFS 存储时，才会考虑为此字段设置的值。

> **Note**
>
> 如果提供了 efs 配置，则必须在应用程序定义部分中指定文件目录。该目录必须是在应用程序级设置的 Amazon EFS 挂载点的子目录。

火箭软件（前身为 Micro Focus）应用程序定义

以下示例定义部分适用于 Rocket Software 运行时引擎，包含必需元素和可选元素。

```
{
 "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": "${s3-source}/batch/jcl",
            "program-path": "/m2/mount/libs/loadlib:$EFS_MOUNT/emergency/loadlib",
            "system-procedure-libraries":"SYS1.PROCLIB;SYS2.PROCLIB",
            "aliases": [
               {"alias": "FDSSORT", "program": "SORT"},
               {"alias": "MFADRDSU", "program": "ADRDSSU"}
            ]
        },
        "cics-settings": {
            "binary-file-location": "${s3-source}/cics/binaries",
            "csd-file-location": "${s3-source}/cics/def",
            "system-initialization-table": "BNKCICV"
        },
        "jes-printers": [
           {
             "name": "printerName",
              "classes": [
                 "A",
                 "B"
             ],
             "description": "printer desc....",
             "exit-module": {
                 "name": "lrsprte6"
                }
             }
         ],
        "xa-resources" : [{
            "name": "XASQL",
            "secret-manager-arn": "arn:aws:secrets:1234:us-east-1:secret:123456",
            "xa-connection-type": "postgres",
            "module": "${s3-source}/xa/ESPGSQLXA64.so"
        }],
        "runtime-settings": {
          "base-configuration-location": "${s3-source}/exported.json",
          "environment-variables": {
            "ES_JES_RESTART": "N",
            "EFS_MOUNT": "/m2/mount/efs",
            "LRSQ_ADDRESS": "<lrsq-address>"
          }
        }
```

```
    }
}
```

火箭软件定义详情

Rocket Software 应用程序定义文件定义部分的内容会有所不同，具体取决于迁移的大型机应用程序在运行时所需的资源。

侦听器 – 必需

使用以下结构指定侦听器：

```
"listeners": [{
    "port": 5101,
    "type": "tn3270"
}],
```

端口

对于 tn3270，默认值为 5101。对于其他类型的服务侦听器，端口会有所不同。除了公认端口 0 到 1023 之外，您可以使用任何可用端口。每个侦听器都应使用一个独有的端口。侦听器不应共享端口。有关更多信息，请参阅 Micro Focus Enterprise Server 文档中的 Listener Control。

类型

指定服务侦听器的类型。有关更多信息，请参阅 Micro Focus Enterprise Server 文档中的侦听器。

数据集位置 – 必需

使用以下结构指定数据集的位置。

```
"dataset-location": {
            "db-locations": [{
                "name": "Database1",
                "secret-manager-arn": "arn:aws:secrets:1234:us-east-1:secret:123456"
            }],
 }
```

db-locations

指定迁移的应用程序创建的数据集的位置。当前，仅 AWS Mainframe Modernization 支持来自单个 VSAM 数据库的数据集。

- `name` – 指定包含已迁移应用程序创建的数据集的数据库实例的名称。
- `secret-manager-arn` – 指定包含数据库凭证的密钥的 Amazon 资源名称（ARN）。

Amazon Cognito 身份验证和授权处理程序 – 可选

AWS Mainframe Modernization 使用 Amazon Cognito 对迁移的应用程序进行身份验证和授权。使用以下结构指定 Amazon Cognito 身份验证处理程序。

```
"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

指定用于对迁移应用程序的用户进行身份验证的 AWS Mainframe Modernization Amazon Cognito 用户池。用户池 AWS 区域 的应与 AWS Mainframe Modernization 应用程序 AWS 区域 的相匹配。

client-id

指定经过身份验证的用户可访问的已迁移应用程序。

identity-pool-id

指定 Amazon Cognito 身份池，经过身份验证的用户可以在该池中交换用户池令牌，获取支持用户访问 AWS Mainframe Modernization的凭证。身份池 AWS 区域 的应与 AWS Mainframe Modernization 应用程序 AWS 区域 的相匹配。

LDAP 和 Active Directory 处理程序 - 可选

您可以将您的应用程序与 Active Directory（AD）或任何类型的 LDAP 服务器集证进行授权和身份验证。

将应用程序与 AD 集成

1. 按照 Micro Focus Enterprise Server 文档中[为 Enterprise Server 安全配置 Active Directory]中描述的步骤进行操作。
2. 为要用于 AWS Secrets Manager 应用程序的每台 AD/LDAP 服务器创建一个包含您的 AD/LDAP 详细信息的密钥。有关如何创建密钥的信息，请参阅 AWS Secrets Manager 用户指南中的[创建 AWS Secrets Manager 密钥]。对于密钥类型，请选择其他类型的密钥，并包括以下键值对。

```
{
    "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>"
}
```

> **⚠ 安全建议**
>
> - 对于`connectionPath`， AWS Mainframe Modernization 支持 LDAP 和基于 SSL 的 LDAP (LDAPS) 协议。建议您使用 LDAPS，因为它更安全，并且可以防止凭证出现在网络传输中。
> - 对于 `authorizedId` 和 `password`，建议您依据以下原则为用户指定凭证：权限不超过应用程序运行所需的最严格的只读和验证权限。
> - 我们建议定期轮换 AD/LDAP 证书。
> - 不要使用用户名 `awsuser` 或 `mfuser` 创建 AD 用户。这两个用户名保留供 AWS 使用。

示例如下：

```
{
    "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"
}
```

使用客户托管的 KMS 密钥创建密钥。您必须向 AWS Mainframe Modernization 授予密钥的 `GetSecretValue` 和 `DescribeSecret` 权限，以及 KMS 密钥的 `Decrypt` 和 `DescribeKey` 权限。有关更多信息，请参阅《 AWS Secrets Manager 用户指南》中的 KMS 密钥权限。

3. 将以下内容添加到您的应用程序定义中。

```
"ldap-ad-auth-handler": {
    "ldap-ad-connection-secrets": [LIST OF AD/LDAP SECRETS]
}
```

示例如下：

```
"ldap-ad-auth-handler": {
    "ldap-ad-connection-secrets": ["arn:aws:secrets:1234:us-east-1:secret:123456"]
}
```

如果应用程序已与 LDAP 集成并已启动，则必须提供凭证才能执行支持的授权列表中提到的至少一项与应用程序相关的操作。

LDAP/AD 身份验证处理程序适用于 Micro Focus (Rocket) 8.0.11 及更高版本。

> **Note**
>
> 目前，LDAP 管理员必须在“OPERCMDS”企业服务器资源的 LDAP 目录中提供对 `casstart` 实用程序的“alter”权限。必须为所有必需的默认用户 [例如，CICSUSER（如果应用程序与 CICS 相关）] 执行此操作才能成功启动应用程序。

为身份验证和授权提供 LDAP 用户凭证

1. AWS Secrets Manager 使用以下键和值创建：

```
{
  "username" : "<USERNAME>",
  "password" : "<PASSWORD>"
}
```

> **Important**
>
> 您应该有权对正在使用的 Secrets Manager 执行 `DescribeSecrets` 和 `GetSecretValue`。此外，还要关联 KMS 密钥和必要的权限 AWS Secrets Manager，如选择中所述 AWS KMS key。

2. 选择 Secrets Manager 参数。

   AWS console

   从 AWS 控制台执行操作时，可以选择需要传递的 Secrets Manager。

   AWS CLI (or SDK)

   从 AWS CLI（或 SDK）执行操作时，API 参数 `auth-secrets-manager-arn` 应与 Secrets Manager ARN 一起传递。

以下是当前支持授权的应用程序操作列表：

- `StartBatchJob`
- `CancelBatchJobExecution`
- `ListBatchJobRestartPoints`

批处理理设置 – 必需

使用以下结构指定作为应用程序一部分运行的批处理理作业所需的详细信息。

```
"batch-settings": {
            "initiators": [{
                "classes": ["A", "B"],
                "description": "initiator...."
            }],
            "jcl-file-location": "${s3-source}/batch/jcl",
            "program-path": "/m2/mount/libs/loadlib:$EFS_MOUNT/emergency/loadlib",
```

```
            "system-procedure-libraries":"SYS1.PROCLIB;SYS2.PROCLIB",
            "aliases": [
               {"alias": "FDSSORT", "program": "SORT"},
               {"alias": "MFADRDSU", "program": "ADRDSSU"}
    ]
}
```

initiators

指定一个批处理启动器，该启动器在迁移的应用程序成功启动时启动，并应用程序停止运行前持续运行。您可以为每个启动器定义一个或多个类，也可以定义多个启动器。例如：

```
"batch-settings": {
            "initiators": [
                {
                  "classes": ["A", "B"],
                  "description": "initiator...."
                },
                {
                  "classes": ["C", "D"],
                  "description": "initiator...."
                }

        ],
 }
```

有关更多信息，请参阅 Micro Focus Enterprise Server 文档中的定义批处理启动器或打印机 SEP。

- `classes` – 指定启动器可以运行的作业类。最多可以使用 36 个字符。您可以使用以下字符：A-Z 或 0-9。
- `description` – 描述启动器的用途。

jcl-file-location

指定迁移应用程序运行的批处理作业所需的 JCL（作业控制语言）文件的位置。

program-path

指定当 JCL 中的程序不在默认位置时运行批处理作业所需的路径。不同的路径名称用冒号（:）分隔。

> **Note**
>
> 程序路径只能是 EFS 路径。

system-procedure-libraries

指定将在其中搜索 JCL 过程的默认分区数据集。但是，在 JCL 中或通过 JCLLIB 语句找不到该过程。必须对这些数据集进行编目并使用目录名称。这些条目用分号（;）分隔。

aliases

定义 JCL 中使用的实用程序和程序名称与实用程序实现名称的映射。 AWS 第三方批处理实用程序（例如 M2SFTP、M2WAIT、Syncsort 等）可以选择使用别名，从而无需更改 JCL。例如：

- FDSSORT Alias FDSSORT for SORT and Alias FDSICET for ICETOOL
- ADRDSSU Alias MFADRDSU for ADRDSSU
- Syncsort Alias DMXMFSRT for SORT

CICS 设置 – 必需

使用以下结构指定作为应用程序一部分运行的 CICS 事务所需的详细信息。

```
"cics-settings": {
            "binary-file-location": "${s3-source}/cics/binaries",
            "csd-file-location": "${s3-source}/cics/def",
            "system-initialization-table": "BNKCICV"
}
```

binary-file-location

指定 CICS 事务程序文件的位置。

csd-file-location

指定此应用程序的 CICS 资源定义 (CSD) 文件的位置。有关更多信息，请参阅 Micro Focus Enterprise Server 文档中的 CICS 资源定义。

system-initialization-table

指定迁移的应用程序使用的系统初始化表 (SIT)。SIT 表的名称最多可包含 8 个字符。您可以使用 A-Z、0-9、$、@ 和 #。有关更多信息，请参阅 Micro Focus Enterprise Server 文档中的 CICS 资源定义。

打印机 - 可选

可使用以下结构指定 jes 打印机。

```
"jes-printers": [
    {
        "name": "printerName",
        "classes": [
            "A",
            "B"
        ],
        "description": "printer desc....",
        "exit-module": {
            "name": "lrsprte6",
            "module" : "program"
        }
    }
],
```

> **Note**
>
> 对于给定的应用程序，最多可以配置 25 台打印机。

name

（必需）指定要与此打印机资源关联的名称。每台打印机的名称必须是唯一的，并且最多可以使用 128 个字母数字字符。

classes

（必需）指定适用于此打印机资源的输出类。最多可以使用 36 个字母数字字符。

描述

（可选）打印机的其他描述性文本。

exit-module

（可选）指定用于打印退出的自定义模块。没有默认值，如果未指定，则不使用退出模块。您可以使用托管打印退出模块，也可以提供自己的模块。托管打印出口模块使用 LRS 队列的保留名称 `lrsprte6` 来定义，或者您也可以通过使用 module 参数指定位置和名称来提供自己的模块。

`exit-module` 结构有两个组成部分：

- `name`：（必需）如果使用了 `exit-module`。退出模块条目的名称。退出模块条目名称不能超过 8 个字符。
- `module`：（可选）打印退出模块二进制文件的 S3 位置。

the section called “打印机”一节中提供了更多定义退出模块的示例。

XA 资源 - 可选

使用以下结构指定应用程序所需的 XA 资源所需的详细信息。

```
"xa-resources" : [{
            "name": "XASQL",
            "secret-manager-arn": "arn:aws:secrets:1234:us-east-1:secret:123456",
            "xa-connection-type": "postgres",
            "module": "${s3-source}/xa/ESPGSQLXA64.so"
}]
```

> **Note**
>
> XA 资源定义已更新为包含可选的 `xa-connection-type` 字段。如果未提供该字段，则假定连接类型为“postgres”。

name

（必需）指定 XA 资源的名称。

secret-manager-arn

（必需）指定包含用于连接数据库的凭证的密钥的 Amazon 资源名称（ARN）。

xa-connection-type

（可选）指定 XA 资源连接类型。

module

（必需）指定 RM 交换机模块可执行文件的位置。有关更多信息，请参阅 Micro Focus 企业服务器文档 XARs中的规划和设计。

运行时设置 - 可选

使用以下结构为运行时设置指定所需的详细信息，以管理支持的环境变量。

```
"runtime-settings": {
          "base-configuration-location": "${s3-source}/exported.json",
          "environment-variables": {
            "ES_JES_RESTART": "N",
            "EFS_MOUNT": "/m2/mount/efs"
    }
}
```

base-configuration-location

（可选）指定 Micro Focus 服务器配置的批量导入位置。此文件应为有效的 JSON，并且与上面定义的应用程序工件位置位于相同的 S3 位置。要从现有应用程序导出配置，请参阅 Rocket 软件文档中的从西亚经社会导出区域部分。

environment-variables

指定应用于此应用程序的运行时的 Micro Focus 支持的环境变量。

- ES_JES_RESTART是一个 Rocket Software 环境变量，它支持 JCL 重启处理。或者，您也可以用ES_ALLOC_OVERRIDE作 Rocket Software 环境变量。
- EFS_MOUNT 是一个自定义环境变量，可供应用程序用于识别环境的 EFS 挂载的位置。

你可以在《适用于 UNIX 的 Rocket E nterprise Server》指南中访问所有 Rock et S

AWS 大型机现代化数据集定义参考

如果您的应用程序需要多个数据集进行处理，则在 AWS 大型机现代化控制台中逐个输入这些数据集效率低下。我们建议您创建一个 JSON 文件来指定每个数据集。尽管许多参数都很常见，但在 JSON 中，要以不同的方式指定不同的数据集类型。本文档描述了导入不同类型数据集所需的 JSON 的详细信息。

> Note
>
> 在导入任何数据集之前，必须将数据集从大型机传输到 AWS。数据集必须采用可以加载到所选运行时引擎的格式。在许多情况下，这可以是一个顺序文件，但对于 Rocket Software（前

身为 Micro Focus）VSAM，它必须采用其专有格式。DFCONV 实用程序是转换文件的建议方法。在数据集定义 JSON 文件中指定存储桶和文件夹的名称。
有关 Rocket Software 运行时引擎的更多信息，请参阅 R ocket Softwar e 文档中的 DFCONV 批量文件转换。
有关 AWS Blu Age 的更多信息，请参阅the section called “AWS Blu Age 运行时配置”。

主题

常用属性

部分参数是所有数据集通用的。这些参数涵盖以下区域：

- 有关数据集的信息（datasetName、datasetOrg、recordLength、encoding）。
- 有关从何处导入的信息，即数据集的源位置。该位置不是大型机上的位置，而是您上传数据集的 Amazon S3 位置的路径 (externalLocation)。
- 有关导入到何处的信息，即数据集的目标位置。该位置可以是数据库或文件系统，具体取决于您的运行时引擎。（storageType 和 relativePath）。
- 有关数据集类型的信息（特定数据集类型、格式、编码等）。

每个数据集定义都具有相同的 JSON 结构。以下示例 JSON 显示了所有这些通用参数。

```
{
    "dataSet": {
        "storageType": "Database",
        "datasetName": "MFI01V.MFIDEMO.BNKACC",
        "relativePath": "DATA",
        "datasetOrg": {
            "type": {
                type-specific properties
                    ...
```

```
            },
        },
    },
}
```

以下属性是所有数据集的通用属性。

storageType

必需。应用于目标位置。指定数据集是存储在数据库中还是文件系统中。可能的值为 `Database` 或 `FileSystem`。

- AWS Blu Age 运行时引擎：不支持文件系统。您必须使用数据库。
- Rocket Software 运行时引擎：都支持数据库和文件系统。你可以使用 Amazon Relational Database Service 或 Amazon Aurora 来管理数据库，使用亚马逊 Elastic File System 或 Amazon FSx for Lustre 来管理文件系统。

datasetName

（必需）指定在大型机上显示的数据集的完全限定名称。

relativePath

（必需）应用于目标位置。指定数据集在数据库或文件系统中的相对位置。

datasetOrg

（必需）指定数据集的类型。可能的值为 `vsam`、`gdg`、`ps`、`po` 或 `unknown`。

- AWS Blu Age 运行时引擎：仅支持 VSAM 类型的数据集。
- Rocket Software 运行时引擎：支持 VSAM、GDG、PS、PO 或未知类型数据集。

> **Note**
>
> 如果您的应用程序需要的文件不是 COBOL 数据文件而是 PDF 或其他二进制文件的文件，则可以按如下方式指定它们：
>
> ```
> "datasetOrg": {
>             "type": PS {
>                 "format": U
>             },
> ```

VSAM 数据集请求格式示例

- AWS Blu Age 运行时引擎：支持。
- 火箭软件运行时引擎：支持。

如果要导入 VSAM 数据集，请将 `datasetOrg` 指定为 `vsam`。您的 JSON 应与以下示例类似：

```
{
    "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"
}
```

VSAM 数据集支持以下属性。

编码

（必需）指定数据集的字符集编码。可能的值为 ASCII（A）、EBCDIC（E）和未知（?）。

format

（必需）指定 VSAM 数据集类型和记录格式。

- AWS Blu Age 运行时引擎：可能的值为 ESDS (ES) 和 KSDS (KS)。记录格式可以是固定的，也可以是可变的。

- Rocket Software 运行时引擎：可能的值是 ESDS (`ES`)、KSDS (`KS`) 和 RRDS ()。`RR`VSAM 定义包括记录格式，因此您无需单独指定它。

primaryKey

（必需）仅适用于 VSAM KSDS 数据集。指定主键。包括主键名称、键偏移和键长度。`name` 为可选项，`offset` 和 `length` 为必需项。

recordLength

（必需）指定记录的长度。对于固定长度的记录格式，这些值必须匹配。

- AWS Blu Age 运行时引擎：适用于 VSAM ESDS 和 KSDS，`min`是可选的，是必需的。`max`
- 火箭软件运行时引擎：`min`和`max`是必需的。

externalLocation

（必需）指定源位置：即您上传数据集的 Amazon S3 存储桶。

Blu Age 引擎特有的属性

AWS Blu Age 运行时引擎支持 VSAM 数据集的压缩。以下示例说明如何在 JSON 中指定此属性。

```
{
    common properties
        ...
        "datasetOrg": {
            "vsam": {
                common properties
                ...
                "compressed": boolean,
                common properties
                ...
            }
        }
}
```

按如下方式指定压缩属性：

compression

（可选）指定此数据集的索引是否存储为压缩值。如果您的数据集很大（通常大于 100 Mb），请考虑将此标志设置为 `true`。

GDG base 数据集请求格式示例

- AWS Blu Age 运行时引擎：不支持。
- 火箭软件运行时引擎：支持。

如果要导入 GDG base 数据集，请将 `datasetOrg` 指定为 gdg。您的 JSON 应与以下示例类似：

```
{
    "storageType": "Database",
    "datasetName": "AWS.M2.GDG",
    "relativePath": "DATA",
    "datasetOrg": {
        "gdg": {
            "limit": "3",
            "rollDisposition": "Scratch and No Empty"
        }
    }
}
```

GDG base 数据集支持以下属性。

limit

（必需）指定活跃生成数或偏差。对于 GDG base 集群，最大值为 255。

rollDisposition

（可选）指定在达到或超过最大值时如何处理生成数据集。可能的值为 `No Scratch and No Empty`、`Scratch and No Empty`、`Scratch and Empty` 或 `No Scratch and Empty`。默认值为 `Scratch and No Empty`。

PS 或 GDG 生成的数据集请求格式示例

- AWS Blu Age 运行时引擎：不支持。
- 火箭软件运行时引擎：支持。

如果要导入 PS 或 GDG 生成数据集，请将 `datasetOrg` 指定为 ps。您的 JSON 应与以下示例类似：

```
{
```

```
    "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"
}
}
```

PS 或 GDG 生成数据集支持以下属性。

format

（必需）指定数据集记录的格式。可能的值为 F、FA、FB、FBA、FBM、FBS、FM、FS、LSEQ、U、V、VA、VB、VBA、VBM、VBS、VM 和 VS。

编码

（必需）指定数据集的字符集编码。可能的值为 ASCII (A)、EBCDIC (E) 和未知 (?)。

recordLength

（必需）指定记录的长度。必须指定记录的最小 (`min`) 和最大 (`max`) 长度。对于固定长度的记录格式，这些值必须匹配。

externalLocation

（必需）指定源位置：即您上传数据集的 Amazon S3 存储桶。

PO 数据集请求格式示例

如果要导入 PO 数据集，请将 `datasetOrg` 指定为 `po`。您的 JSON 应与以下示例类似：

```
{
```

```
    "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/"
}
}
```

PO 数据集支持以下属性。

format

（必需）指定数据集记录的格式。可能的值为 F、FA、FB、FBA、FBM、FBS、FM、FS、LSEQ、U、V、VA、VB、VBA、VBM、VBS、VM 和 VS。

编码

（必需）指定数据集的字符集编码。可能的值为 ASCII（A）、EBCDIC（E）和未知（?）。

memberFileExtensions

（必需）指定包含一个或多个文件扩展名的数组，支持您指定要包含哪些文件作为 PDS 成员。

recordLength

（可选）指定记录的长度。记录的最小 (min) 和最大 (max) 长度都是可选的。对于固定长度的记录格式，这些值必须匹配。

externalLocation

（必需）指定源位置：即您上传数据集的 Amazon S3 存储桶。

> **Note**
>
> Rocket Software 运行时引擎的当前实现将 PDS 条目添加为动态数据集。

AWS 大型机现代化中的托管运行时环境

如果您不熟悉 AWS 大型机现代化改造，请参阅以下主题以开始使用：

- 什么是 AWS 大型机现代化？
- 为 AWS 大型机现代化做好准备
- 开始进行 AWS 大型机现代化改造
- 教程：为 AWS Blu Age 设置托管运行时
- 教程：为 Rocket Software（前身为 Micro Focus）

AWS 大型机现代化中的运行时环境是 AWS 计算资源、运行时引擎和您指定的配置详细信息的命名组合。运行时环境托管一个或多个应用程序。 AWS 大型机现代化中的应用程序包含迁移的大型机工作负载。您可以为所创建的环境选择运行时引擎。如果您使用的是自动重构模式，请选择 AWS Blu Age；如果您使用的是平台重构模式，请选择 Rocket Software（前身为 Micro Focus）。您还可以选择适合您的应用程序的计算资源量，也可以选择将存储附加到运行时环境。 AWS 大型机现代化为您启用 Amazon CloudWatch 指标和日志记录，以便您可以监控运行时环境。

主题

- 创建 AWS 大型机现代化运行时环境
- 更新 AWS 大型机现代化运行时环境
- 停止 AWS 大型机现代化运行时环境
- 重启 AWS 大型机现代化运行时环境
- 删除 AWS 大型机现代化运行时环境

创建 AWS 大型机现代化运行时环境

使用 AWS 大型机现代化控制台创建 AWS 大型机现代化环境。

以下说明假定您已完成为 AWS 大型机现代化做好准备 中的步骤。

创建运行时环境

创建运行时环境

1. 打开 AWS 大型机现代化控制台，网址为https://console.aws.amazon.com/m2/。

2. 在 AWS 区域 选择器中，选择要在其中创建环境的区域。
3. 在环境页面上，选择创建环境。
4. 在指定基本信息页面上，提供以下信息：

    a. 在名称和描述部分，输入环境的名称。
    b. （可选）。在环境描述字段中，输入环境的描述。该描述有助于您及其他用户了解该运行时环境的用途。
    c. 在引擎选项部分，选择 Blu Age 进行自动重构，或选择 Micro Focus（Rocket）进行平台重构。
    d. 为所选引擎选择一个版本。
    e. （可选）。在标签部分中，选择添加新标签以向环境添加一个或多个环境标签。环境标签是一种自定义属性标签，可帮助您组织和管理 AWS 资源。
    f. 选择下一步。

5. 在指定配置页面上，提供以下信息：

    a. 在可用性部分中，选择独立运行时环境或高可用性集群。

        可用性模式决定了您的应用程序在运行时的可用性。独立模式适用于开发用途。高可用性模式适用于必须始终可用的应用程序。

    b. 在资源中，选择实例类型和所需容量。

        这些资源是 AWS 大型机现代化托管的 Amazon EC2 实例，用于托管您的运行时环境。独立运行时环境提供两种实例类型选择，但仅允许一个实例。高可用性运行时环境提供两种实例类型选择，并最多允许两个实例。

        有关更多信息，请参阅 [Amazon EC2 实例类型]，并联系 AWS 大型机专家寻求指导。

6. 在安全和网络部分，执行以下操作：

    a. 如果您希望应用程序可公开访问，请选择允许部署到此环境的应用程序被公开访问。
    b. 选择网络类型。如果您愿意 IPv4， AWS 大型机现代化环境应用程序仅处理 IPv4 请求。在双栈模式下，应用程序将同时 IPv6 处理 IPv4 和请求。如果您选择双堆栈模式，请确保至少有 1 个 VPC IPv6 启用了子网。
    c. 选择一个虚拟私有云 (VPC)。
    d. 如果您使用高可用性模式，请选择两个或更多子网。如果您在 AWS Blu Age 引擎中使用独立模式，请选择两个或更多子网。如果您在 Rocket Software 引擎中使用独立模式，则可以指定一个子网。

e. 为所选的 VPC 选择安全组。

> **Note**
>
> AWS 大型机现代化为您创建了一个 Network Load Balancer，用于将连接分发到您的运行时环境。请确保您的安全组入站和出站规则支持从 IP 地址访问您在应用程序定义的 the section called “### – ##” 属性中指定的端口。有关更多信息，请参阅《网络负载均衡器用户指南》中的更新网络负载均衡器的安全组。

f. 如果要使用客户托管，请在 KMS 密钥字段中选择自定义加密设置 AWS KMS key。有关更多信息，请参阅 AWS 大型机现代化服务的静态数据加密。

> **Note**
>
> 默认情况下， AWS 大型机现代化使用 AWS 大型机现代化为您拥有并管理的数据进行加密。 AWS KMS key 不过，您可以选择使用客户托管的 AWS KMS key。

g. （可选）选择 AWS KMS key 按名称或亚马逊资源名称 (ARN)。或者，选择 “创建” AWS KMS key，进入 AWS KMS 控制台并创建一个新 AWS KMS key的。

h. 选择下一步。

7. （可选）在附加存储空间页面上，选择一个或多个 Amazon EFS 或 Amazon FSx 文件系统。

安装到 AWS 大型机现代化环境的文件系统必须由合适的用户拥有，才能供在 AWS 大型机现代化控制台中运行的应用程序使用。

要配置这些用户设置，您可以将驱动器连接到 Linux Amazon EC2 实例。然后，创建一个 ID 为 `101` 的组和一个 ID 为 `3001` 的用户。此外，请确保将供应用程序使用的必需数据文件夹归该用户所有。

例如，该`myFiles`文件夹可供在 AWS 大型机现代化管理中运行的大型 AWS 机现代化应用程序使用。

```
groupadd -g 101 mygroup
useradd -M -g mygroup -p mypassword -u 3001 myuser
mkdir myFiles
chown myuser:mygroup myFiles
```

> **Note**
>
> 要启用对文件系统的访问，应配置以下安全组规则，以便在 EFS 和 M2 环境实例之间建立网络连接：
>
> - M2 环境安全组：包括支持流量通过 NFS 2049 端口的出站规则。
> - 文件系统挂载目标安全组：包括支持来自实例安全组（如上所列）的流量通过 NFS 2049 端口的入站规则，以及支持流量通过 NFS 2049 端口的出站规则。

8. 选择下一步。
9. 在维护时段部分，选择何时将待处理的更改应用于环境。
   - 如果您选择 “无偏好”，则 AWS 大型机现代化将为您选择优化的维护时段。
   - 要指定特定的维护时段，请选择选择新的维护时段。然后，选择维护时段的开始日期、开始时间和持续时间。

   有关维护时段的更多信息，请参阅 AWS 大型机现代化维护窗口。

   选择下一步。
10. 在审核和创建页面中，检查您输入的信息，然后选择创建环境。

更新 AWS 大型机现代化运行时环境

使用 AWS 大型机现代化控制台更新 AWS 大型机现代化运行时环境。您可以更新运行时引擎的次要版本或托管运行时环境的实例类型。您可以选择是要立即应用更新，还是在首选维护时段内应用更新。

以下说明假定您已完成为 AWS 大型机现代化做好准备 中的步骤。

更新运行时环境

更新运行时环境

1. 打开 AWS 大型机现代化控制台，网址为https://console.aws.amazon.com/m2/。
2. 在 AWS 区域 选择器中，选择创建要更新的环境的区域。
3. 在环境页面上，选择要更新的环境。
4. 在该环境的详细信息页面上，选择操作，然后选择编辑环境。

5. 进行以下任何更改：

   - 在引擎选项部分，选择所需的引擎版本。
   - 在资源部分，选择所需的实例类型。
   - 在维护时段 部分，选择所需的日期、时间和持续时间。

   > **Note**
   > 您可以选择在维护时段内应用的唯一更改是对引擎版本的更改。因此，必须立即应用所有其他更改。

6. 选择下一步。
7. 在何时应用更改中，选择立即或在下一个维护时段内。然后，选择更新环境。

如果选择立即，则当环境完成更新时，您会看到一条消息。

AWS 大型机现代化维护窗口

每个运行时环境每周都有两小时的维护时段。在此时段内，会应用所有系统更改。您可以利用维护时段控制进行修改以及软件和安全修补的时间。如果在指定周内计划了维护事件，则维护会在两小时的维护时段内启动。大部分维护事件也会在两小时的维护时段内完成，但较大的维护事件可能需要几小时以上才能完成。

两小时维护时段是从每个区域的 8 小时时间段中随机选择的。如果创建运行时环境时未指定维护时段，则 AWS Mainframe Modernization 将在随机选择的一星期的某一天中分配 2 小时维护时段。

AWS 在进行维护时，大型机现代化会消耗环境实例中的一些资源。在维护过程中，您可能会观察到对性能的影响甚微，也可能会看到应用程序中断。

停止 AWS 大型机现代化运行时环境

使用 AWS 大型机现代化控制台停止 AWS 大型机现代化运行时环境。停止环境时，当前的应用程序部署将保留，并且在环境重新启动之前，您无需支付该环境的费用。

以下说明假定您已完成为 AWS 大型机现代化做好准备 中的步骤。

停止运行时环境

如果您需要停止 AWS 大型机现代化运行时环境，请按照与更新环境部分类似的步骤进行操作。

使用 AWS 大型机现代化控制台停止 AWS 大型机现代化运行时环境。停止环境时，当前的应用程序部署将保留，并且在环境重新启动之前，您无需支付该环境的费用。

> **Note**
> 必须先停止所有应用程序，然后才能停止环境。

停止运行时环境

1. 打开 AWS 大型机现代化控制台，网址为https://console.aws.amazon.com/m2/。
2. 在 AWS 区域 选择器中，选择创建要停止的环境的区域。
3. 在环境页面上，选择要停止的环境。
4. 在该环境的详细信息页面上，选择操作，然后选择编辑环境。
5. 在编辑环境页面上，找到资源部分，然后将所需容量更新为零。

   > **Note**
   > 要停止环境，仅能选择立即停止。
6. 选择下一步。
7. 在何时应用更改中，选择立即。然后，选择更新环境。

环境容量更新后，您会看到一条消息。

重启 AWS 大型机现代化运行时环境

使用 AWS 大型机现代化控制台重新启动 AWS 大型机现代化运行时环境。重新启动运行时环境时，将恢复对该环境的计费。

重新启动运行时环境

要重新启动 AWS 大型机现代化运行时环境，请按照与停止环境部分类似的步骤进行操作。

重新启动运行时环境

1. 打开 AWS 大型机现代化控制台，网址为https://console.aws.amazon.com/m2/。
2. 在 AWS 区域 选择器中，选择创建要重启的环境的区域。
3. 在环境页面上，选择要重新启动的环境。
4. 在该环境的详细信息页面上，选择操作，然后选择编辑环境。

> Note
>
> 独立环境的所需容量仅能更新为 1。要重新启动运行时环境，仅能选择立即重新启动。

5. 在编辑环境页面上，找到资源部分，然后将所需容量从零更新为所需值。
6. 选择下一步。
7. 在何时应用更改中，选择立即。然后，选择更新环境。

环境容量更新完成并且环境重新启动后，您会看到一条消息。

删除 AWS 大型机现代化运行时环境

使用 AWS 大型机现代化控制台删除 AWS 大型机现代化运行时环境。

以下说明假定您已完成为 AWS 大型机现代化做好准备 中的步骤。

删除运行时环境

如果您需要删除 AWS 大型机现代化运行时环境，请务必先从该环境中删除所有已部署的应用程序。您无法删除已部署应用程序的运行时环境。

删除环境

1. 打开 AWS 大型机现代化控制台，网址为https://console.aws.amazon.com/m2/。
2. 在 AWS 区域 选择器中，选择创建要删除的环境的区域。
3. 在环境页面上，选择要删除的环境，然后依次选择操作和删除环境。
4. 在删除环境窗口中，输入 `delete` 以确认要删除该运行时环境，然后选择删除。

使用 Blu Age 自动重构应用程序 AWS

AWS Blu Age 的自动重构为大型机应用程序的迁移和现代化提供了一种 end-to-end解决方案。重构过程的步骤如下：

- 分析清单
- 分析依赖项
- 自动转换代码
- 捕获和管理测试场景

您可以在 Blu Insights 工具中完成前面的步骤，该工具可通过 AWS 大型机现代化控制台的单点登录获得。有关 Blu Insights 的更多信息，请参阅 Blu Insights 文档。

当您对转换后的源代码感到满意时，就该转到 AWS：使用构建和部署重构后的应用程序了。the section called "AWS 蓝光时代运行时间"

本指南将帮助您了解运行时概念以及如何在现代化项目中使用它们。

主题

AWS Blu Age 发布

AWS Blu Age 引擎有多个版本可供您选择。本页概述了 AWS Blu Age 版本控制的工作原理、每个版本版本有哪些变化、版本升级说明、如何与客户沟通 AWS Blu Age 的更新以及这些版本的生命周期。

the section called “AWS Blu Age 版本控制”页面详细介绍了有关发行版以及如何通过主要版本和次要版本识别每个发行版的信息。该the section called “AWS Blu Age 发行说明”页面包含每个主要版本和次要版本的详细发行说明。 the section called “AWS Blu Age 安全漏洞”页面提到 AWS Blu Age 如何处理常见漏洞和暴露 (CVE)。 the section called “升级 AWS 蓝光时代”详细介绍了 AWS Blu Age 版本的升级说明。并the section called “AWS 蓝光时代生命周期”包括有关 AWS Blu Age Runtime 主要版本生命周期结束 (EOL) 日期的所有细节。

主题

AWS Blu Age 版本控制

AWS Blu Age Transformation 和 Runtime 产品使用兼容 semver（语义版本控制）的方案进行版本控制。要部署您的应用程序，您需要使用与现代化代码兼容的相应运行时版本。如果您对使用哪个版本有疑问，请联系您的 AWS Blu Age 交付经理。

发行版

每个发行版都由一个 **`[Major].[Minor].[Patch]`** 模式标识。例如，在 AWS Blu Age Runtime 版本`4.1.0`中，主要版本为 4，次要版本为 1，补丁版本为 0。

我们打算每月发布新的 AWS Blu Age Runtime 次要版本，并在产品或其依赖项发生重大变化时发布新的主要版本。

有关每个版本中可用的新特征的详细信息，请参阅 the section called “AWS Blu Age 发行说明”。

Alpha 预发行版

每个 alpha 预发行版都由一个 **`[Major].[Minor].0`** 模式标识。

Alpha 预发行版是高频的短期版本，旨在用于现代化改造项目期间的快速迭代。新的 Alpha 预发行版本没有固定的发布节奏，并且会在开发和测试时发布。

有关版本控制、升级和支持的更多信息，请参阅 组件生命周期。

Important

Alpha 预发行版应只在现代化改造项目阶段使用，不得用于生产或关键工作负载。

AWS 蓝光时代运行时间

AWS Blu Age 提供单个 Runtime，可满足您现代化旅程的不同阶段和运营需求。本页描述了它的特性、用例以及如何访问它。

AWS 蓝光时代运行时间

借 AWS 助 Blu Age Runtime，您可以自己部署现代化的应用程序 AWS 账户 ，从而管理自己的基础架构。此选项提供发行版和 alpha 版预发行版本，使您可以灵活地操作以所需方式运行现代化应用程序所需的所有技术组件。您可以在生产环境的稳定版本或用于测试和开发目的的 alpha 预发行版本之间进行选择。

AWS Blu Age Runtime 由客户部署和管理，可以更好地控制运行时环境。它提供自动重构功能，适用于自定义部署方案。

何时使用

AWS Blu Age Runtime 适用于测试和生产环境，在需要对运行时环境进行特定自定义时特别有用。

如何访问

AWS Blu Age 运行时可从 Blu Insights 工具箱访问。

Note

Blu Insights 工具箱的访问权限是你的 AWS Blu Age 项目参与的一部分。

部署

AWS Blu Age Runtime 可在以下平台上部署：

- Amazon EC2
- 亚马逊上的 Amazon ECS EC2
- 亚马逊上的 Amazon EKS EC2
- Amazon ECS 由 AWS Fargate

在亚马逊上部署 EC2 可以直接在实例中完成，也可以通过 Docker 容器化应用程序完成，这是使用 Amazon ECS 或 Amazon EKS 时的首选方式。

有关详细的部署说明，请参阅设置 AWS Blu Age 运行时文档。

AWS Blu Age 发行说明

本节包含 3.5.0 版以后的 AWS Blu Age 运行时和现代化工具的发行说明，最新版本在前，按版本号整理。

对于每个版本， AWS Blu Age Runtime 都已在特定版本的工具（Tomcat、Postgres、Oracle、MQ 等）和依赖项（Spring、Angular 等）上进行了验证。使用其他版本（例如，升级到新版本）需要全面的客户测试和验证。

> **Note**
>
> 有关本文档之前的发行说明，请联系 AWS Blu Age 交付服务。有关 Blu Insights 最新特征的信息，请参阅 Blu Insights 发行版。

版本说明 5.1.0

发布日期：2025 年 11 月 18 日

此版本的 AWS Blu Age Runtime 和 Transformation Engines 对平台功能、性能优化和现代化技术进行了重大增强。一些主要功能和变化包括：

有关此版本所包含更改的更多信息，请参阅以下部分。

- AWS Blu Age 转换引擎和运行时间 GS21

  在MAJI（现代化加速联合计划）合作伙伴 GS21 关系的背景下，增加了对富士通专有大型机的支持。主要功能包括几种新语言，例如PSAM、ADL、ACSGEN或JCL XSP、NDB网络数据库的现代化、AIM、OFM和ACSAPI等 GS21 生态系统的实现以及对JEF编码的支持。
- Java 版本升级

  从 Java 17 升级到 Java 21，提高了安全性和性能，并允许客户部署和运行以更现代的语言实现的应用程序以及使用最新的第三方框架版本

此版本的 AWS Blu Age Runtime 已使用以下堆栈进行了测试：

组件	测试的版本
Java	Java 21
表示层	节点 JS 22.17.1
	Npm 10.9.0
	有角度 20
服务层	Spring Boot 3.5.7
	Spring Core 6.2.12
	春季会议 3.5.2
	Spring statemachine 4.0.0
持久层	PostgreSQL 引擎 15.10
	甲骨文 21c
报告	Jasper 6
应用程序服务器	Apache Tomcat 10.1.40

AWS 蓝光时代运行时间

zOS

改进

- COBOL
  - 增加了对 CEELOCT、CEESECS、CEEDATE 实用程序例程的支持
  - 改进了对 JSON 生成语句和选项 COUNT 的支持
  - 改进了程序司声明中对 BY VALUE 子句的支持
  - 改进了对 COMPUTE 语句的支持，以处理数字文字的指数表示法
  - 改进了对选项 TRANS 和 SYSID 的 CICS START 命令的支持
  - 改进了对处理溢出数字文字的 MOVE 语句的支持

  - 改进了对固定精度类型的除法缩放的支持
  - 改进了对带有 USING 子句的呼叫语句的支持
- JCL-排序
  - 增加了对日期格式 “4MD/” 的支持
  - 增加了对重启关键字的支持
  - 增加了对 DATENS 和 TIMENS 关键字的支持
  - 增加了对 Y2S 格式的支持
  - 改进了对包含多个文件的 OUTFIL 语句的支持
  - 改进了对备注字段的支持
  - 改进了对 OUTREC 语句和 SYMNAMES 绑定的支持
  - 改进了对 INCLUDE 语句的数字和字母数字检查的支持
- JCL-DESNUTILB
  - 改进了使用 GDG 文件作为参数时对 COPYDDN 的支持
- JCL-IDCAMS
  - 改进了对使用单引号定义数据集的支持
- JCL-杂项
  - 增加了对处理通过 JCL 传递给程序的空 PARM 的支持
  - 增加了对 INZUTILB 实用程序的支持
  - 在 GDG DD 语句中添加了对参数化世代号的支持
  - 增强了对具有记录大小的 SYSIN 文件的读取
- Blusam
  - 将属性 openwarmup 的默认值从 true 更改为 false
  - 添加了新的 YML 属性 useBatchInMemoryAtomic 以启用 Redis 批量读取操作 in-memory-atomic 模式
  - 改进了对遇到无效密钥时重写索引数据集的支持
  - 改进了对使用 OID 数据类型创建大型 KSDS 内部表的支持
  - 改进了对获取页面时读取大量 KSDS 数据的支持 previous/next
- SQL
  - 改进了 SQL CONNECT 语句中对主机变量的支持
  - 改进了 SQL 查询中对数据转换的支持

- 改进了对使用双引号标识符的参数化查询的支持
- IMS
  - 改进了 XDFLD 语句中对 NULLVAL 的支持
  - 增加了对 SEG 语句和多个 END 语句的支持
- CICS
  - 改进了对处理 INPUTMSG 选项的 RETURN 命令的支持

AS400

改进

- 角色扮演游戏
  - 在 CTDATA 的 LOOKUP 中添加了对多表的支持
  - 在 CEELOCT API 的反馈代码处理中增加了对*OMIT 参数的支持
  - 增加了对分配给日期、时间和时间戳字段类型的*LOVAL 的支持
  - 增加了将 ISO0 表达式转换为时间戳的支持
  - 改进了索引数组的字节大小计算
  - 改进了对分区和打包异构阵列类型之间的 MOVEA 操作的支持
  - 改进了对在 DS 中定义的数组进行 MOVEA 操作的支持
  - 使用基于外部数据结构的数组改进了对 MOVEA 的支持
  - 使用带有 MOVEA 的变量改进了动态指标的访问
  - 改进了对从数据结构到独立字段的 MOVE 语句的支持
  - 改进了对嵌套内置 %DEC (%SUBST ()) 的支持
  - 改进了对 EXTFILE 关键字的支持，使其仅在未指定库时才使用库列表
  - 使用选项 A、J、P、K 和 CHECK (RB) 改进了对 EDITCODE 的支持
  - 增加了对关键字 *JOBRUN 的支持，用于为日期和时间类型设置分隔符
- CL
  - 增加了对 QRCVDTAQ 命令的支持
  - 增加了对 RNMOBJ 命令的支持
  - 改进了对 CPYFTOIMPF 命令的支持，该命令涉及小于 1 的十进制值和空值
  - 改进了对 CPYTOIMPF 命令的支持，以保留插入顺序，并为不可为空的列中的空字符串输出单个空格

  - 改进了对带有 “QRYSLT” 选项的 OPNQRYF 命令以及与 Java 实体字段名称不同的数据库列的支持
  - 改进了实体字段名称与数据库列名不匹配时对 OPNQRYF 命令的支持
  - 改进了对包含 MAPFLD 值的 KEYFLD 参数的 OPNQRYF 命令的支持
  - 改进了对带参数 MSGTYPE (*STATUS) 和 TOPGMQ (*EXT) 的 SNDPGMSG 命令的支持
  - 改进了对分区不存在时的 CPYF 命令和 MBROPT 选项的支持，以及 CRTF（*是）
  - 通过添加转换前检查和零值回退，改进了对具有数字数据验证和错误处理的 CPYF 命令的支持
  - 改进了对带有 INCREL 参数的 CPYF 命令的支持
  - 改进了对 CPYF 命令的支持，该命令用于处理手动行插入以处理空数值字段
  - 改进了对 SBMOBJ 命令的支持，该命令可在定义用户名时在上下文中检索用户名，否则使用默认值 JOBD
  - 改进了对参数 MSGQ 和 MSGKEY 的 RMVMSG 命令的支持
  - 改进了对 DSPDBR 命令输出格式的支持
  - 改进了对处理本地数据区域的 CHGDTAARA 命令的支持
  - 通过实现正确的大小写处理、默认值和空格填充，同时确保一致的记录编号和成员名称显示，提高了 DSPDBR 命令输出的准确性
  - 改进了对处理 USRLIBL 属性的 RTVJOBA 命令的支持
  - 改进了对处理平面文件的 MOVOBJ 命令的支持
  - 改进了对 RMVM 命令的支持，以将 RangeReference 文件作为参数处理
  - 改进了对平面文件的支持 PgmAlcObj 和忽略锁定操作的 PgmDlcObj 命令
  - 使用平面文件和*LIBL 改进了对 OVRDBF 的支持
  - 改进了对 RTVOBJD 的支持，以返回带磁盘文件的 LIB 名称
  - 增加了对捕获所有未 CPF999 处理消息的消息的支持
  - 改进了 RTVMBRD 在没有 WhereConditions
- 数据库访问
  - 增强了文件覆盖功能以支持 JOIN 文件
  - 增强了处理多个 Select/Omit 条件时 AND/OR 运算符的优先级规则
  - 使用基于时间戳的记录版本控制改进了并发缓存同步
  - 通过批量写入操作改进了对 OVRDBF 的支持
  - 改进了多线程访问时的数据库记录锁定机制
  - 改进了读取操作时的缓存初始化

  - 改进了对提交控制下的删除操作的支持
  - 改进了对包含多格式逻辑文件的主文件的支持
  - 增加了对跨作业的 LIBL DataArea 和 CURLIB 的支持
- 屏幕
  - 改进了字符串值中单引号的转义方式
  - 改进了在处理没有叠加层的写入操作时的 Web 组件清理能力，特别是对于标有 keep 关键字的组件
  - 改进了对关键字 SLNO 的支持
  - 改进了对终端 ID 的支持
  - 改进了对输入字段中大数字的支持
  - 通过按行对显示文件进行列排序和按列对显示文件进行排序，改进了在前端加载表格时的光标定位
  - 增加了对显示文件的 PUTRETAIN（Put-Retain）关键字的支持
  - 改进了在线会话中对终端 ID 的支持
- 打印机
  - 改进了对页码生成的支持
  - 增加了对报表中溢出处理的支持
  - 增加了对打印机文件的 INFDS 关键字的支持
- 任务
  - 改进了在线会话结束时的工作资源清理工作
  - 改进了 Quartz 作业日志记录，在作业属性中包含用户名和格式化的 JobNumber
  - 改进了 JobHistory 创建功能，可以动态使用 NetName 进行交互式会话
  - 改进了对作业提交和日期格式验证的支持
- Misc (杂项)
  - 为 Postgres 添加了对 EBCDIC CP1 047 排序功能的支持
  - 用户空间-改进了 Errc0100 用户空间格式的错误代码参数
  - 将 Jedis 升级到 6.0.0 版，以改进 Redis 中的会话持久性，并扩展了跨多个端点的会话跟踪支持。
  - 当程序 ID 格式包含 “/” 字符时，改进了对组合程序的支持
  - 改进了对定义为程序参数的 DDS 字段的处理
  - 改进了对命令 QMHSND ErrorCode PM 的参数的支持
  - 改进了对从 ISO 到 EUR 的日期转换的支持

AWS Blu Age 转型引擎

zOS

改进

- COBOL
  - 改进了对连续行和多行字符串的支持
  - 改进了对包含多个字段的语句初始化的支持
  - 改进了对包含依赖项中出现语句的抄写本的支持
  - 通过删除数值中的逗号分隔符并为 COMP 字段添加二进制类型支持，改进了文件格式生成
  - 使用选项 BY VALUE ZERO 改进了对呼叫语句的支持
  - 改进对 JSON GENERATE 语句的支持，以处理名称短语
- CICS
  - 改进了对带有 SYNCONRETURN 选项的链接命令的支持
  - 改进了对带有 NODATA 选项的 GET 容器命令的支持
  - 改进了对选项 ROUTECODES 和 NUMROUTES 的写入操作员文本命令的支持
  - 增加了对 INQUIRE TSPOOL 命令的支持
- IMS
  - 改进了 DBD 文件转换中对 SLASHSX 和 DDATA 的支持
  - 增强了对 MFS 地图中打印后标题关键字的支持
  - 为 MFS 文件添加了对 COPY 和 EQU 语句的支持
- SQL
  - 接受 FLOOR 关键字作为列名
  - 改进了对时区字段的支持
  - 改进了 FROM 和 JOIN 子句中对 if 条件和连续子查询的支持

AS400

改进

- 角色扮演游戏
  - 改进了对 DO 操作的支持
  - 增加了对 MOVE/MOVEL 操作时结果指示器的支持

  - 改进了对独立阵列字段的 MOVE 操作的支持
  - 改进了将数值从数组移动到指标时对 MOVEA 操作的支持
  - 改进了对结果字段作为数组的 Z-ADD 操作的支持。
  - 改进了对诸如 SETLL 之类的数据访问功能的*Start 关键字的支持
  - 改进了对使用在外部文件中定义的 byValue 传递的参数调用过程的支持
  - 改进了对 sizePrefixedAlphanumeric 使用*ALL 比喻的支持
  - 改进了对 EXTIND 功能的支持
  - 增加了对子例程中 GOTOs 指令的支持，而子例程中的相应子例程位于主 TAGs 子例程中
  - 改进了 IFs/LOOPs 条件中对 NOT 关键字的支持
  - 通过输入规范改进了字段重命名
  - 改进了对乘法的中间结果 scale/precision 的跟踪
  - 已实现输出规格指标条件
  - 改进了对查找操作的支持
  - 改进了对跨程序指针的支持
  - 改进了对数字输入的 %CHAR 的支持
  - 改进了对与单个参数一起使用的 %dec 内置函数的支持
  - 改进了数据结构字段的初始化
  - 改进了处理*ALL 文字时对二进制整数目标类型的处理
  - 改进了操控性 *LOVAL，包括打包、浮动和双精度类型
- CL
  - 改进了对 DCL 语句的支持，以处理重复事件
  - 改进了对子例程和其他语句的支持
- DDS
  - 改进了压缩类型字段的关键字段检测
  - 改进了物理文件中对 LIKE 关键字的支持
- DSPF
  - 改进了在未明确指定颜色时对 DSPATR 关键字的支持
  - 通过关键字 REFFLD 改进了对字段覆盖的支持
  - 改进了对新设计中没有输入字段的记录输入规范的支持
- PRTF

- 改进后的程序描述了使用空的.prtf 旧文件进行的 prtf 转换
- 改进了 JASPER 模板生成的布局设置
- 改进了使用编辑词（格式化模式）对 AS4 00 输出规范的处理
- 在 RPGLE 输出 EditCode规范中实现了数据格式化功能 EditWord 和负号处理
- 添加了对 “DLTEDT” 关键字的支持
- 增加了对 PRTF 文件输出指示器的支持
- Misc (杂项)
  - 改进了对 PFkey 转换的支持，以便在转换符号常量为 CICS 辅助密钥时考虑编码

AWS Blu Age 转换引擎和运行时适用于 GS21

改进

- 语言:
  - ADL：AIM 描述语言，用于描述 AIM 工件（PEDs、NDB（子）架构、VSAM 架构、AIM 程序等...）
    - 转换为数据库配置记录、DDL 脚本、由数据迁移器处理的 JSON 文件等...
  - ACSGEN：ACS 环境生成器，用于生成系统数据集、控制表、ADL 源、应用程序入口点...
    - 转换为数据库配置记录
  - PSAM：演示服务访问方法，用于描述在线屏幕
    - 变成 Angular 伪影
  - JCLXSP： GS21 XSP 平台上使用的作业控制语言：具有特定功能的 JCL 的特殊语法
    - 转换为 Groovy 脚本
  - GS21 COBOL：对 GS21 特定结构的支持（主要与日语支持有关）
- 数据库：
  - NDB：网络数据库
    - 可自定义转换为现代关系数据库
    - 数据迁移处理 GS21 TDUMP 格式
    - 将网络链接转换为现代 SQL 关系（使用外键和其他列来保留源自旧数据库的 NEXT/PRIOR 顺序）
    - 由新引入的 GS21 运行时支持的现代 Java DAO 层的代码生成
- GS21 生态系统：

  - 目标：高级信息管理器。主 GS21 中间件通过队列支持的消息传递方法处理屏幕、打印机、工作站、数据库和外部通信。
  - ACSAPI：实现 ACS 协议的 GS21 特殊程序，支持屏幕场景（程序相互通信、 PSAM/printer 交互、VSAM 访问等...）（AIM 的祖先）。
  - OFM：在线格式管理，用于使用消息文件（ACS 的祖先）进行程序通信的 GS21 子系统。
  - JXGIJSM、KQCAMS、、JXKUNLOD KDJBR14、JXKRELOD 等...：各种 JCL 实用程序 GS21
  - Support 对操作系统命令的支持
- 特定编码：JEF
  - 双字节编码（与 IBM93 0 的原理相同，但 SOSI 值不同（0x28、0x29））。
  - 自定义 JEF-PU 编码，保留私人使用字符
  - 中使用了相应的字体，以便 webapps/pdfs 以最大保真度再现显示画面

版本说明 4.10.0

发布日期：2025 年 8 月 29 日

此版本的 AWS Blu Age Runtime 和 Transformation Engines 引入了对平台功能、性能优化和现代化技术的增强。一些主要功能和变化包括：

- 增强的 AS/400 库分辨率

  引入了智能库列表管理系统，该系统可在保持最佳性能的同时提供原生 AS/400 库列表行为。系统会根据系统的库列表 (*LIBL) 自动解析对象库，而不会影响应用程序的速度。
- 前端现代化

  前端应用程序已升级到 Angular 版本 20，为用户界面带来了最新的功能和性能改进。

此版本的 AWS Blu Age Runtime 已使用以下堆栈进行了测试：

组件	测试的版本
Java	Java 17
表示层	节点 JS 22.17.1
	Npm 10.9.0

	有角度 20
服务层	春靴 3.4.6
	弹簧核心 6.2.2
	春季会议 3.4.1
	Spring statemachine 4.0.0
持久层	PostgreSQL 引擎 15.10
	甲骨文 21c
应用程序服务器	Apache Tomcat 10.1.40

AWS 蓝光时代运行时间

zOS

改进

- COBOL
  - 通过强大的字节级处理改进了对 JCL-COBOL 参数通信的支持，可防止数组边界异常
  - 改进了副本报表中对 DFHEILBK 和 DFHEIBLC 的支持
  - 改进了对 MOVE 语句功能的支持，以处理预期输出值为零的非数字到数值字段的转换
  - 改进了对使用整数类型参数的 EXAMINE 语句处理的支持
  - 增加了对使用过程指针的支持
  - 改进了 XML 转义字符
  - 增加了对免费语句和改编分配语句的支持
- JCL-IDCAMS
  - 在 REPRO 语句中添加了对关键字 SKIP 的支持
  - 改进了对非 VSAM 文件的 DELETE 语句的支持
- JCL-制冰机
  - 改进了对假人 DCB 参数的处理 SYSUT1

  - 改进了在定义 SYSUT1 为 DUMMY 时对动态控制块 (DCB) 参数处理的支持，确保正确分配数据集并防止在对虚拟输入数据集进行复制操作期间可能发生的处理错误
- JCL-排序
  - 增加了对串联空文件情况的支持
  - 通过更新 ZD（十进制区域）字段的字节计算改进了记录比较
  - 改进了对 JOINKEY-填充和重新格式化字段的支持
  - 改进了串联文件处理，以正确保持输入数据集之间记录大小的一致性
  - 增强了 X'.. '的排序语法和相应的运行时处理 （十六进制文字）、DATE= (...) 和 OUTFIL 的标题和尾部子句中使用的时间关键字。
- JCL-ICETOOL
  - 增加了对某些 COUNT 运算符选项的支持：空、NOTEMPTY、HIGHER (x)、LOWER (y)、EQUAL (v)、NOTEQUAL (w)
- JCL-DESNUTILB
  - 改进了对插入空 DATE 列的支持
  - 添加了在使用分隔符时对 VARCHAR、DECIMAL 和 INTEGER 数据类型的支持
  - 改进了在控制卡中未指定列时对默认值插入的支持
  - 通过从记录内容中排除记录描述符字 (RDW) 字段，改进了对可变长度记录处理的支持
  - 改进了对在 LOAD 操作期间处理数据库列中的空值的支持
  - 改进了对十进制数据类型加载的支持，以确保精确的数字转换
  - 与 “格式分隔” 和 “位置 (*)” 规范结合使用时，改进了对 WORKDDN 参数的支持
- JCL-杂项
  - 增加了对使用基于逗号的语法的 SystemOut 定义的支持，例如 SYSOUT =(,)
  - 改进了对包含特殊字符 # 的 DD 语句的支持
  - 改进了对处理封闭方括号内的多个条件的支持
  - 增加了对用双引号括起的 DSN 语法的支持
  - 增加了对 COPYDDN（名称）语法规范的支持
  - 通过保持保留第一个赋值的类似 z/OS 的行为，改进了对重复的 DD 语句的支持
  - 改进了对使用 SPIN 和 FREE 参数处理 SYSOUT 数据集的支持，从而能够正确缓冲和 end-of-job 打印 SYSOUT DD 内容
  - 在 Groovy 执行 API 中增强了作业的错误报告

  - 通过 YML 属性启用临时文件的动态分配

- Blusam
  - 通过增强元数据清理流程来改善对 CLEAR 操作的支持，以确保完全删除对先前存储的数据的所有引用
  - 启用了写入模式后，提高了写入操作的性能
  - 通过优化表存在性验证流程、消除冗余的查找操作以及通过减少数据库查询来提高整体性能，从而改善对数据集操作的支持
  - 通过添加新的打开模式（仅限数据集级别锁定模式），将批量插入模式设置为 true，从而提高了性能。此配置在数据集级别锁定足够大的自定义数据集还原场景中非常有用。
  - 允许使用 yml ds 配置文件来设置属性，从而扩展了大型 <noloc>Blusam</noloc> KSDS 的配置
  - 在蒙版生成中增加了对图形字母数字的支持
  - 改进了数据集元数据中缺少备用索引时的错误报告
  - 启用异步执行 Redis 批量写入操作以提高性能。该功能可通过 YML 属性进行配置。
- SQL
  - 增加了对 CONNECT 语句的支持
  - 增强了对变量名和运算符表达式 UPDATE 查询的 SQL 支持，涉及游标
  - 改进了 SQL 准备语句中的字符串替换
- 安全性
  - 增强的 API 安全性：启用安全配置后，对所有 gapwalk-应用程序 REST 端点实施了强制身份验证和基于角色的授权。
  - 通过实现自定义 scope-to-role映射改进了 JWT 令牌授权，允许在使用 Cognito 发行的不记名令牌时正确访问受保护的端点。
- Misc (杂项)
  - 增强了 Redis 集成，具有全面的日志记录、错误处理和诊断，包括 50 多项新的上下文检查和 137 条经过改进的错误消息。
  - 改进了首次运行事务时任务的初始化

AS400

新特征

- 增加了对库列表功能的支持，可根据可用库和库列表自动解析库
- CL
  - 增加了对检索数据库文件描述的 QDBRTVFD 命令的支持

- 增加了对用于检查和分析的命令 QCMDCHK 的支持。

改进

- 角色扮演游戏
  - 改进了对 Z-ADD 操作的支持，以处理添加 2 个打包类型数组的情况
  - 使用内置函数改进了对常量初始化的支持
  - 改进了对带空字符串的 EDTWRD 的支持
  - 改进了对 MOVE 语句的支持，以处理分配给不带整数部分的打包字段的负值的情况
  - 改进了对角色和分区类型的 *LOVAL 关键字的支持
- COBOL
  - 改进了对 COBOL 文件状态的支持，以处理字段不是基本范围的情况
  - 改进了在随机模式下检索时文件状态码的设置
  - 改进了 COBOL 函数 DISPLAY-OF 的字符串编码
- CL
  - 增加了对命令 QCMDCHK 的支持
  - 改进了对处理多记录格式的 CPYF 命令的支持 LFs
  - 改进了对 RMVM 命令的支持，该命令可在其他对象依赖成员时删除成员
  - 改进了对参数的支持，可处理命令 RMVM 中的尾随空格
  - 改进了 QCMDEXC 命令，增强了参数处理，包括支持位置参数和正确的参数顺序检索
  - 通过为未指定的 FORMTYPE 参数实现默认值 (*STD)，改进了 OVRPRTF 命令
  - 改进了对 RTVJOBA 的支持，以返回包含 6 个字符的任务编号
  - 改进了对 OPNQRYF 命令和 KEYFLD 参数的支持
  - 改进了对 OPNQRYF 命令和单引号转义的 Unicode 字符的支持
  - 改进了命令 CPYF 的输出文件名的处理
  - 改进了 RTVMBRD 命令，具有增强的视图检索功能，包括支持没有条件的操作以及对行计数进行适当的标题处理
  - 改进了对同时使用 PATH 和 JOB 参数处理呼叫的 DSPJOB 命令的支持
  - 改进了对处理目录中文件的 RTVOBJD 命令的支持
  - 改进了对 ADDPFM 命令的支持，以处理包含尾随空格的参数

  - 改进了 CHGDTAARA 中的线程同步

- 数据库访问
  - 增强了文件覆盖功能以支持 JOIN 文件
  - 增强了处理多个 Select/Omit 条件时 AND/OR 运算符的优先级规则
- 屏幕
  - 改进了两个屏幕共享一个字段名称时的光标位置
  - 改进了对工作站文件的 RRN 字段的支持
  - 处理了来自模态窗口的其他消息屏幕
  - 改进了将消息行显示为 DS4 格式数组
  - 改进了 AS4 00 个应用程序上错误消息的显示以及其他消息信息屏幕
  - 改进了现代化前端中对字符 @ 的支持
  - 改进了前端的会话处理，允许在具有多个选项卡的同一个浏览器上运行多个会话
- 打印机
  - 改进了对具有多个记录标题的打印机文件的溢出处理。
- Misc (杂项)
  - 添加了 Redis 对用户空间的支持
  - 增加了对 API QUSLFLD 的支持，该接口使用用户空间生成指定文件记录格式名称中的字段列表。

AWS Blu Age 转型引擎

zOS

改进

- COBOL
  - 改进了对 CLOB-LOCATOR/DBBLOB-LOCATOR /BLO
  - 改进了对带有缩写二进制条件的 IF 语句的支持
  - 改进了对 DFHEIBLK/EIBLK 处理有复制语句和不带复制语句的支持
  - 通过增强的注释处理和针对多节调用的优化退出逻辑，完善了状态机功能。
  - 改进了对退出部分语句的支持
  - 改进了 INITIALIZE 语句中对 “全部替换” 的
- SQL
  - 改进了对更新查询的 SQL 别名声明的支持

  - 通过忽略结尾和包含关键字改进了对 PostgreSQL 分区创建的支持
- 公用
  - 改进了转换错误的错误报告

AS400

改进

- DSPF
  - 增强了字段长度计算功能，可准确匹配值格式，防止出现显示问题
- PRTF
  - 改进了多页报表页码的生成
  - 改进了报告上的持续显示
- 角色扮演游戏
  - 在阵列上与 Z-ADD 一起使用时增加了对*ALL 的支持
  - 改进了对十六进制类型的比喻常量*ALL 的支持
  - 当指定为 INZ 选项的参数时，增加了对*HIVAL 的支持
  - 处理扩展器 H 以汇总 DIV 操作的结果
  - 改进了对 LOOKUP 函数的支持，以根据所传递的输入数组的种类来处理高、低和等于的指标（ASCENDING 或 DESCENDING）
  - 改进了对 ADD 函数的支持，以便在因子 1、factor2 和结果字段都是数组时执行元素加法
  - 改进了字段声明中对 EDTWRD 函数的支持
  - 改进了对使用输入文件的 CLEAR 操作的支持
  - 改进了定义为独立字段的数组的初始化
  - 改进了对叠加关键字中*NEXT 选项的支持
  - 改进了在 MOVEA 操作中使用关键字*ZEROS 时的支持
- Misc (杂项)
  - 使用改进了对大数的支持 BigDecimal

版本说明 4.9.0

发布日期：2025 年 7 月 17 日

此版本的 AWS Blu Age 运行时和转换引擎引入了核心依赖项的关键更新：

- AS400：除了我们的 DAO 框架中现有的 JPA 支持外，还引入了基于 JDBC 的替代实现。用户现在可以通过 YML 配置在 JPA 和 JDBC 实现之间切换。初步基准测试表明，新的 JDBC 支持带来了显著的性能改进，与 JPA 实施相比，执行时间缩短了大约 50%。
- zoS：引入了 a DB2 z/OS to DB2 LUW migration solution addressing the subtle but critical syntax differences between DB2 z/OS 和 DB2 LUW 环境。尽管数据库语法看起来很相似，但它们的不同要求要求需要对架构管理、表空间和数据类型进行自动转换处理。这可以确保从大型机成功迁移到分布式平台，同时保持操作完整性。

我们使用以下堆栈测试了这个版本的 R AWS Blu Age untime。

组件	测试的版本
Java	Java 17
表示层	节点 JS 22.11.0
	Npm 10.9.0
	有角度 19.1.3
服务层	春靴 3.4.6
	弹簧核心 6.2.2
	春季会议 3.4.1
	Spring statemachine 4.0.0
持久层	PostgreSQL 引擎 15.10
	甲骨文 21c
应用程序服务器	Apache Tomcat 10.1.40

运行时版本 4.9.0

zOS

改进

- COBOL
  - 增加了对 DATE-TO-YYYYMMDD函数的支持
  - 改进了对 UNSTRING 语句的支持以增强计数计算
  - 在 88 级变量上添加了对 DFHRESP（NORMAL）值的支持
  - 改进了涉及 COMMAREA 的绑定操作，尤其是在嵌套程序调用时
  - 增加了对行计数器和页面计数器特殊寄存器的支持
  - 根据通道的机器控制字符改进了对记录顺序文件的 WRITE... BEFORE 语句的支持（C01 到 C12）
  - 改进了对涉及 variable-length-record文件的 READ/WRITE 语句的支持
  - 改进了通过引用将大小相差的参数传递给子程序时对 CALL 语句的支持
  - 当结构有 88 个等级字段时，改进了对排序键的支持
  - 改进了 DISPLAY 语句中对有符号整数值的支持
  - 通过减少计算开销来提高字符串连接操作的性能
  - 改进了对 MOD 和 REM 计算的支持，以处理负值
- PL/I
  - 通过位运算改进了对 SUBSTR 函数的支持
- JCL-IDCAMS
  - 增强了对按复制顺序排列的 GDG 文件的支持
  - 支持在 DATA 组件中定义 KEYS 时覆盖任何集群级别的规范
  - 增加了对 ALTER NEWNAME 语句的支持
  - 为删除语句添加了对 GDG 可选关键字的支持
  - 通过确保 MAXCC 覆盖通过控制命令 (SET, IF-THEN-ELSE) 保持不变，同时在功能命令之前进行适当重置，从而增强条件代码处理
- JCL-排序

- 改进了对使用压缩数据和数字数据组合处理 ZD 字段的支持
- 改进了对 INCLUDE 语句中与字符字段进行分区十进制比较的支持
- 改进了对 INREC FINDREP 的支持，用于处理 OUT 参数中的方括号
- 改进了对 “全部” 和 “无” 条件的 OMIT 语句的支持
- 改进了对 END 语句的支持
- 改进了对 COND 短语的逻辑运算符优先顺序的支持
- 改进了 JOINKEYS 中对未配对的支持

- JCL-DESNUTILB
  - 增加了对双精度数据类型的支持
  - 改进了对使用 COLDEL 关键字加载数据的支持以及对正则表达式特殊分隔符的使用
  - 改进了对 LOAD 语句模板的支持

- JCL-杂项
  - 改进了对基于 LRECL 截断固定区块输出记录的支持
  - 改进了文件删除功能，以防止在高可用性环境中出现问题
  - 增加了对 IKJEFT1 B 系统实用程序的支持
  - 根据先前执行的步骤，改进了对整个 JOB 的记录大小的支持
  - 在 JCL 和 PROC 中增强了对以 PARM 和 STEP 为重点的 PARM 的支持
  - 改进了对 DCB=*.DD 的支持，使其与用于引用先前定义的 DD 信息的向后引用兼容
  - 改进了对使处理过程变量在多线程环境中可用的支持

- <noloc>Blusam</noloc>
  - 改进了在页面或索引边界导航到下一条/上一条记录时对 <noloc>Blusam</noloc> Large KSDS 的读取
  - 改进了大型 <noloc>Blus</noloc> am 文件的清理和加载能力
  - 改进了在缺少可选数据集时为 <noloc>Blusam</noloc> Large KSDS 创建索引表的支持
  - 改进了 <noloc>Blusam</noloc> 写入实现以解决删除操作问题，强制批量更新排队
  - 通过使用本地索引缓存机制提高了 <noloc>Blusam</noloc> 大型数据集记录操作（添加和删除）的性能，从而减少了与远程缓存的多次交互造成的延迟

- SQL
  - 改进了对带有 HAVING 和没有 GROUP BY 子句的语句
  - 改进了数据库架构元数据缓存机制

- 消息队列集成
  - 改进了以多线程方法执行作业时对 MQ CLOSE 操作的支持
  - 通过实现 MQ 操作的 JMSTemplate 重复使用来提高性能

- MFS
  - 增强了对 MFS DEV 语句中的 DSCA 参数的支持

- 屏幕
  - 改进了数据输入机制，防止在双字节字段中输入单字节字符，尤其是在使用虚拟键盘（例如文本合成系统）时。

- Misc (杂项)
  - 为 Redis 连接池添加了更多配置选项，例如`connectionPoolSize`、`connectionMinimumIdleSizeidleConnectionTimeout`、和`connectTimeout`
  - 增强了 /triggerscript/ {scriptID: .+} 端点，支持带有 JSON 有效负载的 POST 请求，除了现有的查询参数外，还允许复杂的参数传递

AS400

改进

- 角色扮演游戏
  - 改进了对总计算中`L0`指标的支持
  - 通过任务初始化改进了对 TIME 操作码的支持
  - 改进了读 DaoCycleManager 取具有不同数量匹配字段的主文件和辅助文件
  - 改进了 %EDITCODE 'J' 数字格式
  - 改进了对编辑代码 4 的编辑代码分组支持

- COBOL
  - 改进了在文件 RECORD KEY 上指定带重复子句时对动态模式下的 REWRITE 的支持
  - 使用 DUPLICATES 关键字在动态模式下启用键字段更新
  - 改进了 SORT 操作的资源处理
- CL
  - 改进了对CHKOBJ命令的支持，以处理对象名称中的尾随空格
  - 改进了对CHKOBJ使用库检查视图是否存在的命令的支持
  - 改进了命令参数列表的解SBMJOB析
  - 改进了对CLRSVF命令和MSGMON在文件不存在CPF9812时引发代码的支持
  - 改进了对用于处理参数*END值的OVRDBF命令的支持 POSITION
  - 改进了对源文件为*REPLACE空CPYF时输入的支持
  - 改进了对不带 VAR 可选关键字的参数处理CHGVAR命令的支持
- 数据库访问
  - 改进了对并发作业的 DELETE 操作的支持
  - 改进了对 DELETE 操作作为对文件的第一个操作的支持。
  - 提高了从联接文件中读取数据时的完整性
  - 改进了 INSERTNULL 标志处于禁用状态时 READ/WRITE 操作中控制字符的处理。
  - 在 SQL 视图中添加了技术列 SRCFILE 和 MEMBER，以提供处理库的机制
  - 改进了对象锁中的剩余超时计算
  - 改进了对 JOIN 文件重写文件的支持
  - 增加了对交易文件上相对密钥的支持
- 屏幕
  - 改进了对同一字段的多条错误消息的处理
  - 处理了发送到任务的外部消息队列时状态消息的显示
  - 添加了消息行服务以动态控制错误消息的显示
  - 提高了其他留言屏幕的消息分辨率

- Misc (杂项)

- 改进了会话跟踪机制，以返回格式正确的上次更新时间戳和创建时间戳
- 提高了数据库中记录时所有 updating/deleting 记录集合的一致性
- 通过引入一个新的 YML 属性来改进作业启动时的 as400 开关的初始化`job.default.encoding`，该属性指定默认编码，用于初始化作业级存储，默认 = `CP1047`
- 改进了 SQL 语法以接受引号转义序列
- 改进了 OVRDBF POSITION 对打包密钥类型的支持。

AWS Blu Age 转型引擎 4.9.0

zOS

新特征

- SQL
  - 增加了对 DDL 和内联查询从 DB2 zoS 转换到 LUW 的支持 DB2

改进

- COBOL
  - 改进了对带有 01 级父字段的 88 级字段的支持
  - 改进了对特殊名称`CTL`和`WRITE .... AFTER CTL`语句的支持
  - 改进了期间从段落到章节的间隔 `PERFORM SECTION`
  - 增加了对带有相对路径的`COPY`语句的支持
  - 添加了对`BODY`作为有效 COBOL 字段名的支持
  - 改进了对`EVALUATE ... WHEN`声明的支持
  - 改进了对`SD`文件中`RECORDING MODE`子句的支持
  - 在语句中添加了对比喻常量的支持 `WRITE`
  - 添加了对 S01 到 S05 和 C01 到 C12 作为有效字段名的支持
  - 为`SELECT ASSIGN`子句添加了对`PRINTER`作为目标文件的支持
  - 在子句中增加了对 REPLACE 语`FD`句的支持
  - 在`FD`子句`COPY`中添加了对数据定义行中的支持
  - 增加了对带有相对路径的`COPY`语句的支持

  - 增强了对`STRING`语句的支持，以支持将数组元素作为分隔符，例如。`STRING ABCDEFG DELIMITED BY IDX-1 (SUB) INTO IDX-2 POINTER ID3`
  - 添加了 Support 对数据结构的 88 级变量`DFHVALUE`值的支持
  - 当参数为 SUBSTRING 表达式时，改进了对`FUNCTION LENGTH`语句的支持
  - 增加了对锁定关闭模式的支持
  - 在 START 语句 OF/IN 中添加了支持
  - 改进了未指定 RIGHT 关键字时对 JUSTIFIED 子句的支持
  - 改进了对 FD 命名的支持，以防止保留关键字发生冲突
  - 改进了对带条件子句的特殊名称的支持
  - 在源计算机配置中添加了对调试模式的支持
  - 改进了对身份识别部门和数据写入可选段落的支持
  - 改进了声明部分的安全退出状态
  - 为顺序文件添加了对无效键语句的支持
  - 改进了对 “工作存储” 部分和 “链接” 部分中重复名称的支持
  - 增强了对 DFHCOMMAREA 的 SET ADDRESS 语句的支持
  - 改进了对 COBOL 声明式调用的支持

- PSB
  - 改进了对`SENSEG`陈述前置`PROCOPT`位置的支持 `PARENT`

- JCL
  - IDCAMS-分别为 REPLICATE、IMBED 和 WRITECHECK 增加了对缩写 REPL、IMBD 和 WRITECHECK 的支持

- SQL
  - 改进了对嵌套表表达式的支持

AS400

改进

- CL

  - 改进了 CL-to-groovy生成时对字段类型的支持
- DDS
  - DPSF RTNCSRLOC 关键字支持多个空格分隔参数
  - 改进了 SQL-DDS 中索引的记录格式名称确定。
  - 改进了 SQL-DDS `' '` 中对引号转义序列的支持。
- 角色扮演游戏
  - 角色扮演游戏
    - 改进了对重写为支持 GOTO 的循环内`LEAVE`语句的支持
    - 增加了对过程校准中使用的`%TIMESTAMP`内置函数的支持
    - 改进了对包含混合大小写的 F-Spec 关键字的支持
    - 改进了对由控制级别指示器控制的代码的支持
    - 改进了对停机指示器的支持
    - 通过控制级别改进了 TAG 支持
    - 通过处理外部记录布局中的重复数据，改进了对`CLEAR`报表的支持
    - 改进了对指标阵列的支持
    - 改进了对 CTDATA 阵列的支持
    - 改进了对基本固定数组分配的支持
    - 在 RPG 计算中增强了对复杂嵌套函数表达式的支持
    - 使用 UDS 增强本地数据区域 read/write 生成
- SQL
  - 增强了对 FROM 子句中带括号的 SQL 语法的支持。

版本说明 4.8.0

发布日期：2025 年 4 月 23 日

此版本的 AWS Blu Age 运行时和转换引擎引入了关键更新，以增强数据库转换能力和性能：

- IBM IMS 数据库转换 — 增加了对将 IBM IMS 数据库转换为我们的 JHDB 定制Blu Age解决方案的支持。
- 内存中缓存功能 — 增加了内存缓存功能，使用户能够在内存中缓存只读数据，从而提高数据密集型程序的性能。

我们使用以下堆栈测试了这个版本的 AWS Blu Age Runtime。

组件	测试的版本
Java	Java 17
表示层	节点 JS 22.11.0
	Npm 10.9.0
	有角度 19.1.3
服务层	春靴 3.4.2
	弹簧核心 6.2.2
	春季会议 3.4.1
	Spring statemachine 4.0.0
持久层	PostgreSQL 引擎 15.10
	甲骨文 21c
应用程序服务器	Apache Tomcat 10.1.17

运行时版本 4.8.0

zOS

新特征

- 引入了对通过 YML 属性`gapwalk-application.security.customAllowedHeaders`为担保交易定义自定义标题的支持。此属性与有价值的`gapwalk-application.identity`属性有关`oauth`。
- 添加了一项功能，允许客户根据 <noloc>Blusam 数据库表中的实际记录重建 <noloc>Bl</noloc> usam</noloc> 元数据
- 使用集成保护 JHDB 连接 AWS Secrets Manager

改进

- COBOL
  - 改进了对读取文件语句的 AT END 条件的支持
  - 改进了对在不移动符号的情况下将带符号的数字文字移动到字母数字字段的支持
  - 改进了对 UNSTRING 语句的支持，以处理非数字到数值字段的转换，并在找不到分隔符且没有值移动到其他变量时保留低值
  - 改进了动态链接实体的地址绑定操作
  - 通过包括对移位 in/out 字节 (SOSI) 的支持，改进了字符串连接操作
  - 改进了对带源代码的 COBOL ACCEPT 语句的 DAY-OF-WEEK支持
  - 增加了对 SYSIN 和 SYSOUT 作为特殊名称值的支持
  - 改进了对 STRING 连接的支持，以处理表单馈送字符
- JCL-IDCAMS
  - 增加了对使用不带 GDG 参数的基本名称删除 gdg 世代的支持
  - 改进了对将 VOLUME 参数作为 DEFINE CLUSTER 语句的标准 VOLUMES 参数的变体进行处理的支持
- JCL-排序
  - 添加了对使用 JOINKEYS 命令的符号名称的支持
- JCL-IEBGENER
  - 通过引入验证检查和自动回退到 ICEGENER，改进了对 IEBGENER 处理无效 SYSIN 内容的支持，确保即使控制语句不正确也能持续运行
- JCL-INFUTILB
  - 改进了对卸载 INTO 和 FROM 子句的处理
- JCL-DESNUTILB
  - 增加了对双精度数据类型的支持
- JCL-杂项
  - 改进了在作业的子过程中使用 DD 名称重写文件配置
  - DSNTIAUL/INFUTILB-增强了对卸载实用程序的 SYSREC 填充字节的自定义
  - 添加了对 DISPLAY 语句重定向的 OUTDD 选项的支持
  - 改进了对固定块顺序文件写入的支持。当 COBOL 计划中的 LRECL 与 JCL LRECL 不同时，JCL LRECL 优先。
  - 为 SORTOUT 文件添加了对目录 DCB 属性的支持，但仅支持永久数据集

  - 增强了对条件 JCL 步骤转换的支持

  - 改进了对带有 SYSIN 的 ACCEPT 和带有尾部空格的超出目标大小的行的支持
  - 改进了对 SET 语句的支持，以处理不包括变量名的命令，内部定义的命令包括一个应该可以在 JCL 中访问的成员文件，命令包含内联注释
  - 增加了在 JCL 检查点元数据和作业上下文中保留作业级别 “ScriptContext” 参数的支持，以便在重启时使用
- <noloc>Blusam</noloc>
  - 改进了加载空数据集后写入时 <noloc>Blusam</noloc> 大型索引数据集上的记录插入
  - 通过启用预热机制和为记录引入可选的预取窗口以及索引和页面的本地存储，提高了大型数据集的性能
  - 改进了对导出数据集的支持，以处理更大的数据集 AWS Key Management Service
- SQL
  - 改进了对 JDBC BLOB 数据类型的支持
- 消息队列集成
  - 在激活 XA 时，在 MQ PUT 命令上添加了对用户标识符的支持
  - 在激活 XA 时增加了对并发 MQ 连接的支持
- IMS
  - 添加了 YML 属性，`jhdb.keepParent`用于控制设置为 true 时 IMS ISRT 调用是否填充数据库中的 _parent 和 _logicalparent 字段
  - 通过删除无效交易代码的消息队列来提高稳健性

AS400

新特征

- 增加了内存缓存功能，使用户能够在内存中缓存只读数据，从而提高了数据密集型程序的性能。

改进

- CL
  - 改进了 OPNQRYF 命令中对 QRYSLT 参数的支持，用于处理 RANGE 关键字、在字符串列具有纯数字值时解析值以及处理空字符串值
  - 增加了对 FTP SENDPASV 命令的支持
  - 改进了对使用 Qcmdexc 执行的命令 DSPDBR 中的查询形成的支持

  - 改进了对 MONMSG 异常的支持，使其在执行堆栈中正确传播
  - 改进了 CPYF 上具有不同列数和数据类型的表之间的记录复制
  - 提高了消息监控优先级：命令级消息现在可以正确优先于过程级消息
  - 改进了对 CPYF 命令的支持，以考虑由 IVLIB 定义的库 BLU4
  - 改进了对命令 MonMsgs 中 CHKOBJ 的支持，以处理库中不存在文件的情况
- 屏幕
  - 通过手动构建 BigDecimal 值然后将其转换为要显示的数字，而不是让数字四舍五入，改进了对超过 15 位数的值的处理
- Misc (杂项)
  - 改进了对 UserSpace 已存在时 UserSpace 使用 “替换为 NO” 的支持
  - 增强了 BLU4 IVLIB 功能，支持检索和向被覆盖的库写入记录
  - 使用 DIM 和 ACCURES 操作改进了对多维数组字段的支持
  - 增强了对在 DAO 操作中作为关键字段值传递的低值的支持
  - 删除了本地和全局监视器可以处理的消息在 MONMSG 指令上的冲突
  - 改进了 “插入重复记录” 的设置 FileStatus
  - 通过首先使用注册表来检查用户空间是否存在，减少了用户空间锁定冲突并降低了创建失败的风险
  - 改进了对 () 在删除 readPriorEqual () 后跳过记录的支持 ()
  - 改进了承诺控制下的记录锁定机制
  - 改进了子例程中对 RPG 语句 RETURN 的处理
  - 改进了中对用户信息的支持 SharedContext

AWS Blu Age 转型引擎 4.8.0

zOS

改进

- COBOL
  - 改进了多个文件的打印机模式生成。
  - 改进了对 SPECIAL-NAMES 的支持，以处理可选的结尾点和多个点。
  - 使用相对路径 /REPLACE OFF/COPY-REPLACE 语句改进了 COPY 支持。
  - 添加了对 COL 作为有效字段名的支持。

  - 改进了解析功能，可以处理 SPECIAL-NAMES、X-COMPUTER 和声明之后的空格。
  - 添加了对 S01 到 S05 和 C01 到 C12 作为有效段落名称的支持。
  - 添加了对 LENGT 函数作为 LENGTH 函数别名的支持。
  - 当 cobolMaxCode宽度配置值与 COBOL 包含器不同时，改进了 Copybook 的解析能力
  - 改进了对使用逗号和运算解析表达式的支持，例如 MIN (A * B、(3 + 1) /2、3 + 4)。
  - 在 PROGRAM ID 子句中添加了对 IS 初始程序的支持。
  - 在 SEARCH ALL 语句中添加了对 88 级条件的支持。
  - 增加了对报告部分的支持。
- SQL
  - 增加了对从 IBM IMS 数据库进行的 DDL 转换的支持。

AS400

改进

- CL
  - 通过将转换设置为 JAVA 作为默认支持，增强了对 CL 的支持
  - 通过添加元模型的使用改进了向 groovy 脚本的转换
  - 在保留关键字列表中添加了关键字 CONTINUE 和 FOR
- DDS
  - 与之前定义的字段相比，改进了对定义的 DSPF 字段的支持
  - 改进了对带有关键字 VARLEN 的 DDS 字符字段的支持
  - DAOs 通过在转换 transformation/generation 和更新 JPAFrom Vo 方法中生成重命名字段的显式设置来改进 LF 实体。
- 角色扮演游戏
  - 改进了对 PREFIX 关键字参数的支持，以处理值没有引号的情况
  - 改进了对数组索引中嵌套方法调用的处理
- COBOL400
  - 使用指令 COPY 改进了从 DSPF 导入的记录实体的生成 DDS-ALL-FORMATS
- SQL
  - 改进了带有非封闭式 OR 条件的 SQL WHERE 子句的转换。

版本说明 4.7.0

发布日期：2025 年 3 月 12 日

此版本的 AWS Blu Age Runtime 和 Transformation Engines 引入了对核心依赖项的关键更新： AWS Blu Age Runtime 已升级为使用最新的 Spring Boot 版本 3.4.2，现代化的前端应用程序现在使用最新的 Angular 版本 19。

我们已经使用以下堆栈测试了这个版本的 AWS Blu Age Runtime。其他版本可能也兼容。

组件	测试的版本
Java	Java 17
表示层	节点 JS 22.11.0
	Npm 10.9.0
	有角度 19.1.3
服务层	春靴 3.4.2
	弹簧核心 6.2.2
	春季会议 3.4.1
持久层	Spring statemachine 4.0.0
	PostgreSQL 引擎 14
甲骨文 21c	
应用程序服务器	Apache Tomcat 10.1.17

运行时版本 4.7.0

zOS

新特征

- 添加了新的 YML 属性`dataSimplifier.doubleFloatingType`来定义编码和解码浮点数时要使用的格式-允许的值为`IEEE_754`（默认）/`HFP`（对于十六进制浮点格式）

- 增加了对数字图片上的十进制缩放位置 P 的支持

改进

- COBOL
  - 改进了对 INSPECT 语句的支持，其中包含初始语句之后和初始子句之前
  - 升级了对数字编辑类型的支持，优化了字节转换，提高了整体数据处理效率
  - 增加了对带字母数字比较功能的最小值和最大值的支持
  - 改进了对负值函数整数的支持
- PL/I
  - 使用 FBA/VBA SYSPRINT 数据集增强了对 PUT 语句的支持
- JCL-排序
  - 增加了对 OUTREC 选项 SQZ 的支持，子参数为 SHIFT、LEAD、TRAIL、MID、PREBLANK 和 PAIR
  - 增强了 SORTOUT 文件的编目 DCB 属性
- JCL-DESNUTILB
  - 增强了对十进制外部数据类型的支持
  - 改进了对与另一列的值/位置相关的 NULLIF 条件的支持
  - 改进了对 BYTEA 列的支持
- JCL-INFUTILB
  - 改进了在卸载操作中对关键字 CURRENT [DATE/TIME/TIMESTAMP] 的支持
- JCL-改进了对被调用者的限定返回码的支持 PROCs
- 数据处理
  - 在处理从字母数字类型重新定义为数字类型的空值字段时，增强了数值比较逻辑，可确保对零进行正确的求值
- <noloc>Blusam</noloc>
  - 改进了对 START 命令的支持，以使用主键段处理部分按键搜索，从而提供更灵活的记录检索功能
- SQL
  - 增加了对在 ON 子句中传递的输入参数的支持
  - 增加了对重复功能的支持
  - 改进了对包含列名的赋值的 UPDATE 语句的支持

  - 改进了多行游标读取中的 SQLCODE 和 SQLSTATE 处理
  - 增加了对 DB2 数据库目标 DB2 的 XML 函数的支持
- 消息队列集成
  - 增强 MQ GET 操作中的错误处理，为空消息正文提供更清晰的状态指示 MQCC_FAILED
  - 通过在 XA 事务上下文中处理 MQRFH2 标头，增强了对 MQ PUT 操作的支持
  - 改进了对 AWS Blu Age 运行时密钥的支持，以处理 JMS MQ 的其他属性
  - 增强的 JMS MQ 连接池配置容量
- Misc (杂项)
  - 系统集成-在 DFSRRC00 程序调用中增加了对架构环境的支持
  - 增加了禁用 <noloc>Blusam</noloc> 时的 IMS 事务的兼容性
  - 改进了对顺序文件 end-of-file条件的支持，以符合旧规则 “EOF 被视为读取失败”

AS400

新特征

- 增加了对 RUNSQLSTM 命令的支持，用于执行存储在外部源文件中的 SQL 查询

改进

- CL
  - 通过 ALCOBJ/DLCOBJ 命令添加了文件锁定机制
  - 改进了 CPYF 命令功能，以便在文件复制期间保持记录序列的完整性
  - 改进了 CPYF 命令以支持以平面文件形式输出
  - 增强的参数处理功能，可在程序调用期间自动调整数据长度
  - 改进了用于处理范围参考的参数支持
  - 在 DLTDTAARA 命令中添加了对 MONMSG CPF21 05 的支持
  - 在 DATAAREA 上添加了对 QTEMP 库对命令 CHKOBJ 的支持
  - 增强了对 ADDLFM 命令的支持，提高了逻辑文件成员管理功能
  - 在 DLTDTAARA 命令中添加了对通用数据区名称的支持
  - 增加了对 SBMJOB 的支持，以处理通过 SWS 参数传递的开关
  - 通过将读取和写入调用放入同步块，改进了数据区写入逻辑

  - 增强了数据区域并发管理，通过实施精细的锁定机制来防止同时写入，从而消除了并行操作期间的错误
  - 在 RTVDTAARA 命令的 DTAARA 参数中添加了对关键字 *ALL 的支持
- 屏幕
  - 改进了前端白色属性的颜色
- 角色扮演游戏
  - 改进了对 TESTB 操作的支持，以更好地处理不同的编码
  - 使用编辑代码 'Y' 和 'P' 改进了对 EDITC 操作的支持
  - 改进了对代码 K、Q 和 Z 的 EDITCODE 的支持
  - 在带有参数的本地过程上实现对 Pro ExtProc totype 关键字支持的支持
  - 为外部过程按值传递的已处理参数
- COBOL
  - 改进了 SORT 语句以处理目标库
  - 在输出模式下打开关联的 DAO 时截断了分区
  - 增加了对 PREFIX 关键字的支持
- DataQueue
  - 为 RabbitMQ 数据和混合消息队列添加了密钥库和信任库配置选项，从而在生产部署中实现安全消息传递
  - 通过消除冗余的数据队列存在检查来优化服务器重启性能
  - 通过移除 DataQueueRegistry 和 DataQueueConnectionProvider 组件来简化架构
  - 无需注册表验证即可通过直接处理简化数据队列的删除
  - 通过确保正确的交换创建和 queue-to-exchange绑定来改进队列创建流程
- Misc (杂项)
  - 添加了事务请求大小验证，以防止因请求范围过大而导致系统过载
  - 改进了作业资源清理的锁定管理
  - 改进了 DAO 锁定机制，以防止在输入模式下打开文件时锁定记录
  - 优化了动态实体映射的性能
  - 改进了 JPA 持久性上下文的处理以提高 DAO 操作的性能
  - 通过添加时间戳列来处理记录元数据，改进了数据区域支持
  - 通过实现同步读写操作来防止出现并发访问问题，从而提高数据区域的可靠性
  - 改进了对 DateHelper #moveDate 的支持以支持 date-to-timestamp转换

- DDS
  - 改进了对 Select/Omit 条件的支持，并结合了综合操作，例如 SST 或 CONCAT
- SQL
  - 优化了 SQLExecutor生成器以更有效地处理部分记录读取，尤其是在检索多行且读取限制超过一行时

AWS Blu Age 转型引擎 4.7.0

zOS

改进

- COBOL
  - 增加了对 CODE-SET 语句的支持
  - 添加了对 WRITE 语句中非页尾选项的支持
  - 在 ALTER 上添加了对新语法的支持
  - 为 COPY 替换语句添加了对多行替换的支持
  - 改进了对不同子句定义中隐含的 CHARACTERS 关键字的解析
  - 改进了对输入输出部分中涉及的备用按键的支持，当按键的名称重复但具有不同的父项时
  - 通过将大字段与空格进行比较，改进了对 IF 条件的支持
  - 通过使其不区分大小写来改进逻辑文件引用静态分辨率
  - 增强了代码转换，可为每个包含 LINAGE 子句的 FD 条目自动生成 LINAGE-COUNTER 隐式字段
  - 改进了对 IDCAMS 集群定义的支持，名称用简单引号括起来
  - 改进了对基本格式的 PERFORM 语句的支持，《泰晤士报》短语格式和格式涉及章节限定
  - 启用了包含多个 GIVING 文件的 SORT 语句的转换，并添加了一个异常以在执行时跟踪这种情况
  - 改进了使用 REDEFINES 子句在群组上指定重命名时的转换
- PL/1
  - 为 PUT STRING 添加了对选项 DATA 或 LIST 的支持
  - 增加了对多项入口计划支持的支持

AS400

改进

- CL
  - 改进了 QCMDEXC 参数的解析，使其能够通过 QCMDEXC 执行 ALCOBJ/DLCOBJ
  - 添加了对 CL ElseIf 语句的支持
  - 添加了对内置函数 %SWITCH 的支持，用于设置作业切换。
- DDS
  - 改进了对程序描述的 DSPF 记录的支持，以处理输入规范中的虚拟记录
  - 改进了 CREATE INDEX SQLDDS 文件的 DAO 生成 WHERE 子句条件
- 角色扮演游戏
  - 增加了对无符号整数上的 ZEROES 初始化 INZ (*ZEROS) 的支持
  - 改进了对命令的支持，但工作站和 SQL 索引文件类型除外
  - 改进了对外部定义的数值字段的 RPG 内部数据类型的支持
  - 改进了对内置 %ERROR 的支持，可在对文件进行操作后获取当前错误状态
  - 改进了对 COMMIT 文件关键字的支持，以处理多格式逻辑文件
  - 增强了对多格式逻辑文件的 SELECT/OMIT 处理
  - 改进了 MOVEA 的处理，用于将数组设置为空白，将 Packed、Zoned 和 Binary 的固定数组移动到另一个相同类型的数组。
  - 改进了 Z-ADD *ZEROS 和 MOVEA *ZEROS 的数组处理
  - 改进了从数字和字符组合移动时的 MOVE/MOVEL 操作
  - 在带有参数的本地程序上实现对 RPGLE Pro ExtProc totype 关键字的支持
  - 改进了赋值语句中*HIVAL 比喻常量的处理
  - 改进了对 PREFIX 关键字的支持以处理字符替换
- COBOL400
  - 改进了对动态访问模式下重写和更新操作的支持
  - 在 CBLLE 中增加了对 ACCEPT 语句 FOR 子句的支持
  - 改进了对内置 %STATUS 的支持，可在操作后获取文件的当前状态
- Misc (杂项)
  - 改进了定义与指定 PF/LF 中某个字段同名的数据结构的程序的转换
  - 改进了汉字变量的数组访问生成

版本说明 4.6.0

发布日期：2025 年 1 月 24 日

我们已经使用以下堆栈测试了这个版本的 AWS Blu Age Runtime。其他版本可能也兼容。

组件	测试的版本
Java	Java 17
表示层	节点 JS 22.11.0
	Npm 10.9.0
	Angular
服务层	Spring Boot 3.5
	Spring Core 6.1.14
	Spring statemachine 4.0.0
持久层	PostgreSQL 引擎 14
	甲骨文 21c
应用程序服务器	Apache Tomcat 10.1.17

运行时版本 4.6.0

zOS

改进

- COBOL
  - 增强`WRITE ADVANCING`功能，提高了顺序文件行写入的准确性，支持多个上下文（`BEFORE`> `AFTER`、和隐式用法）和完整的`PAGE`语句实现
  - 增强了`FILLER`对嵌套表 FILLER 用作组且表作为子表的情况的支持
  - 改善了在一个区段内接触父母模棱两可的孩子的机会
  - 添加了对带有 picture='-----' 的数字编辑类型的支持

  - 改进了 BINARY 类型数据的显示处理
- PL/I
  - 改进了赋值语句中二进制文字值的转换
- JCL — 排序
  - 改进了对同一OUTFIL语句中连续OVERLAY参数的支持
- JCL — DSNUTILB
  - 优化加载机制，使数据检索时间缩短 25%
  - 增强了对外部业务数据源的 XA 事务的支持
- JCL — INFUTILB
  - UNLOAD-增加了对 FLOAT8 数据类型的支持
- JCL — IDCAMS
  - 优化了对IDCAMS命令返回码的处理
  - 增加了根据 GDG 基本名称删除所有 GDG 世代的支持
  - 增加了对不带NONVSAM参数的文件删除的支持
- JCL — 杂项
  - 增强了 Batch Restart 元数据处理功能，可改善重启模式下的工作流程状态管理
- <noloc>Blusam</noloc>
  - 在 Ehcache 和 Redis 实现中都增加了<noloc>对 Blusam</noloc> 缓存的 TTL 支持
  - 改进了对 B <noloc>lus</noloc> am KSDS 文件的 COBOL 文件描述中DEPENDING ON字段FD的支持
  - 增强了 Redis <noloc>Blusam</noloc> 读取操作中的线程安全性，可同时执行多作业
  - 改进了 <noloc>Blusam</noloc> 架构的创建，提高了数据库用户权限的稳健性
  - 改进了可变块串联输入数据集右侧的填充 READ
- BAC
  - 增加了对在多架构模式下创建数据集的支持，包括一个新的 “架构” 列，用于指示每个数据集的架构关联
- MFS
  - 改进了用户信息从前端到共享上下文的传播，确保正确传播到 JHDB 上下文
  - 添加了对 XA 事务上的 IBM MQ IMS 信息标题的支持
- SQL
  - 增强了SQLCODE处理能力，可在所有列值均为 NULL 时在游标读取期间设置 305

  - 增加了对涉及WHERE条件OCCURS参数的IN子句的支持
  - 增加了对DECLARE GLOBAL临时表语句的支持
  - 通过在执行时根据目标数据库引擎进行专用转换，扩展了对午夜 24 小时 DB2 特定时间戳格式的 DB2 SQL 支持
- Misc (杂项)
  - 增强了 IBM93 0 字符集，允许 Unicode 字符 U+2014 和 U+2015 与 EBCDIC 中的 x'44x4a' 相对应
  - TDQUEUE-重构了 SQS 实现以支持多线程
  - 改进了 GDG 数据集名称解析，允许客户存档具有相同 GDG 前缀的文件（例如，A.B.C.G0002V00是当前文件并且A.B.C.G0001V00.1236是存档文件）
  - SQLConverter::toPgmDate/Time/Timestamp已增强，可根据传统格式调整日期计算

AS400

新特征

- 增加了对为平面文件和重复实体动态创建的 AS4 00 表的支持，允许访问通过 CRTPF、CRTDUPOBJ 和 CPYF 等 CL 命令创建的表
- 通过注册表添加了一项支持库列表的服务，该注册表处理每个表的默认库

改进

- CL
  - CLRPFM-改进了在 QTEMP 库中调用命令时对成员的处理
  - SMBJOB-改进了对参数的支持PARM，以处理动态构建的参数
  - CPYFRMIMPF-增加了对参数、和的支持 TIMFMT ERRRCDFILE ERRRCDOPT
  - CPYFRMIMPF-改进了对包含单引号的数据库字母数字值的支持
  - CPYF-改进了多FROM成员文件的命令查询结构 TOMBR(*ALL)
  - CPYF-增强了对处理FMTOPT参数的支持 MAP DROP
  - CPYTOIMPF-改进了对表格处理参数FROMFILE的支持 MEMBER
  - RTVUSRPRF-增加了对参数的支持 RTNUSRPRF
  - DSPDBR-彻底修改命令以匹配预期的传统行为，即打印出有关表中存在的视图以及它们所属的库和成员的信息

  - DSPFD-改进了对参数的支持 `FILE`
  - dspfd-改进了对参数`TYPE MBR`输出的支持以包括其他值：mbfile、mblib、mbfcdt、mfccn
- 屏幕
  - 改进了的光标位置优先级 `DSPATR(PC)`
  - 通过忽略 “受保护” 字段的前端验证，改进了子文件记录字段的验证
  - 改进了对使用多个数组字段共享组件名称的工作站中初始化记录的支持
  - 增强了对`DSPF`关键字（`SFLMSGSFLMSGID`、`CHANGE`和命令键）中响应指示符的支持
- 角色扮演游戏
  - 增强了对程序周期的支持，可以更好地处理从 primary/secondary 文件中读取的字段
  - 增加了对用于 primary/secondary 文件读取的分割控制字段的支持
  - 增强了处理比较语句中双字节字段的`%SUBST`内置方法
  - 改进了对 MVR 操作的 ZERO 指示器的支持
- DDS
  - 增加了对具有记录格式的多格式逻辑文件的支持，这些文件引用相同的物理记录
- DataQueue
  - 通过在中断期间清理使用者，改进了等待数据队列消息的作业的作业中断处理
  - 从 RabbitMQ 迁移到 Spring-amQP 以实现更好的频道管理和线程扩展
- Misc (杂项)
  - 改进了 SQLExecutor Builder，支持包含多个空格和不带前导空格的左大括号的查询
  - 改进了 DAO 支持，可在更改读取方向时正确处理光标位置
  - 改进了检索和删除操作后的密钥初始化，以确保在插入更新的记录之前正确删除相关记录
  - 优化 DAO 映射器生成的代码以提高时间执行性能

AWS Blu Age 变身引擎 4.6.0

zOS

改进

- COBOL
  - 使用可选`AREA/AREAS`文字改进了对`RESERVE`子句的解析
  - 通过可选`DATA DIVISION`声明增强对 COBOL 的支持，支持简化的测试用例

  - 通过添加对、、和CLASS子句ALPHABETSYMBOLIC、开关和FORMFEED变量的支持，改进了特殊名称段落
  - 添加了对语句中SYSIN作为助记符名称的支持 ACCEPT
  - 增强了PIC逻辑大小计算中对 “$”、“0”、“CR”、“DB” 符号的PICTURE子句支持
  - 改进了多个文件场景的USE语句转换
  - 增强了多项变更的ALTER语句转换
  - 在子句中添加了对比喻常量的ZEROHIGH-VALUELOW-VALUES支持 delimited by
- SQL
  - 改进了 PostgreSQL 目标的默认值的转换，以处理默认值周围的CURRENT_TIMESTAMP引号
  - SQL 视图的处理WITH CHECK OPTION子句

AS400

改进

- DDS
  - 改进了对多次引用同一物理记录的多格式逻辑文件的支持
- 角色扮演游戏
  - 增强MOVE了MOVEL操作以更好地处理填充零
  - 增强了对评估和条件中嵌套函数调用的处理
- COBOL400
  - 增加了对在SELECT语句中转换IN关键字的支持
  - 改进了对数据描述条目中缺失点的支持，与最新的 COBOL 版本保持一致，其中缺少点时假设点
  - 增强了REWRITE操作时的光标定位
  - 增强了对START语句的支持，可将记录锁定在当前文件位置
  - 改进了对生成所有 input/output 数据结构COPY DDS的编译器指令的支持
- Misc (杂项)
  - StateMachines -改进转型，以增强符合 stateless4j 范式的复合州声明
  - 改进了对包含特殊字符的 LF 文件的清理
  - 改进了对*ALL带有十六进制值的比喻的支持
  - 改进了对从数字类型到字符类型的隐式转换的MOVE操作支持

- 优化了报告 Bean 的生成，可按关联的打印机名称进行排序，防止名称重复或冲突
- 改进了对关键字的支持，`USROPN`并`EXTFILE`结合起来处理文字值和格式 `libname/filename`

版本说明 4.5.0

发布日期：2024 年 12 月 20 日

此版本的 AWS Blu Age Runtime 和 AWS Blu Age 转换引擎包括以下主要功能。

- JCL 支持 — 现在可以在运行时上下文中即时生成和执行 JCL 脚本。此功能增加了批处理作业的灵活性和自动化程度。我们在运行时更新了对 JCL 实用程序的支持，对 SORT、ICETOOL、INFUTILB 和 IDCAMS 进行了一系列改进（详见以下章节）。这些增强功能提供了更强大、更高效的数据处理能力。
- 绑定目录和激活组支持 AS/400 现代化应用程序 — 绑定目录通过管理导出的过程引用来增强系统组织，而激活组则简化了执行环境管理。这些功能提高了精度和可靠性，增强了资源管理，并优化了系统交互。结果是为现代化的 AS4 00 应用程序提供了一个更具弹性、更有条理和更高效的系统。
- 依赖项更新：— 将所有前端框架（BAC/JAC 和现代化应用程序）更新为长期支持 (LTS) 版本。Angular 从 v17 更新到 v18 引入了新的反应性模型和简化的状态管理，从而降低了复杂性并改善了开发人员的应用程序维护。Node.JS 也已从 v20 更新到 v22。

我们已经使用以下堆栈测试了这个版本的 AWS Blu Age Runtime。其他版本可能也兼容。

组件	测试的版本
Java	Java 17
表示层	节点 JS 22.11.0
	Npm 10.9.0
	Angular
服务层	Spring Boot 3.5
	Spring Core 6.1.14
	Spring statemachine 4.0.0

持久层	PostgreSQL 引擎 14 甲骨文 21c
应用程序服务器	Apache Tomcat 10.1.17

运行时版本 4.5.0

zOS

新特征

- JCL — 增加了从在线程序调用批处理作业的功能。当现代化程序即时生成 TDQueue 时，我们添加了一项服务来处理存储在专用文件中的 JCL 脚本。该服务使重建 JCL 消息成为可能，将此消息重构为 groovy 脚本，并运行这个 groovy 脚本。
- ADABAS — 增加了对 ADABAS 程序的支持。有了这种支持，运行时可以模拟 ADABAS 命令进行数据库访问（仅适用于 Oracle）。

改进

- COBOL
  - 利用 “不推进” 选项增强对 DISPLAY 语句的支持
  - 提高了货币符号管理的准确性，使用户能够从更精确的转换后的 COBOL 结构中受益
  - 改进了在将无符号字段移动到有符号字段时对值分配的支持，反之亦然
  - 改进了对 GDG 文件和串联文件块大小的支持
- CICS
  - 增加了对 <noloc>Blusa OpenStatus m 数据 EnableStatus 集和数据集</noloc>的支持
  - 增加了对该`SET DATASET`命令的支持
- JCL — 排序
  - 改进了对数据集记录大小的处理
  - 改进了对`OUTFIL`语句的支持，即根据 STARTREC 和选项中指定的值生成仅包含输入文件中记录的输出文件 `ENDREC`
  - 增强对`OVERLAY`报表的支持
  - 改进了对`OUTREC`语句的支持，以处理该`EDIT`选项的变体。除了`EDIT(...)`，我们现在还支持`EDIT=(...)`

- 增加了对算术运算中模式（p、m、f、OPERATOR、p2、m2、f2）的支持
- 你可以使用 JCL 中`SORT`程序的 DUMMY file 子句来处理空的输入文件，并从空文件生成中受益

- JCL — ICETOOL
  - 通过该`SORT`计划增强对`SORT FIELDS=COPY`声明的支持
- JCL — INFUTILB
  - 如果未在 JCL 中指定记录大小并且属性 DFSIGDCB 被禁用，则改进了对记录大小计算的支持
  - 通过根据 into 子句字段更新精度和小数位数，改进了 DECIMAL 的 INTO 子句的 UNLOAD
  - 改进了中的格式化方法 VarcharFormatter
  - 通过新的可配置选项增强了支持，该选项允许用户控制在数据卸载期间如何处理 VARCHAR 字段的填充行为，从而确保数据提取过程的灵活性和准确性。
- JCL — IDCAMS
  - 改进了带有通配符后缀和名称的文件的删除功能，这些文件可以直接用括号括起来，也可以用简单引号括起来
  - 提高了利用 MAXCC 返回码的准确性
- JCL — IKJEFT01-添加了功能标志`systin.encoding`（默认 =`ASCII`）以支持 SYSTSIN 文件数据集的特定编码
- JCL — 改进了对 JCL 步骤中生成的输出文件的 BDW 属性的支持，后续步骤使用与输入相同的文件系统和 `DISP=PASS`
- MF
  - 改进了对记录顺序文件的 2 字节标题的支持
  - 改进了对 DELETE 命令返回码的处理
  - 改进了记录顺序文件的写入前进行
- Redis
  - 改进了 JCL 检查点和 Jics 的 Redis 模板初始化 TSQueues
  - 改进了 Redis 数据集记录锁定信息的可访问性和可读性
- SQL
  - 使用引用子句改进了对外键的解析
  - 提供了可扩展的缓存功能，用于在数据库中存储原始的传统图形类型，从而增强了数据可追溯性并简化了图形计算
  - 增强了对跨运行时实用程序的 CASE WHEN SQL 查询模式的解析支持

  - 改进了运行时所依赖的 SQL Postgres Blu Age 内置函数 gwdecimal 以适应 DECIMAL 内置函数 DB2 。
- Misc (杂项)
  - 增强了对 NumericEditedType 使用 SIGN 操作数的支持
  - 改进了现代化应用程序 SpringBootLauncher 中的主数据源配置生成
  - 提高了将应用程序日志与与被调用作业相关的路径分开的灵活性。
  - 改进了在比较来自的字段时对空白值的支持 NumberUtils
- FILE — 改进了对基础文件中可变块数据集的支持
- MQ — 改进了 MQ 连接管理，为高可用性环境做好准备
- 通过添加对非 JMS 客户端的支持来增强编码和字符集处理，从而改进 MQ 队列的兼容性
- 改进了对 Ebcdic 文件的 ANSI 控制字符的支持

AS400

新特征

- 增加了对绑定程序中导出数据的支持
- 增加了对除以零的 ILE 特定支持

改进

- COBOL400
  - 改进了 “文件状态” 中对 EOF 的支持
  - 增加 Cobol START 语句的精度支持，以支持 KEY IS 子句中的 EQUAL 关键字
- CL
  - 增加了对命令 UPDENVPARM 的支持
  - CRTPF-增加了对使用分区访问的表的支持
  - RCVF-改进了对带覆盖功能的逻辑文件的支持
  - FTP-使用 OVRDBF 和增强的输出日志改进了对逻辑 I/O 文件的支持，并增加了对工作目录中 I/O 文件的支持
  - CPYFRMIMPF-增加了对参数、、的支持 `ERRRCDFILE` `TIMFMT` `ERRRCDOPT`
  - CPYF-改进了 QTEMP 分区的创建
  - CPYF-添加了*FROM 文件为空时的监控消息

  - OVRPRTF-增加了对新参数的支持：PAGESIZE、、、OUTQ、、DEV、LIP、、CPI、OVRFLOW LVLCHK FORMTYPE HOLD
  - 在CPYF命令中使用带MAP和DROP选项的FMTOPT参数以允许将包含额外列的源文件中的数据复制到目标文件时提高了准确性
  - 提高了管理命令中文件系统路径通配符模式映射的RMVLNK准确性
  - RMVM（移除虚拟机）命令已得到增强，可以处理DROP分区表，确保完全清理相关资源。
  - OPNQRYF-改进了对命令参数*FILE 的支持
  - 实现了 CPF0000 处理以涵盖所有 CPFx 消息
  - CHGDTAARA-增加了对*ALL 关键字的支持，以更改整个数据区域
- 屏幕
  - 通过提高滚动和光标的精度来改善 tables/subfile 显示 position/priority 效果
  - 改进了非数字CHECK(RZ)和非符号字段的CHECK(RB)功能
  - 改进了对关键字帮助屏幕功能的支持 HLPARA
- 角色扮演游戏
  - 改进了对内置的支持 %SubDt
  - 改进了对使用外部描述的本地数据结构的过程的支持
  - 增加了对可选错误代码参数QMHSNDPMQMHRMVPM 、和的支持 QMHRCVPM
  - 增强了对%SUBST内置方法的支持，可以更好地处理双字节字段。
  - 增加了对内置 %TLOOKUPLE 及其变体（%TLOOKUPGE、%TLOOKUPGE、%TLOOKUPLE、%TLOOKUPLT）的支持
- 数据区域
  - 改进了 factor1 为空时对 OUT 操作的支持
  - 改进了对同一数据区域的并发读取
  - 添加了配置变量blu4iv.dtaara.library.disable以禁用数据区域的库
  - 扩展了对通过数据区操作利用命名库的支持，使用户能够随心所欲地构造数据区域位置。
- DataQueue
  - 改进了 RabbitMQ 频道的使用情况
  - 改进了 RabbitMQ Consumer，使其只尝试取消消费者一次
  - 仅在等待时间为 0 时才尝试 basicGet，从而改进了从 RabbitMQ 检索数据队列的功能
- Misc (杂项)
  - 用户空间-改进了多个作业尝试同时检索同一个用户空间时的行为

- 改进了对在承诺控制下删除未提交的记录的支持
- 实体-改进了对连续省略的支持，因为 OMIT 带有隐含含义 AND
- 在实体映射器设置器中增加了对驼峰大小写的支持，以处理通过其他重构定义的自定义名称
- 改进了从 AS4 00 个环境事务到整个应用程序的用户信息的传播。
- 提高了 Quartz 在中断情况下结束由 Quartz 安排的作业时的准确性。
- 改进了对承诺控制的支持，使其成为程序范围

AWS Blu Age 变身引擎 4.5.0

zOS

改进

- JCL-改进了基于 LISTCAT 解析的 KSDS 数据集的 groovy 生成
- COBOL
  - 改进了COPY-REPLACING语句解析，以便在该子字段名称存在歧义时处理限定子字段的替换
  - 改进了对SPECIAL-NAMES语句中SYSOUT定义的支持
  - 改进了语句中对比喻 ZEROES 的支持 ADD n TO ZERO
  - 改进了对REPLACE语句的支持，通过扁平化多行键和文本块来处理多行问题
  - 改进了对带子句的算术运算ADD/SUBTRACT/MULTIPLY/DIVIDE的GIVING支持
  - 已启动对报告部分及其相关操作（启动、终止、生成报告）的解析支持
- 杂项-改善天气报告的生成和稳健性

AS400

改进

- DDS
  - 改进了对 DATE 类型隐式长度的支持
  - 改进了关键字 EDIT stop-zero-suppression WORD 上对字符的支持
  - 改进了对列名 DESC 的支持，因为它是 DB 中的保留字
- 角色扮演游戏
  - 改进了对内置 %TIME 的支持

  - 改进了 EVALR 语句的生成，可通过更好的右向调整来处理从字符串值到长度较短的变量的赋值
  - 增强了围绕选项设置的 SQL 解析
  - 改进了 NOMAIN RPGLE 程序中对 PSDS 初始化的支持
  - 改进了对关键字 LIKE 的支持，可将 DDS 数值字段定义为 Packed，无论其外部描述如何
  - 将 “$” 替换为 “DL”，改进了文件名清理
  - 改进了对内置 %SUBST 的支持，以处理双字节值
- COBOL400
  - 屏幕-改进了对围绕 I/O 操作的 DSPF 记录的支持
- CL
  - 改进了保留变量名的重命名
  - 改进了对处理多种格式文件的 Select/Omits 条件的支持
- Misc (杂项)
  - 减少了文件操作周围的重复实体（EOF、FOUND、EQUAL）
  - 改进了 AS/400 上的标准打印机 QPRINT 的 JRXML 文件生成。使用该文件时，创建的 JSON 文件将不包含对程序或文件的任何引用。只生成了一个 JRXML 文件 (qprint-qprint.jrxml)
  - 改进了显示程序队列消息的组件的附加消息信息的显示

版本说明 4.4.0

发布日期：2024 年 11 月 13 日

此版本的 AWS Blu Age Runtime 和 Transformation Engines 侧重于升级关键依赖项和支持的技术，同时提高多种功能的性能。此版本中的一些关键特征和更改包括：

- 依赖项更新：控制台应用程序（BAC 和 JAC）和现代化应用程序现已在 Bootstrap 5 上运行。 AWS Blu Age Runtime 现在由 Spring Boot 3.3.5 框架提供支持。
- 性能：在将 Spring State Machine 库从版本 2.5.1 升级到 4.0.0 之后，新的实现克服了性能下降的问题，从而提高了状态机的执行性能（速度提高了 10 倍）。此升级不是可选的，因为 2.5.1 版本已不再维护，并且包含 “临界” 和 “高 CVEs”。它包括平台上的运行时状态机实现到一个新的库，它具有轻量级高效的状态机实现，没有 CVE，并且具有更好的整体性能。
- 简化数据库访问：完成了对用于访问数据库的组件的重大修改，包括 JPA 实体 DAOs、DDS DataSimplifier 实体和 Mappers。之所以进行这种重新设计，是因为需要为00个项目中常见的 OVRDBF（覆盖数据库文件）功能提供更好的支持。 AS4它允许通过生成的代码的简化架构来处理更多案例。

我们使用以下堆栈测试了这个版本的 AWS Blu Age Runtime。其他组件版本可能也兼容。

组件	测试的版本
Java	Java 17
表示层	Node JS 18.18
	Npm 9.8
	Angular
服务层	Spring Boot 3.5
	Spring Core 6.1.14
	Spring statemachine 4.0.0
持久层	PostgreSQL 引擎 14
	甲骨文 21c
应用程序服务器	Apache Tomcat 10.1.17

有关此版本所包含更改的更多信息，请参阅以下部分。

运行时版本 4.4.0

zOS

新特征

- COBOL-增加了对 JSON 生成语句的支持
- COBOL-增加了对控制块的支持
- MF-增加了对 FCDREG 编译器指令的支持
- <noloc>Blusam</noloc>-添加了功能性的 VSAM 文件集，其实现基于数据库架构-仅支持 PostgresSQL
- <noloc>Blusam</noloc>-增加了对处理 B <noloc>lusam</noloc> 缓存数据项（Redis 缓存引擎）的 TTL（生存时间）的支持

- JCL-IDCAMS-添加了新属性`idcams.encoding.forced`以强制使用字符集解码 SYSIN 卡
- JICS-将`jics.db.dataScriptLocation`属性从扩展`application-main.yml`为接受文件和文件夹路径列表。清单的顺序很重要。首先执行第一个 SQL 文件，依此类推。执行文件夹时，其包含的 SQL 脚本将按不确定的顺序执行。
- 增加了对 CEE3 ABD 实用程序的支持

改进

- <noloc>Blusam</noloc>-为使用 PostgreSQL 引擎的客户缩短了从传统大型数据集到 <noloc>Blusam</noloc> 的加载时间和内存占用（我们观察到大型数据集的加载速度提高了 8 倍）
- <noloc>Blusam</noloc>-改进了 toS exportDataSet 3 API，支持凭证
- <noloc>Blusam</noloc>-改进了用于创建数据集的 LISTCAT 上传文件
- <noloc>Blusam</noloc>-使用显式密钥改进了对动态读取的支持
- <noloc>Blusam</noloc>-改进了写入机制逻辑
- JCL-增强了 JES 支持，可改善并行执行中的文件锁定
- JCL-增加了对语句的支持 `INCLUDE MEMBER`
- JCL-DNSUTILB-改进了对重复密钥的支持，以处理主键包含空格的特殊情况
- JCL-DSNUTILB-经过改进 LoadTask 以优化加载图形数据时的性能
- JCL-INFUTILB-增加了对`fetchsize`未定义时的支持 `chunksize`
- JCL-INFUTILB-改进了对返回空结果集的查询的支持
- JCL-INFUTILB-提高了在 CHUNK 中处理数据时的稳健性
- JCL-INFUTILB-改进了对带有可空字段的卸载的支持
- JCL-INFUTILB-改进了对数字类型的支持
- JCL-INFUTILB-改进了可为空字段的卸载
- JCL-排序-改进了对 OUTREC 语法的支持
- JCL-排序-改进了语句的解析 DATE1
- JCL-SORT-使用 RDW 改进了对子句 INREC PARSE 的支持
- JCL-SORT-使用编辑掩码改进了字段格式
- JCL-SORT-改进了 OUTREC 中对 “SubString” 的支持
- JCL-SORT-改进了对兼容 MF 的卡片的支持

- JCL-UNLOAD-Postgresql 改进了对字段大小的支持
- JCL-IDCAMS-通过引入批量模式提高了文件加载 VSAM 数据集的性能
- PL/1-增强对 NumericEditedType 格式的支持以防止比例差异
- IMS-改进了对 IMS 数据库 _right 列的支持 NodeSorter
- CICS-改进了`RECEIVE MAP`带有 and no `SET` 的命令 `INTO`
- BMS-改进了对字段初始值的支持
- SQL-改进了 DateTimeFormat 模式解`ddMMMyy`析
- COBOL-改进了在获取 NumericEditedType 值时不考虑小数点时对值的支持
- 改进了对读取行序文件中可变长度字段的支持
- 改进了对 GDG 文件从数据集目录继承记录大小的支持
- 允许自定义前进行，改进了对打印报告的支持
- 改进了变量块 (VB) 文件记录数据的初始化

GS21

新特征

- 屏幕-增加了对 PSAM 文件的支持
- 屏幕-增加了对屏幕的支持 ATTR2
- 增加了对 AIM（高级信息管理器）生态系统的支持。
- 在 AIM 中添加了 PED 支持

改进

- 改进了要处理的 BitUtils 签名 RangeReference
- 改进了对 DummyFileConfiguration 添加 recordSize/rdw/bdw/blkszlim 属性的支持
- 改进了对 VPOINT 语句的支持，以处理未找到记录的情况
- 增加了访问记录字节数组时的稳健性
- 改进了 JEF 字符集字符映射
- 改进了对在 JDBC 映射中处理数组和条件的支持
- 改进了对不同 NDB 语句中的 SQL 请求的支持，在 SQL 查询的每个部分使用常量更好地处理 SQL 语法的变化。

- 改进了对 GS21 PackedType 最后一个 nibble 为 C、D 或 F 的支持，用于数值验证
- 屏幕-改进了对 ACSAPI 以及 SPA 和 ENT DefaultPsamController ER 的支持
- 屏幕-增强了对 ACSAPI 和 NDB 动词的支持

AS400

新特征

- 增加了对多记录格式数据库文件的支持
- 重新设计了 AS4 00 数据库访问框架
  - 增强了文件覆盖方面的功能
  - 删除了过时的组件并降低了复杂性
  - 简化了从旧版程序生成的代码
  - 将 DAOCycle Manager 组件集成到 Blu4iV 插件中，使我们能够利用自定义运行时中 AS4特定于 00 的功能。
- JOB-改进了对任务管理 (Quartz) 的支持，增加了中断作业 job/group 的功能。添加了 REST API 端点，用于中断具有指定执行 ID 的作业（由于它是主键，因此每个作业都是唯一的）。成功中断后，运行时会将作业的状态更新为 “已中断”。
- 增加了对实用程序 CEERAN0 的支持
- 增加了对被动模式的支持。添加了 YAML `configuration gapwalk-application.cl:ftpservice:passive` 以启用被动模式
- 增加了创建 QTEMP 会话和延迟 QTEMP 清理的功能
- 增加了对编译功能 BNDDIR 的支持，用于定义程序之间的显式依赖关系
- 增加了对激活组机制的支持

改进

- CL-改进了程序消息队列上的 RMVMSG 命令以处理 *PREV 关键字
- CL-改进了 OPNQRYF 中对覆盖的支持
- CL-增加了对 RTVMSG 命令的 MSGLEN 和 SECLVLLEN 参数的支持
- CL-改进了对 CRTDUPOBJ 的支持，以便在未通过 NEWOBJ 时管理大小写，并增加了对通用表名的支持
- CL-改进了对 FTP 的支持，以处理参数 GET、RMTSYS 和 BINARY

- CL-提高了 CLRPFM 查询性能，并添加了使用 TRUNCATE 而不是 DELETE 的选项
- CL-改进了 SBMJOB，可以正确处理用户参数，以便在提交作业时将其用作用户
- CL-改进了 DLTOVR 命令支持，可以处理 *ALL 的情况
- 数据区-DataArea 通过添加异常处理日志来改进对 Blu4 的支持
- 数据区-改进了对 Blu4 的支持DataArea ，可以为每个线程获取一个新 DataAreaDao 实例
- 数据区-改进了数据区域锁定，避免了记录级别的锁定，而是使用了新实现的锁定机制
- Data Area-当未获取锁并提供错误指示器时，数据区写入操作现在会继续执行
- 报告-改进了对打印报告的报表输出 path/naming 惯例的支持。允许客户自定义报告输出路径和名称。客户可以指定自己的路径和命名惯例，而不会影响任何其他项目。
- JOB-改进了对作业管理 (Quartz) 的支持，可在任务异常终止时更新作业状态。例如：Tomcat 的 “关闭” 或 “异常关闭”
- 屏幕-改进了对带有减号编辑单词的字段上数值的处理
- 屏幕-仅使用屏幕改进了渲染弹出窗口 titleColorTop
- 屏幕-改进了在找不到常规帮助项目时检索帮助信息以处理案例的支持
- 屏幕-改进了在子文件消息行上按下 F1 时显示 “其他信息” 屏幕
- 屏幕-改进 SFLMSG 消息行页脚的显示
- 屏幕-改进了前端，可在新记录重叠时将其全部删除
- 队列-改进了 RabbitMQ 消息检索以减少资源消耗
- 队列-改进了 RabbitMQ 数据队列的实现，一次只能检索一条消息。
- SQL-改进了 SQLExecutor生成器对动态创建和删除表查询的 SQLCODE 的处理
- SQL-改进了查询时对 OVRDBF 的支持
- SQL-改进了 SQLExecutor生成器，以便将 OVRDBF 覆盖应用于准备好的语句
- RPG-改进了对程序描述的磁盘文件的输入和输出规范的支持
- RPG-改进了对使用 MR（匹配记录）指示器读取主文件和次要文件读取的支持。带有匹配字段的 DAO 周期的检索顺序已得到改进。
- RPG-改进了对主文件和次要文件的支持。改进了更新主文件和输出辅助文件 update/write 代码生成。
- RPG-增加了对自由格式的 RETURN 语句的支持
- RPG-改进了数字十进制赋值的转换和运行时处理，

- RPG-改进了二进制变量的生成
- RPG-改进了对 EDITC 的支持
- RPG-改进了对本地数据区域的处理
- 改进了对多种设备类型（磁盘、WORKSATION、PRINTER）共享的 DDS 字段的支持
- 改进了覆盖处理，因此开启的覆盖 PFs 将不再影响 LFs
- 改进了 Blu4，ivWebController 不将用户名和用户 ID 重置为默认值
- 改进了读取方向改变时读取记录期间的索引调整
- 改进了 update/delete 操作后读取记录时的光标位置
- 改进了当读取方向发生变化时在多实体 DAO 上读取的支持
- 改进了对用户空间的支持，以避免所有线程重复使用实例，而不是每个线程都有自己的实例
- 改进了读取记录时对多线程并发访问的支持
- SharedContext 通过 YML 配置改进 username/userid 了输入的存储
- 使用更新的值改进了锁定记录的发布
- 为下一句话语句添加了对 OPM 编译器特定行为的支持

横向功能

新特征

- 添加了新的 metadata.ini 属性，`legacy.compilerto`用于指定要转换的工件的旧版编译器。某些 COBOL 语句（例如 NEXT SENTENCE）的支持会因您设置的值而异。
  - “ZOS” 表示 z/OS 传统系统。
  - “ILE” 或 “OPM” 表示 AS4 00 系统。当 = “as400” 时，默认 `legacy.system` = “ILE”

改进

- 前端-重新设计了屏幕字段组件，以扩大支持的字段类型的范围。此增强功能使运行时能够适应 AS4 00 中涉及的更多种类的用户输入和数据要求。
- 改进了开`isValid()`启单独符号字节的方法 ZonedType
- 改进了对涉及 CRLF `StringConcatenationBuilder::withPointer` 的串联的支持
- 改进了对特定双字节编码的支持，使其成为线程安全
- 通过集成新框架提高状态机性能

- 改进了分配优化算法，可防止意外重写

AWS Blu Age 变身引擎 4.4.0

zOS

改进

- LISTCAT-改进了解析器以防止重复条目
- LISTCAT-改进了 JCL/Groovy 中文件系统对 ESDS 的支持
- CICS-改进了对 CICS 语句长度的支持

AS400

改进

- DDS 记录生成增强功能
  - 改进了对 DDS 记录的支持，以生成与 DDS 记录结构相对应的实体
  - 为共享字段和映射功能提供了支持，使其与旧版更匹配
  - 改进了对外部描述文件和程序描述文件的处理
- RPG-改进了只有自由形态的模块的 RPG 检测
- RPG-改进了对 COPY 语句的支持，以忽略关键字`*LIBL/`作为前缀来查找应用程序抄本
- RPG-PF-使用来自 pfile 的物理记录改进了对输入规范的支持
- RPG-增加了对 On-Exit 语句的支持
- RPG-改进了对关键字的 LikeRec 支持
- RPG-改进了重命名 DSPF 字段的映射
- CL-改进了字段名称解析
- COBOL-改进了对从十六进制转换为字符的支持
- 改进了对十进制类型生成的支持
- 改进了对不支持的旧代码的 FIXME 消息的支持（显示整行旧版）
- 提高了 AWS 转换引擎的性能（AS40 解析步骤）
- 改进了对关键字的支持 LikeRec ，使其与文件规范保持一致
- 改进了对内置函数 %Diff 的支持

- 在 DSPF 标签上添加了对特殊字符货币符号的支持

4.3.0 发布说明

发布日期：2024 年 9 月 16 日

此版本的 AWS Blu Age 运行时和现代化工具侧重于扩展功能和覆盖范围，以实现大型机功能的现代化。此版本中的一些关键特征和更改包括：

- CICS：通过支持带有映射引用的 SEND MAP 命令，进一步支持交换来自终端的数据，并使用传入的数据运行事务。
- JCL：新功能允许从先前失败的步骤重新启动最近一次执行的批处理作业，或者通过绕过先前执行的 JCL/PROC 步骤来触发延迟重启。这样可以更好地控制使用持久的步骤级检查点进行的批处理。
- AS400：额外的库支持、CPYF、OVRDBF、SBMJOB 和 OPNQRYF 等常用命令的增强性能和稳健性等等。

我们使用以下堆栈测试了这个版本的 AWS Blu Age Runtime。其他组件版本可能也兼容。

组件	测试的版本
Java	Java 17
表示层	Node JS 18.18
	Npm 9.8
	Angular
服务层	春靴 3.2.5
	Spring Core 6.1.5
	Spring statemachine 4.0.0
持久层	PostgreSQL 引擎 14
	甲骨文 21c
应用程序服务器	Apache Tomcat 10.1.17

有关此版本所包含更改的更多信息，请参阅以下部分。

运行时版本 4.3.0

zOS

新特征

- CICS：在 SEND MAP 命令中添加了对映射引用的支持
- CICS：添加了对 RECEIVE 命令的支持以及对使用来自 `JicsTransactionRunner` 屏幕的数据运行事务的支持
- 添加了对 JMS 消息的 IIH 标头的支持
- COBOL：在 REPLACING 语句的伪文本中添加了对多个嵌入空格的支持
- COBOL：添加了对 JSON PARSE 语句的支持
- <noloc>Blusam</noloc>-增加了对 KMS 的支持，使其具有 “导出数据集” 功能
- BAC：增加了用于定义记录大小的 `application-main.yaml` 配置，以筛选与该记录大小匹配的已加载掩码
- JCL - INFUTILB：添加了对于在 BMC 控制语句中使用关键字 INTO 的支持
- GS21 -为 JEF 编码添加了 SOSI 处理
- GS21 -JCL- KDJBR14 作为别名添加 IEFBR14
- GS21 -JCL-添加了 KQCAMS 作为 IDCAMS 的别名
- MF：增加了对 COBOL MF 兼容文件的支持，具体取决于字段支持
- MF：添加了对 COBOL MF 兼容文件的 SORT 机制的支持
- MF：添加了对 COBOL MF 兼容打开非可选缺失文件的支持

改进

- JCL - DSNUTILB：使用 ZONED DECIMAL 类型改进了 LOAD 操作
- JCL - DSNUTILB：添加了对重复密钥的支持
- JCL - DSNUTILB：在 LOAD 命令上添加了对回滚机制的支持
- JCL - INFUTILB：使用新属性 FETCHSIZE 和 CHUNKSIZE 改进了 UNLOAD
- JCL- IKJEFT1 A-通过添加当前字符集增强了 SYSTSIN 文件读取能力
- JCL-DFSORT-增加了对选项和的支持 DATE4 DATE5

- JCL - DFSORT：添加了对可变块类型作为输入和固定块类型作为输出的支持
- JCL - DFSORT：添加了对 ALTSEQ 的支持
- JCL：使用作业 Web 标识符增强了检查点元数据
- JCL：改进了 REDIS 的批处理重启检查点清除功能
- IMS：实现了 PURGE 命令的 EXPRESS 函数
- IMS：添加了对 PCB 语句的 PCBNAME 和 LIST 选项的支持
- COBOL：添加了对不带目标的 GO TO 语句的支持
- CICS-改进了 READQ TS RecordAdaptable 中对 INTO 语句的支持
- CICS：改进了对 INQUIRE TRANSACTION 命令的支持
- CICS：在 READNEXT 命令中改进了对 setBytes 的支持
- CICS：改进了对不带 CHANNEL 选项的 START 命令的支持
- CICS-增加了对查询参考类型的支持 TSQueue
- CICS：改进了映射和映射集为“Reference”类型时对 RECEIVE MAP 命令的支持
- CICS：改进了对 RECEIVE MAP 命令的 FROM 和 LENGTH 选项的支持
- CICS-增加了对属性的 RecordAdaptable 支持
- CICS：改进了对 RECEIVE 命令处理溢出的支持
- CICS：在 CICS 语句中添加了对切片规则的支持
- CICS：改进了对链接结构 DFHCOMMAREA 和 DFHEIBLK 的支持。转换引擎支持更多隐式定义
- CICS：添加了对 INQUIRE CONNECTION 命令的 START、NEXT 和 END 选项的支持
- CICS：为 RECEIVE 命令的 LENGTH 选项添加了对“int”和“reference”类型的支持
- CICS：增强了对解析 INQUIRE NETNAME 命令的支持
- CICS-增加了对群组名称的支持 JicsQueueBuilder
- <noloc>Blusam</noloc>-增加了对以通用密钥开头的索引文件的支持
- <noloc>Blusam-改进的 Blusam</noloc> <noloc>装填器</noloc>
- BAC：改进了在多实例环境中使用 Redis 集中缓存值（包括实际数据和锁）时对数据同步的支持
- BAC：改进了 UI（样式、徽标、复选框）
- BAC 和 JAC：增加了“application-main.yaml”配置，通过指定 ARN 从 AWS Secrets Manager 检索密钥中的默认超级管理员用户的用户名和密码
- BAC 和 JAC：将依赖项升级到 Bootstrap 5

- 改进了 JCL 检查点和 JICS TSQueues Redis 模板配置
- 改进了对指针大小的支持，具体取决于 AMode
- 增加了对零比较的支持 NumericEditedType
- 在记录日志之前强制执行 Slf4j MDC 属性
- 改进了对文件读取的支持以处理多个空行
- MF：改进了对初始化 COBOL MF 编译器指令 initPtr 的指针变量的支持
- Redis-通过基于Redisson的实现改进了并发方面的功能 GwFileLock

AS400

新特征

- CL：添加了对 CHGPF 命令的支持
- RPG：添加了对 %HOURS、%MINUTES 和 %SECONDS 函数的支持
- COBOL：添加了对采用 Blu4IV DAO 架构的 SORT 文件的支持

改进

- CL-改进 PgmClose 为可注册为程序并接受各种对象作为 OPNID 参数
- CL：重构了 RTVMBRD 以处理多个库和成员
- CL：在 MOVOBJ 命令中添加了对 TOLIB 参数的支持
- CL：改进了 CPYFRMSTMF 命令上对分区的支持
- CL：添加了对 SNDMSG 参数 TOUSR 的支持
- CL：改进了对 OVRDBF 命令的支持
- CL：改进了 OVRDBF 命令的性能 - 更新 srcfile 和 member 的默认值
- CL：使用 CPYF 命令改进了文件复制
- CL：重新设计了 CPYF 命令，使其更强大，从而可以更好地处理 QTEMP、CRTFILE、FROMRCD 和 TORCD、MBROPT 和 FMTOPT（MAP 和 DROP）
- CL：改进了在 FROMFILE 和 TOFILE 的列不匹配的情况下对 CPYF 命令的支持
- CL：改进了 CPYF NOCHK 在指定 REPLACE 时，对于具有不同名称的列的处理
- CL：为逻辑文件上的 CRTDUPOBJ 命令添加了空实现

- CL：使用 CHGDTAARA 命令解决了子字符串的索引问题
- CL：改进了对 SBMJOB 命令的支持
- CL-Made OverrideManager 和 OpnqryfHelper mappings 不区分大小写
- 屏幕：改进了未指定游标时第一个可编辑字段的初始焦点
- 屏幕：改进了关闭后和使用帮助菜单时的焦点位置
- 屏幕-改进了 up/down 在表格组件中按下页面后的光标焦点
- 屏幕：改进了对多字段错误消息和焦点的支持
- 屏幕：改进了子文件字段的行号计算
- 屏幕：改进了对使用 SFLINZ 初始化的子文件的支持
- 屏幕：改进了对仅限数字输入的支持
- 屏幕：使用 3 个参数改进了 DSPF 中对 WINDOW 关键字的处理
- 屏幕：改进了其中的记录包含多于一行的表的页脚位置
- 屏幕：改进了页面导航，可使旋转消息贴附在“上一/下一页”上
- 改进了编辑代码 3 的 EDITC 功能
- 改进了 Blu4iv 数据区域锁定机制，使其在无锁可解时什么都不做，而不是抛出异常
- 增加了对返回中受影响的行数的支持 StraightQueryBuilder
- 改进了 QTEMP 日志机制
- DAOManager reads/writes/deletes针对被其他 file + 库覆盖的文件的用例进行了改进

横向功能

新特征

- 添加了一种通过配置管理与 SSL/TLS 相关的系统属性的集中方式，允许使用 AWS Secrets Manager
- 增强了 IBMMQ 资源的配置 AWS Secrets Manager
- JCL-通过 YML 属性为运行时解析的 groovy 文件添加了临时位置配置， tempFilesDirectory 并增加了指定是否通过 YML 属性在应用程序启动时清除临时文件文件夹内容的功能 cleanTempFiles DirectoryAtStartup
- 为所有 Redis 凭证添加了 AWS 密钥

改进

- 改进了从字母数字类型到编辑后的数字类型的转换
- 改进 DataUtils:: isNumeric 检查 PackedType
- 增强了为日志文件添加时间戳的功能
- 已单独处理登录 ZonedType。 decodeAsString
- COBOL：改进了对语句 INITIALIZE 的支持
- 改进了对的支持 DataUtils。 compareAlphInt 处理 AS4 00 和 ZOS 的前导和尾随空格
- SQL：改进了隐式只读游标运行时验证
- SQL：改进了元数据缓存机制
- 从 Gapwalk 应用程序中删除 Jics/ <noloc>Blusam</noloc> 数据库连接 `application-main.yml`

现代化工具版本 4.3.0

zOS

新特征

- GS21 -添加对 COBOL GS21 常量截面的支持
- GS21 -为可用字符集添加了 JEF 编码

改进

- CICS：添加了对解析 DOCUMENT CREATE 创建命令的支持
- CICS：添加了对解析 CICS WEB EXTRACT 命令的支持
- CICS：添加了对解析 WEB WRITE 命令的支持
- CICS-增加了对 DB2 CONN SIGNIN 和 PLAN 的转换支持
- CICS：通过忽略 TERMINAL 选项增强了对解析 SEND MAP 命令的支持
- CICS：通过忽略 ENDACTIVITY 选项增强了对解析 RETURN 命令的支持
- MFS：改进了对生成具有特定扩展名的 MFS 文件的支持
- COBOL：改进了对 REPLACE 语句的支持
- COBOL：处理了动态路径和 MF 编译器指令
- COBOL：改进了对 CALL 语句中的 OMITTED 值的支持

- COBOL：改进了多维字段访问以支持有符号值
- COBOL：添加了对为 FILE STATUS 语句的 OF 子句的支持
- COBOL-改进了语句的解析 RESULT-SET-LOCATOR
- JCL - IDCAMS：添加了对 RECORDS 缩写的支持

AS400

新特征

- CL：在 CL 转换中添加了对基于指针的变量和已定义变量的支持
- CL：在 DCLF 中添加了对特殊字符的支持
- 添加了对检索调用堆栈（QWVRCSTK）API 的支持

改进

- RPG：使用 `likeds` 关键字改进了过程参数的转换
- RPG：增加了对关键字 EXTNAME 的审查支持
- RPG：改进了对字面值 *ALL 的支持
- RPG：改进了对输出规范和程序描述文件的支持
- DDS：改进了对引用 PF（该 PF 引用 Dictionary PF）的 LF 中的 DDS 字段的解析
- 屏幕：使用 CLEAR 语句清除 DSPF 中的记录时清除了指示器
- CL-改进 transformation/generation 了带有元素列表的 CL 参数

横向功能

改进

- SQL：改进了包含带波浪号字符的 N 的 SQL 查询的生成
- COBOL：改进了组字段对 LENGTH OF 语句的支持
- COBOL：改进了对使用副本的 REDEFINED 字段的支持

4.2.0 发布说明

发布日期：2024 年 7 月 10 日

此版本的 AWS Blu Age 运行时和现代化工具侧重于性能和安全性。此版本中的一些关键特征和更改包括：

- 改进了转换性能，特别是对于拥有超过 3000 万行代码的大型项目。我们实施了一系列改进，结果显示时间缩短了 150% 以上，运行只需要几分钟而不是几小时才能完成。我们实现的关键改进是配置了超时机制，以限制分配给分析的最大时间，从而跳过检测到问题的文件。我们会标记跳过的文件，以便您可以稍后在必要时对其进行调查。
- 我们为 AS4 00 个项目增加了对分布式锁管理系统的支持。在应用程序的多个实例以同一个数据库为目标的高可用性环境（多节点）中，在这些实例的整个生命周期中保持数据一致性是一项重大挑战。为了有效应对这一挑战，我们添加了 Redis 作为共享和外部缓存服务器，以便在批处理模式下运行时在所有实例之间进行协调。
- 我们为表组件添加了新的动态分页特征。此特征的目标是缩短响应时间并减少包含大量行的表的内存使用量。此特征支持表组件仅加载部分数据，并在您浏览页面的过程中按需获取更多记录。为了进一步改善体验，该平台还支持预取数据。这项新的动态分页特征为具有大型数据集的应用程序提供了更高效、响应更快的用户体验。
- 为了应对一个经常出现的重要挑战，我们添加了对嵌套 COBOL 程序的支持。以前，实现嵌套 COBOL 程序的现代化的解决方法包括手动将程序分成不同的文件，通过链接部分将它们链接起来，然后让它们使用必要的参数相互调用。这个过程不仅耗时，而且容易出错。现在，您无需手动分割即可实现嵌套 COBOL 程序的现代化。

我们使用以下堆栈测试了这个版本的 AWS Blu Age Runtime。其他组件版本可能也兼容。

组件	测试的版本
Java	Java 17
表示层	Node JS 18.18
	Npm 9.8
	Angular
服务层	Spring Boot 3.2.4
	Spring Core 6.1.5
	Spring statemachine 4.0.0

持久层	PostgreSQL 引擎 14
	甲骨文 21c
应用程序服务器	Apache Tomcat 10.1.17

有关此版本所包含更改的更多信息，请参阅以下部分。

运行时版本 4.2.0

zOS

新特征

- DB2 -添加了对在 SQL 查询中调用不带架构限定符的存储过程的支持
- COBOL：添加了对 HEX-OF 函数的支持
- COBOL：添加了对嵌套程序的支持
- COBOL-增加了对函数 TEST-DATE-YYYYMMDD和的支持 TEST-DAY-YYYYDDD
- CICS：在 SET TERMINAL 命令中添加了对选项 UCTRANST 的支持
- CICS-增加了对 INQUIRE DB2 CONN 命令的支持
- <noloc>Blusam</noloc>-增加了对动态访问的 VSAM 上删除密钥的支持
- IMS：添加了对 TERM 命令的支持。
- BAC：添加了对所有 BAC REST 端点的授权检查
- BAC：添加了通过 `application-main.yaml` 定义记录大小的配置，以筛选与该记录大小匹配的已加载掩码
- BAC 和 JAC：添加了通过指定 ARN `application-main.yaml` 来检索密钥中默认超级管理员用户的用户名和密码`command`的配置

改进

- JCL-SORT-增强了对 OMIT 子句的支持，用于处理带有 Shiftin 和字符的条件 ShiftOut
- JCL - SORT：改进了对 BDW 字段的支持
- JCL - SORT：改进了对使用 BDW 字段的多个 GDG 串联的支持
- JCL - DFSORT：添加了对 INREC PARSE STARTAFT / STARTAT 子句的支持

- JCL - IEBGENER：增强了对输出文件的 recordSize 处理
- JCL - INFUTILB：禁用了基于 YML- FIX GRAPHIC CASE 的 NULL INDICATOR
- JCL-改进了 FormatterParser 对处理 OUTREC 字段中常量的支持
- JCL：增强了 DSNUTILB 实用程序中的图形类型数据加载
- JCL - SORT：增强了对分区十进制格式的支持
- JCL-SORT-增强了对 OMIT 子句的支持，用于处理带有 Shiftin 和字符的条件 ShiftOut
- MQ：改进了 MQ 连接的处理，以适应多种业务工作流程
- CICS：增强了对 EXEC CICS READ SET（ptr-ref）语句的指针引用的支持
- COBOL：改进了对 ADDRESS OF 链接区段记录的支持
- COBOL-增加了对 EXP 和 EXP1 0 函数的支持
- COBOL：改进了对使用副本的 REPLACE 语句的支持
- COBOL：改进了多维字段访问以支持有符号值
- MF COBOL：添加了对可变格式顺序文件的支持
- IMS：改进了对 IMS YML 文件配置的读取，以便能够使用环境变量
- IMS：处理了指定段号的其他方法
- IMS：提高了从编程启动的事务调用 IMS 程序时的稳健性
- IMS：改进了 SSA 构建的搜索标准，以便在未提供隐含的分段长度时将 WHERE 子句的当前长度纳入考虑
- IMS：改进了对 IMS YML 文件配置的读取，以支持使用环境变量
- 改进了中对 VALUE 子句的支持 NumericEditedType
- 改进了对字符串串联的支持，以处理要串联的第一个字符串为空、空白或空格的情况

AS400

新特征

- 在 Table 组件中添加了对分页的支持；当加载包含大量行的 Table 组件时，项目可以使用此特征来缩减响应时间和大小
- 在 AS4 00 应用程序上添加了对 SQL 查询的库支持；由于库在现代应用程序中会转换为分区，因此我们调整了运行时以相应地重写查询
- RPG：添加了对用于 SQL 查询的 QTEMP 库的支持
- RPG：在 CONVERT 函数中添加了用于处理空输入值的编码功能

- RPG：添加了对 %HOURS、%MINUTES 和 %SECONDS 函数的支持
- CL：添加了 CHGPFM 命令
- CL：在 CRTDUPOBJ 命令中添加了对 *FROMLIB 关键字的支持
- CL：添加了对表名称超过 9 个字符的表和分区创建的支持
- CL：为 DLTF 命令添加了对删除子文件夹中的平面文件的支持

改进

- 屏幕- ErrorMessage 改进了绑定特定字段并添加到 ArrayMessageLine
- 屏幕：改进了错误消息游标
- 屏幕-改进 ArrayMessageLine 为不包含在 Tab 顺序中
- 屏幕-改进了 AS4 00 屏幕错误消息数组的显示
- SQL：改进了在关闭时提交事务的游标支持，以避免创建分区时出现死锁
- CL-增加了对该 PgmCall 命令的支持并改进了 QCMDEXC 不支持的模式
- CL：改进了对 CHKOBJ 命令的支持，以处理 OBJTYPE PGM
- CL：改进了对处理库和分区的 CPYF 和其他 CL 命令的多库支持
- CL：添加了对在 CALL PGM 命令中传递程序名称变量的支持
- CL：处理了 Object 类型的默认类型的情况
- CL：添加了对 CRTDUPOBJ 命令的多库支持
- CL：增强了多个命令上的数据库连接处理
- CL：改进了对 RMVLNK 的支持，以处理找不到文件或目录以及收到 CPF0000 监控消息的情况
- CL：改进了 CLRPFM，以便在删除记录时将库考虑在内
- CL-CPYF-改进了支持 QTEMP 库、 FmtOpt (*NoChk) 参数和控制字符的命令
- CL：修复了 RMVLNK 和 CPY 命令中对引号和缺失参数的处理
- RPG-增强的变量作用域；现在 DataArea 处于工作范围而不是链接作用域
- RPG：改进了 DAO 读取查询，无需事务即可运行以避免死锁
- 通过在数据库查找时向 MSGQ 添加修剪增强了 MQ 消息查询
- 删除了数据库连接支持上不必要的事务声明
- 改进了出现异常时 Quartz 作业状态的更新
- 添加了对处理未初始化指标数组的情况的支持

横向功能

新特征

- Redis：添加了针对所有 Redis 缓存的全局 Redis 配置
- 添加了会话跟踪功能，以便通过将数据持久保存到 Redis 来存储会话跟踪信息（会话 ID、关联的用户名、创建时间戳和节点 ID）
- 通过 YML 属性 `tempFilesDirectory` 为运行时解析的 groovy 文件添加了临时位置配置；还添加了通过 YML 属性 `cleanTempFilesDirectoryAtStartup` 指定在应用程序启动时是否清除临时文件文件夹内容的功能

改进

- 增强了对实用程序数据来源的连接池实现配置属性的支持
- 根据 ADVANCING 子句和 WRITE BEFORE 子句的使用，改进了对打印机模式和 ANSI 回车控制的支持
- 更新了用于现代化项目的前端应用程序的 Angular 版本
- 增强的密钥管理器 URL 语法构造 DB2
- 增强了 DataUtils. compareAlphInt 添加对尾随空格的支持的方法
- 改进了对 blob 类型输出的 SQL 支持
- 增加了通过端点触发任务的稳健性 post/script

现代化工具版本 4.2.0

zOS

新特征

- CICS：添加了对解析 WEB CICS 命令的支持
- CICS：添加了对转换 MONITOR 命令的支持
- CICS：添加了对解析 CICS 命令 SEND MRO 的支持
- COBOL：添加了对解析 NO REWIND 语句的支持
- COBOL：在 CICS 命令 SET TERMINAL 中添加了对数字类型的 UCTRANST 选项的支持
- COBOL-添加对中多文件子句的支持 I-O-SECTION
- CSD：添加了对转换多个 CSD 文件的支持

- CSD-增加了对从多个 CSD 文件生成 jicsFileAix .json 的支持
- IDCAMS：添加了对创建相对记录数据集（RRDS）的支持

改进

- 提高了计算 SQL 掩码时的性能
- COBOL：改进了 FILE-CONTROL 中对无用的 RESERVE 子句的解析
- COBOL：改进了对 SECTION 和 CLASS 的解析
- COBOL：改进了 DFHRESP 处理
- COBOL：通过执行增强了对 EXIT PARAGRAPH 的支持
- IMS：改进了对使用双圆括号指定的分段名称的支持
- IMS：丰富了调用 SCHD 和 TERM 时生成的状态码
- COBOL：改进了依赖字段的生成
- COBOL-改进了 TO_TIMESTAMP 内置函数 DB2 的转换

AS400

新特征

- 添加了对在 SQL 脚本中将字母数字字段转换为字符的支持
- COBOL400-增加了对程序描述的数据库文件的支持

改进

- DDS：改进了对别名的支持
- 增强了对不带初始值的 float 类型的支持
- COBOL 400：改进了有符号分区类型的大小计算

横向功能

改进

- 改进了有关 DDS 和 SQL 解析的错误 ID 报告
- 改进了条件分支上的代码生成

- 提高了天气报告生成的性能

4.1.0 发布说明

发布日期：2024 年 5 月 31 日

此版本的 AWS Blu Age 运行时和现代化工具侧重于性能和安全性。此版本中的一些关键特征和更改包括：

- 转换和性能：为了能够成功转换具有大型代码库（超过 5000 万行代码）的项目，我们优化了整个转换机制的性能和内存占用。
- BAC/JAC：安全性 AWS 是重中之重。使用 AWS Blu Age 进行现代化改造的应用程序必须符合安全标准。我们对 <noloc>Blusam 管理控制台 (B</noloc> AC) 和 JICS 管理控制台 (JAC) 进行了一些重大升级，以使其更加安全：
  - 已将应用程序更新到 Angular v17。
  - 除了对 AWS Cognito 的原生支持外，我们还添加了通用支持 OAuth，这将使客户能够更灵活地使用他们选择的身份提供商。
  - 使用适当的标头配置和扩展了安全特征。
- AS400-对数据库锁定机制的多节点支持。提供了插入共享和外部缓存服务器 (Redis) 以在多个实例（例如托管 AWS 大型机现代化）上运行批处理应用程序的可能性。

此版本的Blu Age运行时已使用以下堆栈进行了测试。其他版本可能也兼容。

组件	测试的版本
Java	Java 17
表示层	Node JS 18.18
	Npm 9.8
	Angular 16.1
服务层	春靴 3.2.5
	Spring Core 6.1.5
	Spring statemachine 4.0.0

持久层	PostgreSQL 引擎 14
	甲骨文 21c
应用程序服务器	Apache Tomcat 10.1.17

有关此版本所包含更改的更多信息，请参阅以下部分。

运行时版本 4.1.0

zOS

新特征

- 添加了动态 OAuth2 提供程序处理的配置。引入了 SECRET_ OAUTH2 _PROVIDER_NAME_KEY 来指定提供者。更新了密钥检索方法以处理多个提供程序。确保安全地从中检索机密 AWS Secrets Manager。
- 增加了对 DB2 SSL 属性的支持， AWS Secrets Manager 使您可以定义 SSL 证书（sslTrustStore位置）和密码（sslTrustStore密码）来解锁密钥库文件。
- 添加了对外部业务数据来源的支持。
- JCL：添加了对批处理重启的检查点机制的支持。
- JCL：添加了对 DCB 参数记录大小和 RDW 的支持。
- JCL：为生成的临时文件添加了动态文件夹名称配置。
- REDIS：在 JICS 的 Redis 配置中添加了池配置。
- REDIS：在 Catalog 和 JICS 的 Redis 配置中添加了数据库索引。
- BatchScript -添加了用于运行程序执行的步骤名称的传播。
- CICS：添加了对 ADDRESS SET 命令的支持。
- CICS：添加了对 PURGE MESSAGE 和 JUSTIFY 的支持。

改进

- JCL - INFUTILB：增强了对基于 YML 属性禁用空指示器的支持。
- JCL-INFUTILB-改进了对数据类型的支持。 CHAR/BPCHAR
- JCL - ICEGENER：添加了对将多行输入流复制到文件的支持。
- JCL - IEBGENER：改进了对处理从变量块文件到固定块文件转换的支持。

- JCL - DFSORT：改进了对操作日期上多位数参数的支持。
- JCL - DFSORT：添加了对 INCLUDE=ALL 子句的支持。
- JCL：改进了对 SORT 实用程序的支持，以处理输出中的 BDW 字段。
- JCL：改进了对 DD 串联的支持。
- JCL：改进了对输入流的支持。
- JCL - DSNUTILB：改进了对 NULLIF() 语句的支持。
- JCL - INFUTILB：添加了对使用 NOPAD 选项卸载数据的支持。
- JCL - INFUTILB：增强了对 INFUTILB 中当前日期的支持。
- JCL：添加了使用文件之前的文件存在和大小检查。
- JCL - GDG：改进了 GDG 子目录的处理。
- MQ：改进了 JMS 实现中的连接打开。
- MQ：改进了 XA 数据来源的 GET 消息的数据长度设置。
- MQ：分解了 CMQV 标准副本，用于防止编译错误和重构使用。
- <noloc>Blusam</noloc>-改进了对不存在的数据集的删除请求的支持。
- 改进了对 ALLOCATE 语句的支持。
- 提高了 TS-QUEUE 命名的稳健性。
- BatchScript -在重新执行任务时增强了对上一步返回代码的保存。
- 数据集：改进了文件存在且为临时文件时的文件存在检查。
- 数据集：改进了定位要删除的 GDG 文件时的并发性。
- 数据集：添加了对获取 GDG 数据集记录大小的支持。
- CICS：改进了对 INQUIRE TASK LIST 命令中的 SUSPENDED 选项的支持。
- CICS：使用 ADDRESS OF 语句改进了对 LOAD SET 的支持。
- CICS：改进了 CICS 查询时未处理的 CICS 参数 REMOTESYSTEM。
- CICS：增强了对 GETMAIN 命令的支持，以使用由 OF 关键字定义的指针来处理 SET 选项。
- JICS-通过添加事务状态检查提高了 jics XAPrepare () 方法的稳健性。
- JICS XA：添加了事务状态检查并增强了事务线程终止功能。
- BAC-增强了客户端和 refactored/centralized 所有 API 调用的基于角色的身份验证。
- BAC：实现了基于配置阻止对 BAC 和 JAC 的公共访问的特征
- BAC：升级了依赖项：Angular 17。
- BAC-改进了与. 的安全集成。 OAuth2

- BAC：增强了休眠生成的 DDL。
- BAC：改进了导出数据集机制。
- JAC：更新到 Angular 17，并报告来自 BAC 的所有具体工作（ROLE、sadmin conf、XSRF、logout）。
- COBOL：添加了对 CHAR 和 ORD-MIN 函数的支持。
- FileFactory 已增强，可在 MOD 处置中保持目录记录的大小。
- 启用了使用 MDC 对 JICS 事务进行日志记录的功能。
- 改进了为生成临时结果集的存储过程生成的 SQLCA > SQLSTATE。
- 改进了对与上次 Spring 升级相关的任务计划的支持。

AS400

新特征

- 添加了对使用 Redis 的数据库记录锁的多节点支持。
- 为 DDS 类型添加了对二进制字符的支持。
- CL：添加了对生成自定义报告文件的支持。
- RPG-增加了对 primary/secondary 文件重命名关键字的支持。

改进

- 改进了对使用 JOIN 子句处理 CTID 列的数据库支持。
- 改进了多个 DSPATR(PC) 的游标位置。
- 改进了发生读取异常时的日志记录。
- 改进了 Quartz 作业日志记录，将作业属性添加到 MDC 中。
- 改进了对 AS4 00 帮助屏幕的支持。
- CL：改进了对 RMVJOBSCDE 命令的支持，以接受带有尾随空格的条目号。
- CL：改进了对 RMVJOBSCDE 命令的支持，以使用通用作业名称删除作业计划。
- CL：改进了对 SAVOBJ 命令的支持，以按照表格键对记录进行排序。
- CL：改进了对 CPYF 命令的支持，以便为数据库查询建立新连接。
- CL：使用 SNDPGMMSG 改进了在队列消息中插入查询消息的功能。
- CL：改进了作业队列配置以指定默认作业队列。

- CL：改进了 CRTPF 命令以支持 QTEMP 库和 RCDLEN 参数。
- CL：改进了对 CHKOBJ 命令的支持 - 检查带有库的分区。
- CL-改进了 RTVMGS，可以在找 file/ID 不到 CPF2419 时发送 CPF24 07。
- CL：改进了 CPYTOIMPF 和 CPYFRMIMPF 对遗留格式参数的解读。
- CL：添加了对 OVRPRTF 参数 USRDTA 的支持。
- CL：改进了 CPYTOIMPF CL 命令，以通过建立新连接来避免关闭现有结果集。
- CL：改进了 CHGDTAARA，使其在更新内容时不再修改数据区域长度。
- CL-改进了 ClCommand 数据库连接处理。
- 优化了前端和后端之间的交互。
- COBOL：更新了转换以处理副本中的 FILLER。
- 改进了发送到前端的自定义消息的附加消息信息显示。
- 更新了 app.component.ts 中选择器的默认值。
- 改进了 split-dynamic-field显示屏中的文本拆分。
- 改进了多次写入后读取的错误消息的显示。

横向功能

新特征

增加了对 OAuth2 提供商密钥动态配置的支持。

改进

- 打印：改进了对用于处理引号的 QCMDEXC 参数的支持，并改进了报告名称的编排
- 改进了对分隔语法的 RecordAdaptable支持。
- 增强了 InspectBuilder 错误日志记录功能，可添加有关源字符串的上下文。
- DataSimplifier -增加了情感的坚固性。 ByteArray
- 使用新的运行时属性增强了 MDC 日志记录功能。

现代化工具版本 4.1.0

zOS

新特征

- 添加了对多个 CSD 文件转换的支持
- COBOL：添加了对 CICS ALLOCATE 语句的支持。
- COBOL：在 ADD CORRESPONDING 语句中添加了对 ON SIZE ERROR 的支持。
- COBOL：添加了对 EXIT PARAGRAPH 的支持。

改进

- COBOL：改进了对 -INC 副本的支持。
- COBOL：增强了对 FILLER 初始化的支持。
- COBOL：改进了对象征值比较的支持。
- COBOL：增强了对缺少中间代码块的连续 WHEN 子句中的 WHEN ANY 的支持。
- COBOL：改进了对象征常量的支持。
- COBOL：改进了对计算打包类型的大小的支持。
- COBOL：改进了 SPOOLCLOSE 中未处理的 CICS 参数 KEEP。
- COBOL：改进了 TEST-NUMVAL 函数的生成。
- COBOL：改进了 INSPECT 框架支持上的 Java 生成参数。
- CICS：改进了对定义 DFHCOMMAREA 的支持。

AS400

新特征

- RPG：添加了一种捕错机制来生成（不完整的）DDS，以便不会阻止程序生成。
- 添加了对 INCLUDE 文件描述规范关键字的支持。

改进

- RPG：改进了完全自由式解析
- RPG：通过错误捕获提高了稳健性。
- RPG- field/DS 使用导出关键字改进了初始化。
- RPG：改进了 DAO 操作以处理指示器。
- RPG：使用 CTDATA 处理了 PERRCD 的默认值。
- RPG：升级了免费的 RPG 解析器，以根据解析规则记录唯一的错误。

- PRTF：解决了 PRTF 和 JRXML 之间的名称冲突。
- COBOL：改进了对 LIKE 关键字的支持。

横向功能

改进

- 提高了 ErrorID API 的稳健性
- 大型项目转换的性能优化。例如：跳过被阻止文件超时、重复使用 Blu Insights 中的分类以及更好的内存分配。
- 优化了 COBOL/ PL1 转换期间的内存占用。
- 修复了第三方（jQuery 和引导程序）上的 CVE 问题。
- TC 中的托管 timeoutParser 选项。
- 改进了 SQL 查询的多空格重写功能。
- 改进了带有灵敏度属性的只读游标。

4.0.0 发布说明

发布日期：2024 年 4 月 8 日

有关如何从 AWS Blu Age Runtime 3.10.0 迁移到 4.0.0 的说明，请参阅。the section called “从 3.10.0 迁移到 4.0.0”

此版本的 AWS Blu Age 运行时和现代化工具侧重于升级关键依赖项和支持的技术，同时提高多种功能的性能。此版本中的一些关键特征和更改包括：

- - 从 Spring Boot 2.7 升级到 3.2.4，从 Spring Core 5.3 升级到 6.1.5，以及从 Tomcat 9.0 升级到 10.1.17，通过使用正在积极修补和维护的版本来提高安全性、性能和可维护性。
  - 延迟加载前端应用程序，以构建具有超过 2000 个屏幕的速度更快的大型项目，并将显示初始化时间从 10 秒缩短到 300 毫秒。
  - 支持在前端应用程序上显示 DBCS，以增强对双字节字符的支持，从而提供一种处理双字节和单字节字符的新字体，防止在双字节字段中输入单字节，并处理混合了双字节和单字节字符的字段。
  - AS400 Online 应用程序的线程监控功能，用于通过并行化运行 AS4 00 应用程序。
  - 通过添加可配置机制来预@@ RunUnit初始化程序上下文，从而降低加载传统复杂性固有的复杂结构的影响，从而提高了上下文和初始化的性能。

此版本的 AWS Blu Age Runtime 已使用以下堆栈进行了测试。其他版本可能也兼容。

组件	测试的版本
Java	Java 17
表示层	Node JS 18.18
	Npm 9.8
	Angular 16.1
服务层	Spring Boot 3.2.4
	Spring Core 6.1.5
	Spring statemachine 4.0.0
持久层	PostgreSQL 引擎 14
	Oracle 21
应用程序服务器	Apache Tomcat 10.1.17

有关此版本所包含更改的更多信息，请参阅以下部分。

运行时版本 4.0.0

zOS

新特征

- 添加了对包含语句“-INC CPYNAME”的支持。
- CICS-增加了对 HANDLE 语句的 PUSH/POP 支持。
- COBOL：添加了对“ASSIGN TO DYNAMIC”的支持。
- 添加了对使用 INFUTI DB2 LB 卸载的支持。
- 在 INREC 语句的 OVERLAY 中添加了对关键字 SEQNUM 的支持。

改进

- SORT：添加了对排序字符串字面值 C'....' 中的特殊字符（括号和星号）的支持。
- SORT：改进了对 OUTFIL NOMATCH-(..) 参数的支持。
- SORT：添加了对 SYMNAMES 数据定义的支持。
- SORT：改进了对 TO= 和 LENGTH= 参数的处理。
- SORT：改进了对 MOD 处置的处理。
- SORT：添加了对 HIT=NEXT 参数的支持。
- 增强了 ICEGENER，以添加对特定输出文件编码的支持。
- INFUTILB：增强了对 WITH UR 子句的支持。
- INFUTILB-增强了对假时卸载的支持。 writeNullIndicator
- DSNUTILB：提高了 NULLIF 关键字位于可选的 SQL 关键字之后时的加载步骤的稳健性。
- DSNUTILB：增强了对隔离列名称的支持。
- DSNUTILB：添加了对将空文件加载到表格中的支持。
- DNSUTILB：添加了对 DNSUTILB SYSDISC 文件的 MOD 处置的支持。
- IDCAMS：增强了对评论的支持。
- JCL-增加了对带双引号的列的支持。 LoadTask
- JCL：增强了关于删除空格的 UNLOAD SQL 查询处理功能。
- JCL：增强了 Groovy 脚本在处理过程中发生异常时的响应，以确保 JSON 格式。
- JCL：改进了 DISP=NEW 和 DISP=OLD 情况下的检查文件处置方式。
- JCL：增强了对处理 GDG 基本名称中带有特殊字符的多个 GDG 生成引用的支持。
- JCL：增强了对加载虚拟文件的支持。
- JCL-增强了对 tempFilesDirectory YML 参数的支持。
- JCL：改进了需要对字符串元素中的双引号进行转义时的 JSON 返回。
- JCL-已增强 FileUtils ，支持 GDG 基本名称。
- JCL-用于执行多个查询的增强型 DSNTEP 程序。 DB2
- 添加了对 Spring bean 的支持。
- 增强了功能 SQLConverter ，可避免纠正错误的日期。
- 改进了 YYYYDDD 的 JicsTimeBuilder 操控性。
- 支持从 groovy 访问自定义 jar。
- IMS：增强了 IMS 数据库实现中的跨记录导航。

- IMS：增强了 CBLTDLI，以便能够启动程序使用清除功能。
- IMS：DFSRRC00 能够将参数从 groovy 传递到后端程序。
- 添加了对未通过 transactionRunner 调用的 JICS 命令的支持。
- JICS：通过使用可配置缓存提高了性能。
- <noloc>Blusam</noloc>-添加对在打开时禁用预热 <noloc>Blusam</noloc> 的支持，以增强大型数据集的性能。
- <noloc>Blusam</noloc>-改进了常规 <noloc>Blus</noloc> am 数据集的 delete/rename 行为。
- <noloc>Blusam</noloc>-增强了记录操作的性能。
- 增强了用于确定字符串是否为低值的方法的 datasimplifier。
- 增强了对打包十进制和排序顺序问题的支持。
- 使用 AWS 密钥增强了 DB2 作为主数据源的配置。
- 增强了 FileSystem API，用于显示文件状态。
- 使用 lineSeParator 增强了 DynamicFileBuilder 读取流输入。
- 增强的数据简化器，用于在处理 CUSTOM93 0 个字符集时确定字符串是否为低值的方法。
- SQL：改进了 SQL 存储过程输出处理。
- SQL：改进了带有别名的多个表的 lambda 映射。
- COBOL：改进了对 LENGTH OF 语句的支持。
- COBOL：添加了对 TRANSFORM 语句的支持。
- COBOL：添加了对 9 个新数学函数的支持。
- COBOL-改进了对 INTEGER-OF-DAY函数的支持。
- COBOL：改进了对涉及象征值的 88 级别的支持。
- COBOL：改进了 SET ADDRESS 语句的转换。

AS400

新特征

- 删除了重复的指示器实体。
- 添加了对 DBCS 字符的支持。
- 为子文件记录控制引入了 HELP 关键字的处理。
- 添加了配置参数，以切换列名大小写和通过竖线字符分割注释列内容。

- 添加了对使用 0x0c 作为打包类型字段的最后一个字节的支持。
- RPG-使用 ('system' ExtProc) 声明的处理原型。
- CL：处理了 cl-command RMVMSG 的“CLEAR”参数，并引入了内存内的非程序消息队列。
- CL：处理了传递给 SBMJOB CMD() 调用的通用语句。
- CL：添加了 STRCMTCTL 和 ENDCMTCTL 命令。修改了锁定机制并清理了事务和锁。
- CL：添加了对 CPYTOIMPF 命令的 RCDDLM 参数的支持。
- CL：在 SAVOBJ 命令中添加了对填充零的处理。
- CL：添加了对 RTVOBJD 的 OBJ 参数限定名称中包含的库的处理。
- CL：添加了对 CPYTOIMPF 命令参数 STRDLM、STRESCCHR 和 RMVBLANK 的支持。
- CL-增强型 RTVMGS 可以在找 file/id 不到 CPF2419 时发送 CPF24 07。
- CL：改进了 RCVF 命令，以接收来自 DEV 参数中提供的任何库的记录。

改进

- 更改了 Blu4iv 任务执行器的默认值，以改进默认情况下的扩展。
- 修改了 ParameterHelper，可以将字符串列表转换为字符串 ElementaryRangeReference 。
- 增强了 CTID，以处理 POSTGRE 中不存在的列。
- 提高了稳健性，以支持用户空间 API“QUSPTRUS”。
- 增加了对用户空间 APIs QUSRUSAT 和 QUSCUSAT 的支持。
- 增强了对没有错误代码的用户空间 API（QUSPTRUS）的支持。
- 添加了对使用 Quartz 计划 CRON 作业的支持。
- 增强了对 RPG 程序周期的支持。
- 改进了 Blu4iv 事务管理。
- 改进了同一事务中受提交控制的文件的记录锁定。
- 改进了对子文件初始化的处理。
- 改进了消息行滚动指示器的显示。
- 防止在通过数据队列发送的数字上出现尾随零。
- 改进了其他消息通知屏幕。
- 改进了 JPA 写入操作以考虑当前库。
- 改进了执行不带参数的程序 ProgramJobExecutor 时的行为。

- 添加了将参数从前端链接直接传递到后端脚本的功能。
- 改进了作业元数据的事务处理。
- CL：在 RTVMSG 中添加了对参数 SECLVL 的支持。
- CL：为 CLRLIB 添加了空实现。
- CL：改进了对从数据库和 CSV 进行复制的 CPYFRMIMPF 支持。
- CL：改进了 CPYFRMIMPF 实现以忽略多余的列。
- CL：改进了 CPYTOIMPF 和 CPYFRMIMPF 对遗留格式参数的解读。
- CL-在 SAVOBJ 中添加了 removeDecimalPoint 用于格式化数值的参数。
- CL：改进了 RCVF 命令以正确处理 EOF 条件。
- CL - RTVSYSVAL：实现 SYSVAL = QDATETIME。
- CL：修改了 OVRDBF 命令以获取字段作为默认表名。
- CL - RTVJOBA：USRLIBL 参数的值不可用。
- CL：处理了 SNDPGMMSG MSGF 参数中的前导斜杠。
- CL：改进了命令 DSPFFD 中对源文件中通配符的支持。
- CL：改进了 RCVMSG 和 SNDPGMMSG 中对参数 PGMQ 的处理。
- CL：将 RTVMSG 参数 MSG 设置为可选以与遗留文档保持一致。

横向功能

新特征

- 改进了在 OPEN 游标的 USING 子句中传递参数的功能。
- 性能：改进了上下文的预初始化和性能调整。 RunUnit

改进

- 改进了从 INFUTILB 实用程序的 UNLOAD 命令转储低值的机制。
- 添加了对数据源密钥管理器的当前架构选项的支持。
- 增强了运行时，以在不需要时，不考虑在打开游标时传递参数。
- 改进了数字字段的数字格式验证。
- 改进了高度并行执行环境中的 SQL 诊断。

- 为代码页字节序列（FE FD）引入了 Unicode。
- DataSimplifier 性能优化-增强的赋值语句。
- DataSimplifier 性能优化-改进数字类型初始化的默认值以防止无 BigDecimal 用使用。

现代化工具版本 4.0.0

zOS

新特征

- 添加了对处理 Abend PROGRAM 的支持。
- 改进了对生成 AIX 数据集的支持。
- COBOL-增加了对ALPHANUMERIC/ALPHABETIC/GRAPHIC字段上正当子句的支持。

改进

- 改进了 TRANSCLASS 资源定义的 PURGETHRESH 属性处理。
- 改进了对数据定义和 MOVE 语句的支持。
- CICS：增强了对 DELAY 命令的 MILLISECS 选项的支持。
- 改进了带有别名的多个表的 SQL lambda 映射。
- 改进了对父字段查找的支持。
- 改进了为 COMMIT 和 ROLLBACK 操作设置的 SQLCA sqlstate。
- COBOL：通过注释过时的段落增强了解析功能
- COBOL：增强了对 REPLACING 子句的支持。
- COBOL：添加了对数学函数 ASIN ACOS LOG TAN 的支持。
- COBOL：在 PERFORM VARYING 中添加了对多个 AFTER 语句的支持。
- COBOL：增强了对 RENAMES（级别 66）字段的支持。
- COBOL：增强了 LENGTH OF 方法，用于获取数组字段中特定索引处的长度。
- COBOL：在 PERFORM VARYING 语句中添加了对多个 AFTER 语句的支持。
- COBOL：增强了对 RENAMES 子句的支持。
- COBOL：增强了对 PICTURE 关键字的支持。
- COBOL：增强了对级别 88 字段解析的支持。

- COBOL：改进了具有表数据项的 goto 依赖条件。

AS400

新特征

- 添加了将参数传递给直接前端 java 调用的功能。
- CL：改进了 %SST 生成，包括支持带有 CL→Java 的 *LDA。
- RPG：增加了对 DISK 文件的程序描述记录的支持。

改进

- 改进了显示文件，使用“REFFLD”关键字解析引用的字段。
- 改进了对显示文件关键字 SETOF-CSRLOC 的支持。
- 关闭后从提交控制中删除文件。
- 确保由同一个程序对表执行并发读写操作时的行为一致性。
- 处理了对子字符串的 SizePrefixedAlphanumericType赋值。
- 处理了将数据结构传递给具有不同长度的字符串参数的过程的问题。
- 改进了发生 onBlur 事件时对无效数值的保留以及仅为有效字段创建事件侦听器的功能。
- 改进了屏幕上的错误消息，以突出显示输入无效的字段。
- 改进了对以指示器为条件的屏幕字段的处理。
- 启用了使用鼠标滚轮滚动的功能。
- 为“帮助”屏幕添加了对功能键的支持。
- 改进了对 split-dynamic-field组件中长文本的支持。
- 改进了重命名记录时对多记录 LF 文件的处理。
- CL：改进了 RTVJOBD 命令以处理 LF 文件（视图）。
- CL：改进了 OVRDBF 命令在多记录 LF 上的使用。
- RPG：处理了过程定义了与重命名的参数同名的变量的情况。
- RPG：改进了初始化带符号的 binaryInteger 时对 *ZEROS 的处理。
- RPG：改进了对指向非本地（引用）变量的指针的处理。
- RPG-改进了对语句之后 IFxx 的 ELSEIF 语句的处理。
- RPG：添加了对原型上用 LIKE 定义的字段的支持。

- RPG：改进了对 LIKEREC 创建的字段的 LIKE 关键字的支持。
- RPG：改进了带有象征值的运算符的生成。
- RPG：改进了数组表达式 xxx(*) 的解析并在 %lookup 中支持该表达式。
- RPG-改进了带有高和等于（或低和等于）指示器的 LookUp 操作代码。
- RPG：改进了自由形式解析。
- RPG：改进了对遵循 I-card 记录格式的 I-card 命名常量的解析。
- RPG：改进了对 INTEGER 和 UNSIGNED 类型的支持。
- COBOL：在 COPY DDS 语句中添加了对 DSPF 格式的 INDIC 子句的支持。
- COBOL：改进了 DISPLAY 和 ACCEPT 语句的语法，以解锁转换和生成。
- COBOL：添加了对 DISK 文件的支持。
- COBOL：改进了 DDS 显示文件支持程序。
- COBOL：添加了对 LIKE 子句的支持。
- COBOL：添加了对程序描述的 DISK 文件的支持。
- COBOL：添加了对带后缀的文件名的支持。

横向功能

新特征

- 处理了 Web 项目映射组件的延迟加载问题。

改进

- 改进了 SQL 指示器参数的 java 生成。
- 提高了处理 SET DB2 语句中涉及的变量的能力。
- 改进了当输出为单个实体数组时，在获取的游标末尾引发错误的问题。
- Linux 中的托管路径。
- 数据迁移器可管理漏洞并删除未使用的依赖项。

3.10.0 发布说明

此版本的 AWS Blu Age Runtime 和现代化工具侧重于整个产品的核心基准升级和改进，力求提高所有转型和执行步骤的性能和稳健性。此版本中的一些关键特征和更改包括：

- 从 Java 8 升级到 Java 17，提高了安全性和性能，并支持客户部署和运行以更新式的语言实现的应用程序以及使用最新的第三方框架版本。
- 添加了对管理用户或作业之间的大型共享内存空间、存储应用程序或实例重启后可重复使用的数据的支持。
- 使用分页机制更快地访问 <noloc>Blusam</noloc> 中的大型数据集，从而可以增量检索记录子集。

有关此版本所包含更改的更多信息，请参阅以下部分。

运行时版本 3.10.0

此运行时基于 Java17、Spring2.7 和 Angular16。

zOS

新特征

- <noloc>Blusam</noloc>-通过分页机制增加了对大型数据集的支持，该机制使用页面存储和加载索引

改进

- 增强了 DataUtils .compare，可处理从字符串到数字的较低优先级转换
- 增加了通过 YML 属性 Data ByteRange Simplifier 检查是否使用不正确的值创建了否的支持。byteRangeBounds查看
- 增强了 removeOSI ()，支持初始化 GraphicAlphanumericType 带空字符的
- 提高了作业操作的稳健性以及 GDG 状态读取的安全性
- <noloc>Blusam</noloc>-增加了对通过名为.removeCache () 的新方法清除 <noloc>Blusam</noloc> 数据集的 Ehcache 的支持 CoreBluesamManager
- <noloc>Blusam</noloc>-改进了常规 <noloc>Bl</noloc> usam 数据集的 delete/rename 行为
- Redis：增强了对解锁数据集和清除记录锁的支持
- JICS：改进了失败请求的错误消息
- JCL：添加了对基于点字符的 ControlM 变量串联的支持
- JCL：添加了对 GDG 文件的 Write ADVANCING（ADV）的支持
- JCL：增强了对删除所有 GDG 文件后当前生成号的支持
- JCL-增强了对创建数据集时从目录中 rdw/recordSize 读取的支持

- JCL-添加了在打开具有数据输出记录大小的文件时更新资源对象（从 AbstractSequentialFile）的支持
- JCL：改进了 IDCAMS 的性能
- JCL：通过添加“CHAR”作为“CHARACTER”的别名，增强了对 PRINT 语句的支持
- SORT-增强了对从 <noloc>Blusam</noloc> 固定长度数据集复制到可变长度数据集的复制操作的支持
- SORT：增强了排序语法，以处理一些特定的语句

AS400

新特征

- 增加了对用户空间及其相关内容的支持 APIs
- 添加了对 SNDPGMMSG 的 TOMSGQ 参数的支持，并实现了消息队列
- CL：为 OVRPRTF 命令添加了对 FILE 和 SPLFNAME 参数的支持
- CL：添加了对使用 CPYF 命令处理相应分区表的库的支持
- CL：添加了对处理 CHGCURLIB 命令和构建查询时考虑当前库的支持
- CL：添加了对于将 cl 命令作为调用堆栈跟踪的一部分进行处理的支持

改进

- 经过改进 MessageHandlingBuilder ，可以更好地处理调用堆栈跟踪条目
- 改进了 contextPreconstruct 特征的并行执行
- 改进了 SFLINZ 创建记录时的显示属性
- 改进了 SAVOBJ，以支持处理多个输出文件
- 通过在从 Java 程序调用 programCallStack 时将其添加到 groovy 程序的处理方式来改进这些程序的处理
- 改进了帮助模式的顶部定位检测
- 改进了向 SNDPGMMSG 提供 toMsgQ 参数时的 toPgmQ 功能
- 改进了预定义消息的获取和消息加载器的功能
- 改进了 CPYTOIMPF 对内容中的分隔符的处理
- 改进了 READ 记录的释放锁

横向功能

新特征

- 添加了前端系统消息翻译功能
- 在中添加了一个用于 ExecutionContext 返回程序调用堆栈的新方法
- 无论实际环境如何，都要设置行分隔符（用于数据简化器）
- 增加了配置 SQL 模型 JSON 路径的可能性

改进

- 改进了比较方法 DataUtils。 compareAlphInt() 当涉及填充时
- 创建一个标志，表示支持自定义游标查询中出现异常时的行为
- 改进了图形 LOWVALUES 数据库转换

第三方

- 升级以缓解 CVE-2024-21634、CVE-2023-34055、CVE-2023-34462、 IN1-JAVA-ORGSPRINGFRAMEWORKSECURITY-5905484、CVE-2023-46120、CVE-2023-6481、CVE-2023-6378

现代化工具版本 3.10.0

zOS

改进

- COBOL：添加了对 ABS 函数的支持
- JCL：增强了变量作用域：附加到 STEP 而不是 JOB
- 增强了对 low/high 值的游标参数注入
- 改进了 CSD 解析，尤其是对于远程事务

AS400

改进

- 删除了控制水平指示器的空白检查

- 为 IMPORT/EXPORT 关键字添加了对外部名称的支持
- 添加了对字段上的 %LEN 的支持
- CL：添加了对 CLLE 语言的新运算符的支持
- CL：添加了对嵌套的 IF 的支持
- COBOL：改进了与多个键一起使用时对 START 命令的处理
- DSPF：改进了对包含记录号的游标位置的处理
- DSPF：改进了带符号数字、纯数字字段和大规模字段的格式
- DSPF：改进了屏幕常规帮助的标题确定
- DSPF-改进了对规范的 Input/Output 支持
- DSPF：改进了数值字段验证期间对分组分隔符的处理
- 改进了地图 output/DDS 记录
- 改进了打印机文件 REFFLT 关键字解析引用字段的能力
- RPG：增强了对“ALL free”语句的支持
- RPG：改进了条件解析并添加了对处理不带结果标签的 CABXX 的支持
- RPG：改进了对数值字段的输入规范处理
- RPG-改进了条件内IF/ELSEIF/WHEN过程调用的处理
- RPG：改进了在 dspf 文件上调用 READ 命令时的处理
- RPG：改进了对引用不存在的 DDS 的文件的支持
- 改进了传递物理记录格式名称时对 REFFLD 的处理
- 添加了对使用“return”作为数据库列名的支持

横向功能

新特征

- Oracle：支持定义用户而不是 SYS 来存储内置函数

改进

- 将 Java 版本从 v8 升级到 v17
- 改进了具有 Cluster 列名的 SQL 条件
- 为视图添加了对 ORDER BY 子句的支持

3.9.0 发布说明

此次发布的 AWS Blu Age Runtime 和 Moderization Tools 侧重于整个产品的多项横向增强，旨在提高高可用性架构的性能，以及将任务执行提升到更高水平的新功能。此版本中的一些关键功能和更改包括：

- 版本从 Angular 13 升级到 Angular 16，提高了安全性，并推出可提高客户在线应用程序性能的新功能。
- 在 AS4 00 中添加对跨任务功能的支持，主要亮点是作业可以在它们之间同步发送查询消息，从而在现代化作业中实现脱钩。
- Redis 的使用性能改进，包括连接池优化、连接的高安全性以及升级的数据集锁定机制。

有关此版本所包含更改的更多信息，请参阅以下部分。

运行时版本 3.9.0

zOS

新特征

- 排序程序：更新了固定长度的 VSAM 输入
- JHDB DB：添加了可配置的超时

改进

- 增强了对于流的行分隔符（用于文件串联）的支持
- 增强了对打开串联的顺序文件的支持。文件打开 DataSetIndex 后初始化
- 增强 a NumericEditedType 受数值影响时对虚拟小数分隔符的支持
- 增强了 NumericEditedType 对负值的支持
- IDCAMS：现在可以使用.yml 中定义的 “编码” 属性读取 SYSIN 卡 application-utility-pgm
- IDCAMS：更新了语法以支持 DEFINE CLUSTER 语句中的 FILE(..) 参数
- INFUTILB：添加了对 DFSIGDCB 参数的支持，以覆盖 DD SYSREC 的 DCB 参数
- INFUTIL：增强了对“DFSIGDCB YES”参数的支持
- 改进了 SPLICE 以处理庞大的输入文件
- DFSORT：改进了备注字段的处理

- DFSORT：添加了对（签名/未签名）自由形式数字格式（SFF/UFF）的支持
- SORT：添加了对 OPTION PRINT 和 OPTION ROUTE 语句的解析支持
- SORT/ICEMAN：添加了对封闭式除法运算的支持（带有 DIV 运算符的字段）
- 增强了对使用通用密钥的 CICS READ 的支持
- 函数 StringUtils .chargraphic 已修复，用于从图形类型中删除 SOSI
- 开启时增强性能 DataUtils。 isDoubleByte编码
- JCL：增强了对临时数据集的 KEEP 处置模式的支持。系统将处置改为 PASS
- JCL：动态处理 DCB 参数
- JCL：增强了错误值的 SUM FIELDS 输出
- JCL: Common:DDUtils: getContent 现在在目录中搜索 RecordSize
- JCL：创建数据集时从目录中读取 rdw/recordSize 属性
- JCL：添加了对 DCB=.MYDD 的支持，以便将 DD 的 DCB 参数复制到同一作业步骤中的另一个 DD
- JCL：改进了记录大小继承系统
- JCL：添加了（Redis）独占数据集锁
- Redis：添加了对独立模式的 SSL 支持
- Redis：添加了带锁的同步 Redis 锁定计数
- Redis：支持 Redis 锁的池参数
- Redis：优化了 Redis 的元数据刷新
- Redis：改进了 redis 集群支持
- 改进了 IO 模式下打开的锁
- 改进了数据集锁性能并清除了未使用的锁
- 取消注册文件期间数据集的路径增强
- 改进了预取窗口缓存失效问题
- 添加了对使用线程安全实用程序数据源提供程序的支持
- 增强了 datasetState 无效性检查
- 增强了对不重新打开已打开的数据集的支持
- 提高了作业最终操作的稳健性
- 增强了对支持重复的键的索引顺序的支持
- 增强了对跳过列表序列化顺序的支持
- 添加了对调试转储特征的支持，以帮助诊断索引顺序问题

- 增强了对元数据刷新的支持
- 增强了对 <noloc>Blusam</noloc> 批量读取的支持

AS400

新特征

- 创建应用程序上下文注册表
- 支持 DSPF 关键字 CLRL(NO) 支持记录锁监控
- Support 支持 keyed DataQueue
- 支持批处理作业的 INQUIRY 消息
- 为 00 COBOL 添加了对程序描述的打印机文件的支持 AS4
- 处理 RMVJOBSCDE cl 命令
- 改进了 RUNSQL/DLYJOB
- CHKOBJ：为参数 LIB 引发遗留错误代码
- SNDPGMMSG：支持字符串参数
- RTVDTAARA：改进了 LDA 中的子字符串
- DSPFD：添加了对特定文件名的 FILE 参数支持
- RUNQRY：在 QRY PARAM 中支持 sql 文件
- CRTDUPOB：支持在数据区域之间复制数据
- SBMJOB：将指令转换为使用 JobQueueManager
- OPNQRYF：添加了对 Qtemp 库的支持
- CRTDUPOBJ：改进了复制分区内容的逻辑
- CRTDUPOBJ：为视图添加了对 Qtemp 的支持
- RTVSYSVAL：在 CL 命令中支持 SYSVAL 值 QDATFMT
- CHKOBJ：添加了对 OUTQ 的支持
- RTVJOBA：支持 SWS 参数
- SNDPGMMSG 和 RCVMSG：增加了受支持的参数，包括 MSGF、MSGFLIB、MSGDTA、MSGTYPE、KEYVAR、MSGKEY、MSGID

改进

- 改进的工作站 I/O 卡支持

- 改进了对叠加上一条消息的设置消息的处理
- 支持有关 array-messageline 的其他消息信息
- 改进了 EVAL、SortA、象征值中的独立数组包装器访问权限
- 在线申请结束时改善 DAOs 清洁效果
- 添加了对其他日期格式的支持，并改进了对字符串输入的处理
- 通过添加系统值辅助类 Decode 和 CL 命令生成参数，改进了 SYSVAL 的 CVTDAT 处理 SbmJob
- 已从组件扫描中移除软件包 com.netfective.bluage.gapwalk.rt.blu4iv gapwalk-cl-command
- 改进了对消息队列 API 的预定义消息的支持
- 改进了对 retrieveSubfileRecord 用其他程序编写的记录的支持
- 改进了对消息队列 API 的即时消息的支持
- 改进了提交作业时对本地数据区域的处理
- 服务器启动时 JobQueues 自动启动
- 使用 applicationContext 配置来解码 SBMJOB 的参数
- 改进了系统提供的错误消息
- 允许 RTVMSG 在嵌套的子目录中搜索 .properties 文件
- 处理绑定到 bad/invalid 指针的实体的重置
- 已改进 MessageHandlingBuilder ，可将 msgID 和 MsgFile 名称显示为 RCVMSG 的字符串
- 改进了消息队列 API 的 withMsgFile命名方法
- 改进了数据区域锁定机制
- RTVMBRD：支持参数 FILE 的小写和大写格式
- CRTDUPOBJ：改进了视图的处理
- CPYTOSTMF：改进了对连接的处理
- CPYF：改进了从平面文件中复制时的目录名称处理
- RCVF：正确处理适用于 groovy 和 java 的 RCDFMT 的 DEV/RCDFMT 参数和转换
- RCVF：处理后续调用并避免重置游标
- CPYF：添加了对从平面文件写入的支持
- CRTDUPOBJ：添加了使用 Qtemp 库对新对象的处理
- CHGDTAARA：将数据区域的最大长度从 256 增加到 2000
- SAVOBJ：确保保存的记录按插入顺序排列
- RTVDTAARA：已检索到值（未修剪）

- CHKOBJ：成员不存在时返回正确的监视器消息
- RTVDTAARA：添加了对 LDA 子字符串的支持
- RTVDTAARA：返回不超过 RTNVAR 参数中指定变量长度的空格
- RTVDTAARA：支持起始和长度的整数参数，并支持最新的转换格式
- CHGDTAARA：添加了对包含下限和上限的参数的支持
- CHKOBJ：处理参数对象类型的 VIEW 值
- CHKOBJ：如果视图存在，无论成员如何，结果都设置为 true

横向功能

新特征

- 处理生成报告到 .txt 文件
- 为密钥管理器添加了 currentSchema XA 数据源属性
- 添加了 database.cursor.raise.already.opened.error YAML 属性，以使框架能够在打开已打开的游标时引发 SQLCODE 错误 502

改进

- 在 Amazon 的 AWS Blu Age 包装中添加了 gapwalk poms EC2
- 默认使用新的信号处理程序范式
- 添加对处置为 MOD 或 OLD 时锁定的支持
- 添加了缓存以存储数据库日期时间模式
- 改进了的检查功能 PackedType
- 改进 DataUtils Records 的.setto 函数 VariableSizeArray
- 处理与运行单元有关的 MQ SYNCPOINT 选项
- 使框架能够在回滚事务上设置 SQLCODE
- 根据引擎密钥添加了自动驱动程序类别名称
- 程序/事务超时
- 访问游标时在回滚后恢复游标位置

第三方

- 升级 SnakeyAML、Redisson 和 Amazon SDK， YamlBeans 移除（缓解 CVE-2022-25857、CVE-2023-24621、CVE-2023-42809、CVE-2023-44487）

现代化工具版本 3.9.0

zOS

改进

- 增强了对 XML-TEXT 作为字符串类型目标的源的支持
- 增强了 STM 到 UML 的工作流程，支持 X/(Y/Z）分割模式
- JHDB DB：在任何数据库更新之前接受 ROLLBACK 调用
- JHDB DB：即使事务终止也接受 ROLLBACK (NOP)
- JCL：改进了步骤验证函数
- SORT：处理带有区域十进制负值的 SUM 函数
- COBOL：增加了对字符串文字中 single/double 引号转义的支持

AS400

改进

- 通过添加前导零，改进了内置函数 %editc 对编辑代码 X 的处理
- 改进了对仅输入字段初始值的处理
- 为帮助对话框添加了操作键
- 显示在底部的动态表的页脚记录
- 为指定实际 RECORD-KEY 的文件处理了 START 命令，但没有 KEY PHASE
- 为 float 和 NumberUtils:: pow 类型添加了默认值
- 增加了使用 LIKE(IN）定义变量的支持
- 更新了 FOR 循环处理以支持省略可选元素
- 更新了 RPG 解析以将记录与 CTDATA 数组名称相关联
- 改进了对 CABxx 报表指标的处理
- 支持 COMMIT 关键字上的可选参数
- 改进了 LF 中的 FORMAT 关键字支持

- 管理包含高于等于（或低于等于）指示器的 LOOKUP 操作代码
- 处理了双引号中声明的 PF 键名称
- 改进了对 EDTCDE X 的处理，使其不会抑制前导零
- 改进了对打印机文件中 MSGCON 不生成未命名标签的支持
- 字段 CONTENT 由多个数据结构共享
- 结合 SFLMSG/SFLMSGID 处理了 ERRSFL 参数
- 改进了完全免费 rpg 的 proc 声明范围之前的主代码
- 添加了条件控制规范解析
- 改进了 dataholdermapper 中对 setErrSfl () 方法的支持
- 改进了内部创建的变量的类型解析
- 改进了对 Z-ADD opcode 的支持
- 改进了对带有 DFT 值的常量字段的处理
- 改进了对程序状态 ds 中整数字段的支持
- 处理了 ENTRY 参数中的指示器分配
- 改进了通过 ref/reffield 关键字传播的关键字的过滤器
- 支持的未命名 DataArea 数据结构
- 改进了对指针数据类型的处理
- 处理了用于使用输出字段中的 LIKE 关键字支持数组访问定义变量的数组元素
- 改进了对带符号数字的支持，仅显示数字
- 支持 O 卡上的逻辑关系
- 字母数字上 %CHAR 的测试用例
- 支持控件规范关键字 main
- 带有打印机文件中两个参数的 EDTCDE
- 改进了 FullFree RPG 解析
- 增强了动态表，以确保页脚正确定位
- 添加了对使用 ALL 象征常量初始化数字类型的支持
- 改进了对引用相同物理文件的多个 RPG 逻辑文件的处理
- 改进了现代屏幕中的修改字段检测
- 与动态字段模态同步

- 改进了对仅输出有符号数值字段的处理
- 改进工作站 I/O 卡支持

横向功能

新特征

- 数据迁移器工具：添加了 ebcdicFilesWith VarcharIn VB 属性，允许在读取字节时考虑 VARCHAR 2 字节长度
- 实现了通用 API 来记录错误
- 实现 BluAgeErrorDictionaryUtils 和使用通用 API 来记录 COBOL2模型、 RPGCycle生成器、Definitions2Model 中的错误和/或信息 FieldsProcessor
- 改进了 SQL 语法以支持不同的隔离子句定义

改进

- 将 Angular 版本升级到 v16
- Angular：将 ajv 版本从 6 升级到 8.9

第三方

- 将 Groovy 升级到版本 2.4.15

3.8.0 发布说明

此版本的 AWS Blu Age Runtime 和 Moderization Tools 侧重于对整个产品进行多项横向增强，以提高其质量和安全性，同时提高缓存性能和统一单个发行版中的命令支持。此版本中的一些关键特征和更改包括：

- 从 Spring 2.5 升级到 Spring 2.7 版本，提高了平台的维护支持、性能和安全性。
- 作为 over-the-counter发行版的一部分，统一了 82 多个 CL 命令支持，以促进以前使用 CL 脚本的现代化应用程序的使用和部署。
- 新功能 APIs 可用于更好地操作 Blusam 数据集并与 <noloc>Blusam</noloc> 数据集进行交互，例如集成导入托管服务以及列出数据集元数据信息的功能。
- Redis 的性能改进且使用范围扩展，包括集群模式下的可用性、高可用性数据检索、密钥使用的标准化。

有关此版本所包含更改的更多信息，请参阅以下部分。

运行时版本 3.8.0

zOS

新特征

- 将密钥定义作为字符串处理 DynamicFileBuilder
- DFSORT：在 OUTFIL TRAILER1 + DFSORT 语法初始化中添加了对多项的支持
- 常用DDUtils 工具：处理流内数据中的记录大小
- 索引文件：处理 GENKEY 选项

改进

- 外部化的 <noloc>Blusam</noloc> 在单独的 jar 中加载服务
- 增加了支持设置存储临时文件的位置
- 改进了多节点情况下的共享缓存机制
- 共享缓存使用情况：IDCAMS 验证优化
- 改进了嵌入式选择的 ROWID 注入
- JCL：现在，每个流内作业过程是在不同的 Groovy 文件中生成的
- 确保 IDCAMS JCL 卡上有 card-demo-v 2 个覆盖范围
- <noloc>Blusam</noloc>：使用多个实例时避免重复预热
- 减少了缓存水化的内存占用
- Jedis 池配置支持
- 为流添加了行分隔符（用于文件串联时）
- 在 IDCAMS 实用程序中支持 EBCDIC 卡 + 屏蔽注释 (/* ... /)
- 数据库支持查询：在将 level49 转换为 SQL 时支持双字节字符串
- DFSORT 语法：实现 17 条控制语句 + 整合其中 2 条控制语句 (OMIT/INCLUDE)
- 增强 GRAPHIC 列获取 INFUTILB
- 支持读取带有可变大小表格的文件
- Support 支持 ZonedType 带有 nibble 符号的 nibble 符号，其中最后一个字节的第一位是 'E'
- 如果记录与任何 CHANGE 查找常量都不匹配，DFSORT/ICETOOL 会添加对 NOMATCH=(..) 参数的支持

- Redis 集群兼容性
- 根据 groovy 退出代码处理作业状态（失败）
- 改进了 CCICS SYNCPOINT ROLLBACK 支持。
- 预取窗口优化 Redis 缓存使用
- JCL/GROOVY：当 DISP=(,PASS) 时，从上一步的数据集中继承 isRDW 属性
- 处理具有可变大小数组的数据的部分副本

AS400

新特征

- Support 支持用于显示文件的 I/O 卡片
- 支持 DSPF 关键词 ERRMSGID 和 CHKMSGID 的其他消息信息
- 支持在前端屏幕上显示多条错误消息
- 在 gapwalk-cl-command应用程序中添加或改进了对 82 个 CL 命令的支持

改进

- 改进了对提交控制下的删除和读取的支持
- ConvertDate 在内置 %dec 的内部
- 强制使用 XSS 安全标头
- 提高了 STM 生成的稳健性和一致性（更好地处理：自由格式 rpg 中的连续行、小数部分的逗号、定义/声明中的自由形式块）
- 改进了 DataHolderMapper 生成
- 增加了稳健性并更改了范围 DataAreaFactory
- 改进了 Tab 键上的焦点移动
- 提高了 Jasper 报告生成的性能
- 改进了带有填充 0 的十进制显示
- 改进了 INFDS 中对 ROW/COL 外场的支持
- 改进了对从屏幕上修改字段的支持
- 为生成的报告名称和路径添加了 getter
- 改进了数据队列长度

- 改进了作业队列的自动配置，以便符合 Spring Boot 2.7 中的新标准
- 改进了多个并发会话的工作站更新

横向功能

新特征

- 对已打包数据支持“不容忍无效数据”
- 已 pagination/filtering 添加到列表数据集终端节点

改进

- 增强了 ORACLE 查询转换策略，用于将列与空字符串进行比较
- DB2 使用 DSNTEP 和 INFUTILB 实用程序处理 BLOB。BLOB 现在已现代化 DB2 为 BYTEA 类型的 postgres。
- 改进了对游标最后一项的删除
- 增强了对删除 RRDS 文件的支持
- 提高了 AWS <noloc>Blusam</noloc> 的秘密性能
- 改进了 SQL 框架中对数据库连接的处理
- 标准化的 AWS 多数据源密钥管理器密钥
- 性能回归修复
- 改进了的检查功能 PackedType
- 改进了 LOW-VALUE 的处理 PackedType
- 升级了用于 Cognito 连接的 spring 安全封装
- 未在目标数据库上应用 codeshiftpoint 编码和解码 DB2

第三方

- Spring Boot 从版本 2.5 升级到 2.7

现代化工具版本 3.8.0

zOS

新特征

- JCL: 使用回车符“\r”处理流

改进

- 改进了日志记录以防止在对带有 ON SIZE ERROR 子句的 DIVIDE 进行现代化时除以零
- JCL：增强了对在过程中调用过程的支持
- 当存在模棱两可的字段时，支持在 FORMATTIME CICS 命令中使用 OF 关键字
- JCL：支持在变量中使用的 Â¥ 字符
- JCL：根据前面的步骤计算 RC
- 使用 PL1 SUBSTR 时比较字节而不是字符串
- 改进了单个源的多维数组的初始化
- 改进了 COBOL 在 IF 块中涉及单个 SQL 查询时的解析

AS400

新特征

- 支持 CL 中嵌套 IF 语句
- 改进了 RPG freeform 中对 ENDDO 语句的支持

改进

- 改进对调节控制级别的支持
- 改进了使用 LIKE 的原型返回值
- 改进了对处理函数 %months、%year、%days 的支持
- 支持整个屏幕的帮助特征
- 处理作为参数传递的象征性 BLANKS
- 使用 "" 运算符对表达式 EVAL 进行了改进
- 处理没有 KEY PHASE 的 START 命令
- 改进了对关键字 LIKEREC 的处理
- 改进了未命名的子字段
- 改进了返回无符号类型的过程
- 改进了对 RESET 操作（免费 RPG）、%CHAR 和 %DEC 内置函数的支持

- 改进了内置函数 %LOOKUPXX
- 改进了在没有原型的过程上对 LIKEDS 关键字的支持
- 处理 Dim 关键字数组类型（VAR、AUTO）
- 改进了对 XFOOT 的支持
- COBOL：改进了对 RENAMES 字段的支持
- CL：支持 while(true) 条件
- 使用 LIKE 关键字改进对独立数组的处理
- 改进了内置函数 %INT
- 改进了 RPG Full Free 解析
- 改进了对关联中数组的支持
- CL2GROOVY：Support Select 声明
- 改进了 DSPF 关键字“ERRMSGID”
- 改进了使用前导零初始化字节的处理
- 改进了数值字段的 authorizedValues
- 处理 Free form EVAL 语句的头部说明符 H
- CL 到 Groovy：支持 LDA 子字符串
- 改进了对记录 RESET 支持
- 使用参考改进了对 EDTCDE 和 EDTWRD 的处理
- 使用 DDS 字段改进了输入字段映射
- 改进了对于 MOVEA 字符到 IN 数组的支持
- 使用 LIKEDS 关键字改进了原型
- 改进了对 DSPF 关键字 DSPATR 的支持
- 使用 +/- 改进了 D 卡的解析
- 提升了程序调用的稳健性
- 提升了字段解析过程的稳健性

横向功能

改进

- FrontEnd: 模拟 IME 输入的粘贴事件

第三方

- Spring Boot 从版本 2.5 升级到 2.7

3.7.0 发布说明

此版本的 AWS Blu Age 运行时和现代化工具主要包括用于更好地支持命令和实用程序的增强功能、与 AWS Secrets Manager 集成的功能以及新的监控功能。此版本中的一些关键更改包括：

- 现在，多个运行时组件可以使用 AWS Secrets Manager 来增强现代化应用程序的安全设置，这些应用程序主要与实用程序数据源、TS 队列的 Redis、<noloc>Blusam</noloc> 缓存和锁有关。
- 监控允许检索有关资源使用优化和运营管理的事务、批处理和 JVM 指标的端点，例如状态、持续时间、数量等。
- 支持 RPG 中的 IBM MQ 调用的新功能，并增加了 JCL SORT 和 IDCAMS 转换覆盖范围。

有关此版本所包含更改的更多信息，请参阅以下部分。

运行时版本 3.7.0

主题

- zOS
- AS400
- 横向功能

zOS

新特征

- 使用类似 SQL 的语法，改进对实用程序应用程序中涉及的查询的解析。(V7-9401)
- 偏移时处理可变大小索引数组 (V7-9904)
- Support 支持 DB2 以 24:00:00 小时格式插入 SQL TIME 列 (V7-10023)
- 支持从带有 FOR ROWS 和 ATOMIC 选项的数组中插入 SQL 查询 (V7-10105)
- JCL SORT-使用 IFTHE TranscodeTool N 增强功能以支持 OUTREC (V7-10124)
- JCL SORT – 在 OUTREC 命令中添加对 DATE 关键字的支持 (V7-10125)
- JCL – 增加对数据流过程的支持 (V7-10223)

改进

- 标有“PASS”处置的数据集应在所有作业步骤中可用 (V7-9504)
- 支持 JCL 属性 SCHENV (V7-9570)
- 支持带有 CTLCHAR 选项的 SEND (V7-9714)
- COBOL – 在 ACCEPT 语句中处理不同的行分隔符字符集 (V7-9875)
- 避免多次回滚 (V7-9958)
- 允许使用 MOD 处置附加在 GDG 文件末尾 (V7-10031)
- 优化：putAll 重构 (V7-10063)
- PutAll 重构：添加分页 (V7-10063)
- 使 Jedis 客户端读取超时可配置 (V7-10063)
- UseSsl 支持独立模式 (V7-10114)
- 成功打开文件后支持 EIBDS (V7-10147)
- 在收到文件控制请求后支持 EIBDS (V7-10147)
- 改进了 CICS SYNCPOINT 支持 (V7-10187)
- BluesamRedisSerializer: MetadataPersistensate 存在问题 (V7-10202)
- 支持 Redis AWS Secrets Manager 用于 TS 队列 (V7-10204)
- 在自定义 DD 名称大小时支持 JCLBCICS (V7-10224)
- 在 IDCAMS DELETE 语句中添加对绝对路径的支持 (V7-10308)

AS400

新特征

- 实现 AS4 00 个屏幕的帮助功能 (V7-9673)

改进

- INFDS 中的记录数量 (V7-9377)

横向功能

新特征

- 支持 Runtime 开 EC2 启向亚马逊发送日志 CloudWatch (D87990246)
- 添加了新的端点，用于检索有关批处理、事务和 JVM 的指标 (D88393832)

改进

- 支持数据源 AWS Secrets Manager 用于实用程序 pgm (V7-9570)
- 添加了 Db2 对 DSNUTILB DISCARD 的支持 (V7-9798)
- 支持写入记录器而不是默认 SYSPRINT 和 SYSPUNCH 文件中的默认系统输出流 (V7-10098)
- 支持 <noloc>Blusam</noloc> Redis 在 AWS Secrets Manager 中缓存和锁定连接属性 (V7-10238)
- 支持 Db2 XA AWS 密钥上的 SSL 连接 (V7-10258)
- 更新了 IDCAMS REPRO 和 VERIFY 的元数据 (V7-10281)
- 改进了 IDCAMS Abend 返回码管理 (V7-10307)

现代化工具版本 3.7.0

主题

- zOS
- AS400
- 横向功能

zOS

新特征

- PLI – 改进了数组截面和二维数组的赋值 (V7-9830)

AS400

新特征

- 处理控制水平指示器 (V7-9227)
- 支持 EXTNAME 参数 *INPUT (V7-9897)
- 增强的 Goto 重写：支持位于 SELECT OTHER 语句中的标签 (V7-9973)
- 支持 REFSHIT DSPF 关键字 (V7-10049)

改进

- 改进了处理文件描述关键字 EXMIND (*INUx) (V7-7404)
- 改进了 SQLDDS 文件转换 (V7-7687)
- 不再 AS4为 00 个文件生成文件对象 (V7-9062)
- 改进了对文件描述关键字 EXTDESC 的处理 (V7-9268)
- 改进了对内置 %CHAR 的处理 (V7-9311)
- 改进了对没有 SFLEND 的最后一条记录的向下翻页支持 (V7-9322)
- 改进了对前缀数据结构的支持 (V7-9436)
- 支持使用 %SIZE 定义的维度 (V7-9472)
- 支持对使用双引号声明的 PF 字段名进行处理 (V7-9557)
- 改进了文件操作，不区分大小写 (V7-9785)
- 支持初始化为 *USER 的字段 (V7-9806)
- 支持 AS4 00 中的 COMP 类型 (V7-9840)
- 改进了 COBOL4 00 解析开启（不是）InvalidKey (V7-9922)
- 改进了对 SCAN 操作的处理 (V7-9971)
- 改进了对 GOTO 操作码的支持 (V7-9973)
- 改进了对 EXCEPT 操作的处理 (V7-9977)
- 改进了对前缀的支持 (V7-10000)
- 支持 RPG 中的 MQ 调用 (V7-10007)
- 改进了内置 %LOOKUP（键控数组数据结构）(V7-10022)
- 支持 Close *All 操作 (V7-10036)
- 支持 UPDATE AS ROW CHANGE SQLDDS 语句 (V7-10051)
- 改进了对字面值类型 Long 的处理 (V7-10073)
- 改进了 RPG 语法（使用关键字 INZ 作为子例程的名称）(V7-10074)
- 改进了 RPG 语法，支持小数部分为空的数值 (V7-10077)
- 改进了对 CL 和外部文件之间共享的字段的支持 (V7-10081)
- 改进了对 DDS 条件指示器的支持 (V7-10084)
- 通过 COBOL 程序支持 DDS 二进制类型 (V7-10100)
- 改进了链接的名称冲突 (V7-10109)
- 支持混合主过程和导出过程 (V7-10112)

- 改进了对子过程 DataStructure 中的支持 (V7-10113)
- 改进了对 CLEAR 的支持 (V7-10126)
- 改进了对 DO 回路的支持 (V7-10134)
- 在 Full-Free RPG 中支持 SQLTYPE (V7-10151)
- 改进了 DDS 关键字条件的解析 (V7-10155)
- 改进了 DSL 生成 (V7-10163)
- 改进了条件为二进制表达式时的 processIndicators。(V7-10164)
- GOTOs 使用 Else 状态进行了改进 (V7-10168)
- 支持 DSPF 中的时间和时间戳类型 (V7-10173)
- 改进了 DDS 连续行的解析 (V7-10183)
- COBOL 支持 RENAMES FLD OF RECORD (V7-10195)
- 改进了 DSPF 字段上的条件指示器解析 (V7-10221)
- 支持解析 DDS 关键字 NOALTSEQ (V7-10288)
- 支持"帮助"菜单和隐藏字段 (V7-10314)
- 改进了 DSPF 帮助关键字完整性检查 (V7-10328)
- 不再传播 Ref 字段上的所有关键字 (V7-10347)

横向功能

新特征

- 数据迁移器 – 处理 CLOB 数据 (V7-9665)

改进

- 通过 (V7-10225) 将 JCL 属性 SCHENV 从 JOB 传播到 PROC GROOVY 定义 JobContext
- FrontEnd -在没有边框的情况下调整窗口大小 (V7-10358)

3.6.0 发布说明

此版本的 AWS Blu Age 运行时和现代化工具为 zOS 和 AS4 00 传统迁移提供了新功能，主要面向扩展 CICS 支持机制、补充 JCL 功能、优化并发和大容量功能的性能以及添加功能。 multi-data-source此版本中的一些关键更改包括：

- 增强 JCL 动态文件处理、扩展当前语句和连接数据集的管理、在单个块中执行多个语句以及将数据从批处理传输到程序。
- 增强了对多个 CICS 命令的支持，包括对多种 CICS 资源类型的查询。
- 在使用 Blu Age Runtime Utilities 时能够使用不同的数据库，最适合业务数据分布在多个来源的场景。

有关此版本所包含更改的更多信息，请参阅以下部分。

运行时版本 3.6.0

主题

- zOS
- AS400
- 横向功能

zOS

新特征

- JCL- DynamicFileBuilder -增强的文件句柄管理 (V7-9408)
- 增强了在调用 INFUTILB UNLOAD 实用程序时对某些内置 SQL DB2 函数的格式转换 (V7-9554)
- 增强了 PLI 多维数组赋值 (V7-9592)
- 将 sysout 重定向到文件 (V7-9992)

改进

- 为 DB2 RDBMS 添加存储过程触发器 (V7-9155)
- SORT 处理结果转换为 PDF 格式 (V7-9286)
- JCL/GROOVY – 增强 REPRO 语句，以便支持虚拟数据集 (V7-9424)
- 改进了 CICS UNLOCK 支持 (V7-9606)
- 处理 Union 的默认值大小 (V7-9648)
- JCL/GROOVY handle different termination/disposition在串联的数据集中 (V7-9653)
- 为 <noloc>Blusam</noloc> 数据集配置页面大小 (V7-9680)
- DSNUTIL-允许在 LUW 中 DB2加载 24:00:00 作为有效时间 (V7-9697)

- 支持 NumberUtils .ne ()/ NumberUtils.eq () 中的高值 (0xff) 比较 (V7-9731)
- JCL/GROOVY – 支持在 IDCAMS IF-THEN-ELSE 子句中使用 DO... 然后在 IDCAMS IF-THEN-ELSE 子句中使用关键字在单个块中执行多条语句 (V7-9750)
- 在 R JHDBBatch unner 之外调用的 JHDB 程序无效 (V7-9782)
- 支持 SORT OUTFIL 控制卡中的空格字符 (V7-9808)
- 改进了 CICS READ PREV 支持 (V7-9845)
- 改进了对数据集索引的并发访问 (V7-9864)
- 改进了 CICS REWRITE 支持 (V7-9873)
- COBOL – 支持在 ACCEPT 语句中使用多行 SYSIN 将数据从批处理（JCL）传递到程序（COBOL）（V7-9875）
- Groovy-更好地处理文件创建步骤 (V7-9876) ConcatenatedFileConfiguration
- IDCAMS 实用程序 – 处理 DEFINE PATH 语句 (V7-9878)
- SORT BUILD – 调整 TRAN 选项并处理隐式空白 (V7-9925)
- 通过 GENERIC 选项支持改进 CICS DELETE (V7-9939)
- 改进了 CICS STARTBR 和 ENDBR 支持 (V7-9952)
- 提高了并发访问的关闭性能 (V7-9953)
- 改进了启动时的文件状态处理 (V7-9991)
- Groovy-允许在 (V7-10012) 上调用 getDisposition getNormalTermination ConcatenatedFileConfiguration ()/getAbnormalTermination()/()

AS400

新特征

- 支持 COMMIT 关键词上的外部指示器 (V7-6035)
- 写入 SFLCTL 后重置 ReadC 循环 (V7-8061)
- 支持 CALL 中的 LR 指示器 (V7-9250)
- 添加新类型的动态字段（拆分），以便处理多行上的输入字段 (V7-9370)
- Support primary/secondary 文件 (V7-9390)
- 目前，提交作业时，本地数据区会传递给被调用的作业 (V7-9775)
- QTEMP 支持数据区域，支持数据区域值创建。(V7-9916)

- 承诺控制-支持 enable/disable 承诺控制 (V7-9956)
- 支持 COMMIT 关键词上的外部指示器

改进

- 改进 0 值显示和 EDTWRD (V7-8933)
- 支持 DSPF 关键词“CHKMSGID”(V7-9125)
- 在批处理终止时提交 SQL 事务 (V7-9232)
- 改进对字段和数据结构的关键字 EXPORT 和 IMPORT 的支持 (V7-9265)
- Support 中支持小写 DateHelper 字母 (V7-9461)
- 支持将 *CYMD 转换为 *ISO（数字）(V7-9488)
- 改进内置 %len 针对变化字段（表达式的左侧和右侧）的处理 (V7-9733)
- 改善对内置函数 '%LOOKUPXX' XX（“LE”、“LT”、“GE”、“GT”）的支持 (V7-10064)

横向功能

新特征

- CICS – 改进选项状态的查询事务 (V7-9712)
- JCL – 使用系统输出文件改进 sysprint 的负载 (V7-9797)
- CICS – 改进 INQUEU TSQUEUE (V7-9823)
- CICS – 改进选项 userid 的查询终端 (V7-9906)

改进

- 改进与空白比较的处理方式 (V7-8047)
- 改进 Jics 和 <noloc>Blusam</noloc> 的日志记录 (V7-8847)
- 支持 BMS 扩展属性 SOSI 和动态字段的编程符号 F8 (V7-8857)
- 处理程序参数中的缓冲区溢出 (V7-9138)
- 改进 <noloc>Blusam</noloc> 锁注册表的线程写入并发性 (V7-9505)
- 支持 Utility-pgm 的多个数据源配置 (V7-9570)
- <noloc>Blusam</noloc> 仅限录制关卡锁定模式 (V7-9626)
- 确保元数据在服务器重新启动后保持不变 (V7-9748)

- 改进发生异常时的 DAO 清理（浏览器关闭）(V7-9790)
- 支持 INFUT DummyFile ILB SYSPUNCH (V7-9799)
- 在 NumericEditedType (V7-9935) 上增强对负值的支持

现代化工具版本 3.6.0

主题

- [zOS](#)
- [AS400](#)
- [横向功能](#)

zOS

新特征

- JCL – 增强过程结束时的日志记录 (V7-8509)
- PL1 -增强数据类型的袋子生成 PakedLong (V7-8917)
- JCL – 增强在文件包含“结束”标记的情况下过程结束时的日志记录功能 // (V7-9509)
- PL1 -使用定点和 SYSIN 流增强对 GET EDIT 的支持 (V7-9593)
- DB2 -增强对 VARGRAPHIC DB2 类型的支持 (V7-9809)
- CICS – 提高选项 LOGMESSAGE 的命令查询安全性 (V7-9969)
- PL1 -改善 CHARG/chargraphic 内置袋子的生成 (V7-9989)

改进

- PL1-增强对 INCLUDEX 关键字的支持 (V7-9588)
- PL/I-将 CHARGRAPHIC 关键字作为任何方法调用的有效参数处理 (V7-9589)
- 当使用特定字符命名时，可以提高 PL1 主机变量的分辨率 @ # $ §。(V7-9654)
- COBOL – 支持 C01...C12 和 S01...S05 关键字作为解析步骤 WRITE ADVANCING 语句的参数 (V7-9669)

AS400

新特征

- 支持分析器中的 SQL-DDS 转换 (V7-7687)
- 自动检测 SQL-DDS 文件 (V7-7687)
- 实现 SQL-DDS 预处理 (V7-7687)
- 支持 ALIGN 关键字 (V7-9254)
- 支持 ExtName DSPF 和多维阵列 (V7-9663)
- COBOL WRITE 上的 Support InvalidKey 声明 (V7-9793)

改进

- 改进了 TESTB 操作码 (V7-8865)
- 改进了焦点上对 DECFMT 的支持 (V7-8933)
- 处理 MOVE 生成的指示器 (V7-9224)
- 改进了对字段和数据结构的关键字 TEMPLATE 的支持 (V7-9278)
- 改进了 LIKEDS（使用 LIKEDS 定义的 DS 会自动满足要求）(V7-9302)
- COBOL – 改进指示器结构生成 (V7-9423)
- 原型中的常量参数不是只读的 (V7-9437)
- 使用编辑代码“Y”改进 EDTCDE 关键字 (V7-9443)
- 支持在 PSDS 和 INFDS 中生成 *ROUTINE 字段 (V7-9487)
- 将重写字段 XXX 改为独立字段（重写时默认值会丢失）(V7-9522)
- 改进了对 DSPF 关键词的支持 (V7-9658)
- 处理二进制的零默认值 (V7-9666)
- 支持隐式指针 (V7-9719)
- 使用一个参数改进对内置调用 %size 的处理 (V7-9730)
- 改进内置调用 (%ELEM) 中对数据结构引用的处理 (V7-9736)
- 改进定义规范中带有 LIKE 引用的字段的带符号长度的处理 (V7-9738)
- 改进 REWRITE (V7-9791)
- 改进从 DDS 文件生成索引的方法 (V7-9803)
- 提高具有无效数值的映射器的稳健性 (V7-9813)
- 改进 SQLModel 和 allIndexes 文件生成 (V7-9818)
- 改进限定 DS 支持 (V7-9863)

- 改进对 LOOKUP 的支持（在参数中使用类似 DS 的独立字段）(V7-9961)
- 在指示器上改进 LIKE (V7-9985)
- 处理 MVR 上生成的指示器 (V7-9224)
- 支持带波浪号的字符 N (V7-10021)
- 改进从 SQLDDS 遗留文件生成现代 DDL 文件（V7-10067）

横向功能

新特征

- 使用 yml 属性自定义资源位置 (D88816105)
- COBOL – 支持 EXIT PERFORM 语句在不使用 GO TO/PERFORM ... THROUGH 的情况下退出内联 PERFORM (V7-9582)
- 指定要在全局元数据中考虑的默认遗留编码。(V7-9883)

改进

- 改进掩码生成 (V7-9602)
- 改进上下文预热 (V7-9621)
- 使 Charset CUSTOM93 0 线程安全。(V7-9674)
- 改进 MOVEA (V7-9773)

3.5.0 发布说明

此版本的 AWS Blu Age 运行时和现代化工具为 zOS 和 AS4 00 传统迁移提供了新功能，主要面向数据集和消息优化，以及扩展的 Java 功能，以此作为转换过程的资产。此版本中的一些关键更改包括：

- 除了先前存在的 groovy 脚本特征外，还能够将 CL 程序迁移到 Java，可促进其与其他现代化程序的集成，并通过统一生成的编程语言来简化客户的学习曲线。
- 借助新的批量数据特征，缩短时间并优化 Redis 中数据集加载的性能。
- 能够在作业步骤中操作和传递数据集，实现传统数据集行为的现代化。
- 扩展了 SQL 迁移来支持 VB 输入文件，Java 11 简化了迁移。
- 多种新机制可更快地与 IBM MQ 集成，包括额外的标头、扩展 GET/PUT 支持和自动检索队列元数据。

- REST 端点用于数据集元数据和从 S3 存储桶导入数据集。

有关此版本所包含更改的更多信息，请参阅以下部分。

运行时版本 3.5.0

主题

- zOS
- AS400
- 横向功能

zOS

新特征

- JCL SORT – 处理新的关键字叠加 (V7-9409)
- ZOS COBOL – 增强对浮动字符的支持 (V7-9404)
- RedisJicsTSQueue 到 RedisTemplate & 的港口 ListOperations (V7-9212)
- ZOS JCL-如果通过文件目录定义，则使用文件目录增强临时目录的路径 UserDefinedParameters (V7-9012)
- 使用全部（所有数组项目）处理函数 ORD-MAX (V7-9366)
- 在 Redis 中存储 TS 队列时，使用人类可读的带前缀密钥 (V7-9212)
- 为 <noloc>Blus</noloc> am API 添加获取数据集端点
- JCL – 添加对名称包含 # 等特殊字符的批处理作业的支持 (V7-9136)
- TSModel 现在可以按需稳健地执行抓取 (V7-9212)

改进

- LNK 文件中支持非版本控制 INCLUDE (V7-6022)
- MQ – 增强编码支持 (V7-9652)
- 改进对不同字符类型的双字节或混合字符集的支持 (V7-9596)
- JCL – 支持 IDCAMS delete NONVSAM 语句中的 filesDirectory 配置 (V7-9609)
- 支持采用批量模式从文件加载 ESDS 和 RRDS 数据集 (V7-8639)
- 在输入模式下处理空 ESDS 的打开。(V7-9287)

- 通过 ORD/UNORD 缩写支持增强定义集群语句 (V7-9451)
- <noloc>Blusam</noloc> Redis 锁性能改进 (V7-8639)
- 增强 DEFINE CLUSTER 语句，以便支持 DATA () 参数作用域中提供的 RECORDSIZE (V7-9337)
- 在 DEFINE CLUSTER 语句上添加对 BUFFERSPACE/UNIQUE 属性的支持 (V7-9419)
- 改进了可变长度记录数据集的 <noloc>Blusam</noloc> 读取操作。(V7-9391)
- CICS 地址正确地将缺失的 CWA 表示为空 (V7-9491)
- 移除结尾锁定时不必要的写入 (V7-8639)
- 处理缓存中的 Redis 缓存模板注入 (V7-9510)
- 正确解码 BPXWDYN 参数 (V7-9417)
- 改进 LISTCAT 的导出消耗 (V7-9201)
- <noloc>Blusam</noloc> TS 队列名称中支持不可打印的字符 (V7-9212)
- 处理地图集为空的字段的接收地图构建 (V7-9486)
- 改进动态访问模式的 BluesamRelativeFile 删除和重写操作。(V7-8989)

AS400

新特征

- 添加一项功能，通过标准 DS/STM 透视生成作为 Java 程序的 CL 文件 (V7-9427)
- 支持 ADD 模式下的输入文件 (V7-9378)
- 改进了排序顺序和检索管理以支持 cl 命令 OPNQRYF（打开查询文件），并在中添加了对 SHARE 参数的支持。 OverrideItem(V7-9364)

改进

- 在 (V7-8061) 上支持 SFLNXTCHG UpdateSubfile
- 运行 CL 命令时修改 CL 上下文的范围 (V7-9624)
- 处理程序 BPXWDYN 的返回码 (V7-9417)
- 清除本地显示器。(V7-9624)
- 支持 DSPF 关键字 RTNCSRLOC (V7-9389)
- setOnGreaterOrEqual() 未设置等于 1 (V7-9342)
- 开启更新字段缓存 UpdateSubfileRecord (V7-9376)

- 改进 SFLNXTCHG 支持 (V7-8061)

横向功能

新特征

- 忽略文字图形字符串上的 G 前缀。(V7-9420)
- ZOS COBOL – 增强对某些特殊结构的 Fiedl.initialize() 支持 (V7-9485)
- 允许异步初始化上下文，以便改善程序启动性能 (V7-9446)
- SQL 明确释放打开的准备语句和 ResulSet。(V7-9422)
- 增强 JMS MQ- MQRFH2 支持 MQ PUT/V7-7085-支持默认队列管理器 (V7-9400)
- SQL 管理：启用 SET 命令参数的 Lambda 转换 (V7-9492)
- ZOS MQ JMS – 添加对 MQCOMIT 和 MQBACK 的支持 (V7-9399)
- ZOS IBMMQ – 增强对 MQINQ 的支持 (V7-9544)
- 使用双字节编码时，使用字节而不是字符串处理 CONCAT 操作。(V7-8932)
- ZOS IBMMQ – 使用选项 SET_ALL_CONTEXT 增强对 PUT 命令的支持 (V7-9544)

改进

- 使用 $ 字符处理 gdg 文件名 (V7-9066)
- 当上一个 SQL 语句成功时，SQL 诊断将 1 作为 NUMBER 子句返回。(V7-9410)
- 为长度不为空的字段设置大纲 (V7-7536)
- Support 支持内置 PL1 图形功能 (V7-9245)
- MQ – 添加对 MQGMO 字段设置版本的支持 (V7-9500)
- JMS MQ GET – 消息返回的 dataLength 改进 (V7-9502)
- 在 ROWSET 上下文中使用提取的项目数设置 sqlerrd(3)。(V7-9371)

现代化工具版本 3.5.0

主题

- zOS
- AS400
- 横向功能

zOS

新特征

- ZOS PLI – 支持二进制表达式赋值中使用星号索引 (V7-9178)
- JCL to BatchScript -“//” 表示任务执行结束 (V7-9304)
- ZOS PLI – 增强对浮动字符和数字签名编辑类型的支持 (V7-8982)
- COBOL – 支持内置 SUM 函数 (V7-9367)
- JCL – 可选，在空语句之后注释死码 (//) (V7-9202)
- JCL – 支持在条件语句中使用运算符“|”(V7-9499)
- PL/I-在预处理步骤注释预编译指令以防止解析异常 (V7-9507)

改进

- 使用分隔符处理流定义 (V7-9615)
- 改进 LISTCAT 导出处理能力。(V7-9201)
- PL/I-增强了对隐式 “空” 参数的支持 (V7-9204)

AS400

新特征

- 支持 DDS 关键词 CONCAT (V7-9439)
- 重构为 DSPF 关键字生成的 Java 代码。(V7-7700)
- 支持在数据结构定义中的字段上使用不同的关键字 (V7-9029)

改进

- 改进逻辑关系的解析 AND/OR (V7-9352)
- COBOL 改进 vo 和 dsEntity 之间的映射 (V7-9449)
- 如果聚焦数字输入，则显示空值 (V7-9374)
- SQL 声明游标中的局部变量 (V7-9456)
- 空的 DS 的作用域问题 (V7-9466)
- 在解析之前截断 col 80 之后的行 (V7-9632)

- 改进定义规范中关键字（DIM、LIKE 等）中字段引用和内置调用的处理 (V7-9358)
- 支持 SQL 注释 (--) (V7-9632)
- FullFree 正在解析，键入 Date/Time/Timestamp (V7-9542)
- 在 FullFree 解析中包括 SQLCA (V7-9333)
- 改进对控制等级的支持。(V7-9610)
- 处理 DS 与 *BLANKS 的比较 (V7-9668)
- 改进对 DDS 中多个指示器的支持 (V7-9318)
- 改进对多个 DSPF 程序的支持（V7-9657）
- 使用 LIKE（类似数据结构和数组中类似数据结构的情况）改进字段处理 (V7-9213)
- Free RPG，处理文本字符串的延续 (V7-9686)
- 改进对程序结束记录的支持 (V7-9452)
- 支持 CALL 语句中的 LINKINGE 短语。(V7-9685)
- CASXX 操作代码（不带 CASXX 组的 CASBB）(V7-9357)
- 改进 FullFree角色扮演游戏解析 (V7-9457)
- 内置 %LEN 不支持 DS 作为参数 (V7-9267)
- 改进因子 2 为 *ALL'X...' 时的 MOVEA (V7-9228)
- 支持使用 RENAME 字段进行分配 (V7-9385)

横向功能

新特征

- SQL Migrator 工具：在 ebcdic 加载步骤中为可变记录长度添加 OID 选项。(V7-9380)
- SQL Migrator 工具：在 OID 选项上支持 Java 11 (V7-9599)

改进

- 改进对嵌套数组的支持 (V7-9595)
- 将 Â¬ 字符替换为 !（如果原始编码支持 Â¬）。(V7-9465)
- JCL – 支持传递正常终止，以便在作业步骤之间共享数据集 (V7-9504)
- 处理 VARCHAR 和可为空的数据库列类型时，对 ORACLE 上的列定义应用 ON NULL。(V7-9681)
- 提高 Spring 注入合规性 (V7-9635)

AWS Blu Age 安全漏洞

常见漏洞和风险（CVE）是公开已知网络安全漏洞的参考列表。每个条目均包含标识号、描述和至少一个公共参考。

我们建议您始终升级到最新的 AWS Blu Age 版本，以防范已知漏洞。Amazon Inspector 会持续进行安全扫描，并在 NIST 中根据发现的严重性级别对发现进行分类。

以下列表详细说明了每个可用次要版本中由于使用依赖项而导致的 CVEs 修复内容：

版本	CVE
5.1.0	CVE-2025-55754、CVE-2025-55752、CVE-2025-61795、CVE-2020-36843、CVE-2025-11226、CVE-2025-41254、CVE-2025-59052、CVE-2025-4949、CVE-2024-8184、CVE-2024-6763、C20VE-2024-6763、C20VE-2024-6763 25-58057，CVE-2025-58056
4.10.0	CVE-2025-55163、CVE-2025-48924、CVE-2025-53864、CVE-2025-52999、CVE-2025-41234、CVE-2025-49125、CVE-2025-48988、CVE-2025-5889
4.9.0	CVE-2025-49146、CVE-2025-22871、CVE-2022-41404、CVE-2025-22233、CVE-2025-46701、CVE-2025-22235、CVE-2025-46392、CVE-2025-27820、CVE-2025-24813
4.8.0	CVE-2025-22228
4.7.0	CVE-2025-25193、CVE-2025-24970、CVE-2024-47535、CVE-2024-38820、CVE-2024-38809、CVE-2024-22262、CVE-2022-45868
4.6.0	CVE-2024-12801、CVE-2024-12798、CVE-2024-50379、CVE-2024-56337

版本	CVE
4.5.0	CVE-2024-47535、CVE-2024-52316、CVE-2024-47535、CVE-2024-38827
4.4.0	CVE-2024-38820、CVE-2024-38821、CVE-2024-38809、CVE-2024-38816、CVE-2024-47554、CVE-2024-6484、CVE-2024-6485、CVE-2024-6485
4.3.0	CVE-2024-43788、CVE-2022-25898、CVE-2021-30246、CVE-2024-21484、CVE-2024-34750
4.2.0	CVE-2020-11023、CVE-2023-26364、CVE-2019-11358、CVE-2020-11022、CVE-2021-23358、CVE-2017-18214、CVE-2022-24785、CVE-2022-31129、CVE-2023-48631
4.1.0	CVE-2024-29025、CVE-2024-23080、CVE-2024-22262、CVE-2024-30171、CVE-2024-29857、CVE-2024-30172
4.0.0	CVE-2016-1000027、CVE-2022-1471、CVE-2024-1597、CVE-2024-22243、CVE-2024-22233、CVE-2024-22234、CVE-2024-22259、CVE-2024-22257、CVE-2024-29131、CVE-2024-29133

Note

有关先前版本中 CVEs 已修复的详细信息，请联系您的 AWS Blu Age 交付经理

AWS Blu Age 的升级说明

本页包含升级 AWS Blu Age 版本的说明。

通用升级

在大多数情况下，在升级 AWS Blu Age Runtime 版本时，应将先前版本的工件（WARs配置文件、脚本等）替换为新版本中提供的工件（配置文件、脚本等），然后重新启动应用程序。升级后，请务必对经过现代化改造的应用程序进行全面的回归测试。您也可以联系您的 AWS Blu Age 交付经理，获取适用于您的应用程序的具体说明。

某些升级可能需要进行额外的配置来确保兼容性。在这种情况下，请按照该特定升级的说明进行操作。

从 3.10.0 迁移到 4.0.0

4.0.0 中的主要变化是从 Spring Boot 2.7 迁移到 Spring Boot 3.2，以及从 Tomcat 9 迁移到 Tomcat 10。

代码更改

本节列出了使现代化代码与 AWS Blu Age Runtime 4.0.0 兼容所需的更改。如果您决定在 Blu Insights（转换中心）上使用 4.0.0 版本发布新一代应用程序，则可以跳过本节。

POM 更改

Group	ArtifactId	更改
org.slf4j	slf4j-api	删除（是一种传递依赖）
org.yaml	snakeyaml	删除（是一种传递依赖）
org.springframework.boot	spring-boot-starter-web	-将 spring.boot.version 升级到 3.2.4-移除 log4 4j 的排除项 j-to-slf
org.springframework.boot	spring-boot-starter-jta-atomikos	改为 com.atomikos：3-starter：6.0.0 transactions-spring-boot
org.apache.commons	commons-dbcp2	升级到 2.10.0
org.postgresql	postgreql	升级到 42.7.2
com.microsoft.sqlserver	mssql-jdbc	升级到 12.4.2.jre11

Group	ArtifactId	更改
com.oracle.database.jdbc	ojdbc8	改为 ojdbc11 版本 23.3.0.23 .09

从 Javax 迁移到 Jakarta

tomcat 升级附带从 Javax Java 程序包迁移到 Jakarta。请务必将导入相应地从 javax.* 更新到 jakarta.*。

Javax 程序包中引用的几乎所有旧类都可以在 Jakarta 中找到。已知的例外情况是 `javax.sql` 和 `javax.xml` 程序包，它们将仍然保持不变。

Atomikos 更改

由于上文提到的依赖变更，必须将对 `org.springframework.boot.jta.atomikos.AtomikosDataSourceBean` 的引用更改为 `com.atomikos.spring.AtomikosDataSourceBean`。

删除 PostgreSQL 方言

已删除自定义类 `PostgreSQLDialect.java`。主启动器中对该类的引用也必须删除。

部署（AWS Blu Age 运行时）

Tomcat

此版本与 Tomcat `10.1.17` 兼容。需要将 Tomcat 服务器升级到此版本才能运行Blu Age运行`4.0.0`时。请确保移植旧的配置更改（特别是 Catalina 属性）。

共享依赖项

运行时共享文件夹包含 up-to-date依赖关系。

额外的依赖项

如果您使用了额外的依赖项（未包含在运行时中），则可能需要对其进行更新。extra 文件夹中的自述文件列出了支持的版本。

AWS 蓝光时代生命周期

本节定义了 AWS Blu Age Runtime 主要版本的生命周期结束 (EOL) 日期。这使您能够规划版本升级，以便及时了解最新的维护和特征。要升级版本，请参阅 the section called “升级 AWS 蓝光时代”。

建议您每 3 个月检查一次新版本，并经常升级到最新版本。每次升级时，您都应在生产或关键部署之前对经过现代化改造的应用程序进行非回归测试。

> **Note**
>
> 生命周期终止日期可能会因重要安全补救措施的采用而发生变化。有关更多详细信息，请参阅组件生命周期。

AWS Blu Age Runtime 生命尽头 (EOL)

下表汇总了每个主要版本的 EOL。

主要版本	生命周期终止日期
第 5 版	不适用
版本 4	2026 年 2 月 18 日
版本 3	2024 年 7 月 8 日

> **Note**
>
> 主要版本 5 的停产日期将与下一个主要版本的可用性保持一致。

要了解次要版本的支持模式，请参阅 组件生命周期。

AWS 蓝光时代运行时概念

了解 AWS Blu Age Runtime 的基本概念可以帮助您了解如何通过自动重构实现应用程序的现代化。

主题

AWS Blu Age 运行时高级架构

作为将传统程序现代化为 Java 的 AWS Blu Age 解决方案的一部分， AWS Blu Age Runtime 通过提供遗留结构和程序代码组织标准化的库，为现代化应用程序提供了统一的、基于 REST 的入口点，并为此类应用程序提供了执行框架。

这种现代化的应用程序是 AWS Blu Age 自动重构过程的结果，该过程用于将大型机和中端程序（在以下文档中称为 “旧版”）现代化为基于 Web 的架构。

AWS Blu Age Runtime 的目标是再现遗留程序的行为（同功能性）、性能（在程序执行时间和资源消耗方面），以及Java开发人员易于维护现代化程序，尽管使用熟悉的环境和习语，例如tomcat、Spring、getters/setters，很流利。 APIs

主题

AWS 蓝光时代运行时组件

AWS Blu Age 运行时环境由两种组件组成：

- 一组 java 库（jar 文件），通常称为“共享文件夹”，提供遗留结构和语句。
- 一组 Web 应用程序（war 文件），其中包含基于 Spring 的 Web 应用程序，为现代化程序提供一组通用的框架和服务。

以下部分详细介绍了这两个组件的作用。

AWS 蓝光时代图书馆

AWS Blu Age 库是一组 jar 文件，存储在添加到标准 tomcat 类路径的`shared/`子文件夹中，以便所有现代化的 Java 程序都可以使用它们。这类库旨在提供在遗留开发环境中常见但在 Java 编程环境中非原生且不容易实现的特征。这些功能以 Java 开发人员尽可能熟悉的方式公开（getters/setter、基于类、流利）。 APIs一个重要的例子是 Data Simplifier 库，它为 Java 程序提供了传统的内存布局和操作结构（在 COBOL PL1 或 RPG 语言中遇到）。这些 jar 是遗留程序生成的现代化 Java 代码的核心依赖项。有关数据简化器的更多信息，请参阅Bl AWS u Age 中的数据简化器是什么。

Web 应用程序

Web 应用程序档案 (WARs) 是将代码和应用程序部署到 tomcat 应用程序服务器的标准方法。作为 AWS Blu Age 运行时的一部分提供的执行框架旨在提供一组执行框架，再现传统环境和事务监视器（JCL batch、CICS、IMS...）以及相关的必需服务。

最重要的是 `gapwalk-application`（通常缩写为“Gapwalk”），它提供了一组基于 REST 的统一入口点来触发和控制事务、程序和批处理执行。有关更多信息，请参阅 AWS 蓝光时代运行时间 APIs。

此 Web 应用程序分配 Java 执行线程和资源，以便在现代化程序所面向的环境中运行这些程序。以下部分通过举例详细介绍了此类重现环境。

其他 Web 应用程序向执行环境（更确切地说，向下文描述的“程序注册表”）中添加了模拟遗留程序可用的和可从遗留程序调用的程序的程序。其中重要的两类为：

- 操作系统提供的程序仿真：JCL 驱动的批处理特别需要能够调用各种文件和数据库操作程序作为其标准环境的一部分。示例包括 `SORT/DFSORT` 或 `IDCAMS`。为此，提供的 Java 程序可以重现此类行为并且可被调用（使用与遗留程序相同的方式）。
- “驱动程序”，由执行框架或中间件作为入口点提供的专用程序。例如，在 IMS 环境中执行的 COBOL 程序依赖 `CBLTDLI` 来访问与 IMS 相关的服务（IMS 数据库、通过 MFS 的用户对话等）。

程序注册表

为了参与和利用这些结构、框架和服务，从遗留程序现代化得到的 Java 程序遵循AWS 现代化应用程序的 Blu Age 结构中所述的特定结构。启动时， AWS Blu Age Runtime 会将所有此类程序收集到通用的 “程序注册表” 中，以便之后可以调用（并相互调用）。程序注册表提供了松耦合和分解的可能性（因为相互调用的程序不必同时进行现代化）。

执行环境

经常遇到的遗留环境和编排如下：

- JCL 驱动的批处理一旦现代化为 Java 程序和 Groovy 脚本，即可以同步（阻塞）或异步（分离）的方式启动。在后一种情况下，可以通过 REST 端点监控其执行情况。
- AWS Blu Age 子系统通过以下方式提供类似于 CICS 的执行环境：
  - 一个入口点，用于启动 CICS 事务和运行相关程序，同时遵循 CICS 的“运行级别”编排；
  - 资源定义的外部存储；
  - 一组同质的 Java 流畅 APIs 复制语句`EXEC CICS`，
  - 一组可重现 CICS 服务的可插拔类，例如临时存储队列、临时数据队列或文件访问（通常有多种实现可用，例如适用于 Apache Flink 的亚马逊托管服务、Amazon Simple Queue Service 或适用于 TD 队列的 RabbitMQ）；
  - 对于面向用户的应用程序，BMS 屏幕描述格式已现代化为 Angular Web 应用程序，并支持相应的“伪对话”对话框。
- 同样，另一个子系统提供基于 IMS 消息的编排，并支持采用 MFS 格式的 UI 屏幕现代化。
- 此外，第三个子系统允许在类似 iSeries 的环境中执行程序，包括对 DSPF（显示文件）指定的屏幕进行现代化。

所有这些环境都建立在常见的操作系统级服务之上，例如：

- 仿真遗留内存分配和布局（数据简化器）；
- 基于 Java 线程重现 COBOL“运行单元”执行和参数传递机制（`CALL` 语句）；
- 模拟平面连接 VSAM（通过 Blusam 库集）和 GDG 数据集组织，
- 访问数据存储，例如 RDBMS（`EXEC SQL` 语句）。

无状态和会话处理

AWS Blu Age Runtime 的一个重要功能是在执行现代化程序时启用高可用性 (HA) 和水平可扩展性场景。

该特征基于无状态，其重要的示例是 HTTP 会话处理。

会话处理

Tomcat 是基于 Web 的，实现这一点的一个重要机制是 HTTP 会话处理（由 tomcat 和 Spring 提供）和无状态设计。这类无状态设计基于以下几点：

- 用户通过 HTTPS 进行连接；
- 应用程序服务器部署在负载均衡器后面；

- 当用户首次连接到应用程序时，将对其进行身份验证，并且应用程序服务器会创建一个标识符（通常在 Cookie 中）；
- 此标识符将用作将用户上下文保存到外部缓存（数据存储）和从外部缓存（数据存储）检索用户上下文的密钥。

Cookie 管理由 AWS Blu Age 框架和底层 tomcat 服务器自动完成，这对用户是透明的。用户的互联网浏览器将自动对此进行管理。

Gapwalk Web 应用程序可以将会话状态（上下文）存储在各种数据存储中：

- 亚马逊 ElastiCache （Redis OSS）
- Redis 集群
- 内存映射（仅适用于开发和独立环境，不适用于 HA）。

高可用性与无状态

更笼统地说， AWS Blu Age 框架的设计原则是无状态性：重现遗留程序行为所需的大多数非临时状态都不存储在应用程序服务器中，而是通过外部常见的 “单一事实来源” 共享。

此类状态的示例有 CICS 的临时存储队列或资源定义，而这些状态的典型外部存储是与 Redis 兼容的服务器或关系数据库。

这种设计与负载均衡和共享会话相结合，使大多数面向用户的对话（OLTP，“在线事务处理”）都可以在多个“节点”（此处为 tomcat 实例）之间分发。

实际上，用户可以在任何服务器上执行事务，无需关注下一个事务调用是否在另一台服务器上执行。然后，当生成新服务器时（由于自动扩缩或者为了替换运行不正常的服务器），我们可以保证任何可访问且运行良好的服务器都能按预期运行事务，并获得正确的结果（预期的返回值、数据库中的预期数据更改等）。

AWS 现代化应用程序的 Blu Age 结构

本文档详细介绍了现代化应用程序的结构（使用 AWS 大型机现代化重构工具），以便开发人员可以完成各种任务，例如：

- 顺畅地导航到应用程序。
- 开发可从现代化应用程序中调用的自定义程序。
- 安全地重构现代化应用程序。

我们假设您已经掌握了以下方面的基础知识：

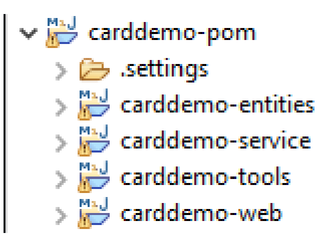
- 遗留的常见编码概念，例如记录、数据集及其对记录（索引、顺序）、VSAM、运行单元、jcl 脚本、CICS 概念等的访问模式。
- 使用 Spring 框架进行 java 编码。
- 为方便阅读，我们在整篇文档中使用了`short class names`。有关更多信息，请参阅AWS Blu Age 完全限定名称映射检索 AWS Blu Age 运行时元素的相应完全限定名称和第三方完全限定名称映射检索第三方元素的相应完全限定名称。
- 所有工件和样本均取自示例 COBOL/CICS CardDemo 应用程序的现代化过程输出。

主题

构件组织

AWS Blu Age 现代化应用程序打包为 java Web 应用程序 (.war)，您可以将其部署在 JEE 服务器上。通常，服务器是一个嵌入了 AWS Blu Age Runtime 的 Tomcat 实例，该运行时目前建立在 Springboot 和 Angular（用于用户界面部分）框架之上。

war 聚合了几个组件构件 (.jar)。每个 jar 都是专用 java 项目的编译（使用 maven 工具）结果，该项目的元素是在现代化过程产生的。



基本组织依赖以下结构：

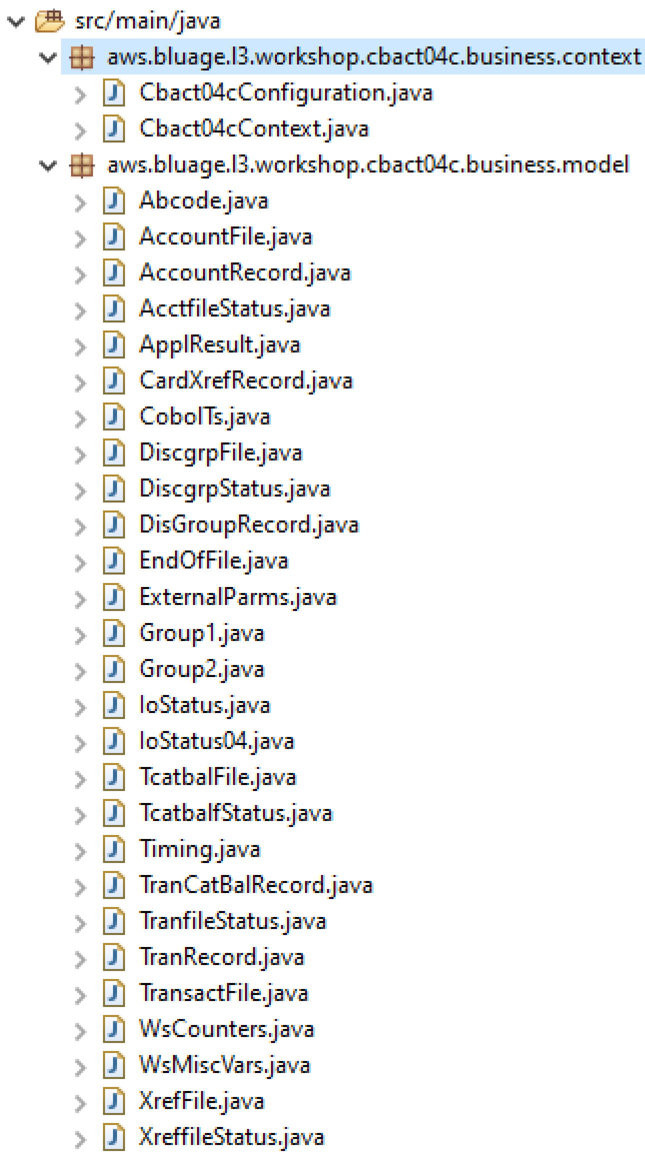
- 实体项目：包含业务模型和上下文元素。项目名称通常以“-entities”结尾。通常，对于给定的遗留 COBOL 程序，这与 I/O 部分（数据集）和数据部门的现代化相对应。您可以有多个实体项目。
- 服务项目：包含遗留业务逻辑现代化元素。通常是 COBOL 程序的过程部分。您可以有多个服务项目。

- 实用程序项目：包含其他项目使用的共享常用工具和实用工具。
- Web 项目：包含 UI 相关元素的现代化（如果适用）。不用于仅限批处理的现代化项目。这些 UI 元素可能来自 CICS BMS 映射、IMS MFS 组件和其他大型机 UI 源。您可以有多个 Web 项目。

实体项目内容

> **Note**
> 以下描述仅适用于 COBOL 和 PL/I 现代化输出。RPG 现代化输出基于不同的布局。

在进行任何重构之前，实体项目中的包组织与现代化程序相关联。您可以通过几种不同的方法来实现这一点。首选方法是使用重构工具箱，该工具箱在您触发代码生成机制之前运行。这是一项高级操作，BluAge 培训中将对此进行了解释。有关更多信息，请参阅重构研讨会。使用这种方法时，您可以保留后续重新生成 Java 代码的能力，例如可以从将来的进一步改进中受益。另一种方法是使用您希望应用的任何 java 重构方法直接在生成的源代码上进行常规的 java 重构，但需自行承担风险。



项目相关类

每个现代化程序都与两个包相关：business.context 包和 business.model 包。

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

  business.context 子包包含两个类，即一个配置类和一个上下文类。

  - 程序的一个配置类，其中包含给定程序的特定配置详细信息，例如用于表示基于字符的数据元素的字符集、用于填充数据结构元素的默认字节值等。类名称以“Configuration”结尾。该类标有 `@org.springframework.context.annotation.Configuration` 注释，并且包含一个必须返回正确设置的 `Configuration` 对象的方法。

Cbact04cConfiguration.java

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

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

/**
 * Creates Datasimplifier configuration for the Cbact04cContext context.
 */
@org.springframework.context.annotation.Configuration
@Lazy
public class Cbact04cConfiguration {

    @Bean(name = "Cbact04cContextConfiguration")
    public Configuration configuration() {
        return new ConfigurationBuilder()
                .encoding(Charset.forName("CP1047"))
                .humanReadableEncoding(Charset.forName("ISO-8859-15"))
                .initDefaultByte(0)
                .build();
    }

}
```

- 一个上下文类，用作程序服务类（见下文）与模型子包（见下文）中的数据结构 (Record) 和数据集 (File) 之间的桥梁。该类名称以“Context”结尾并且是 RuntimeContext 类的子类。

Cbact04cContext.java

```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;
```

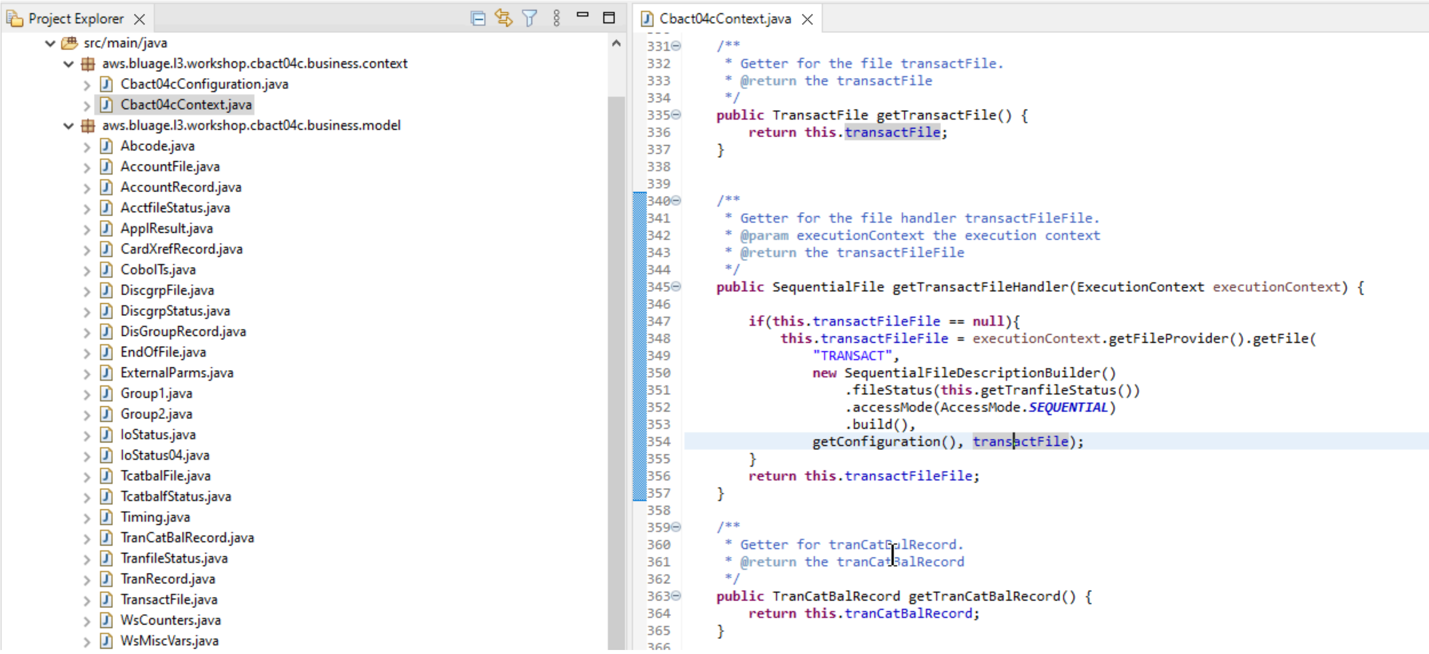
- *base package*.*program*.business.model

  模型子包包含给定程序可以使用的所有数据结构。例如，任何 01 级 COBOL 数据结构都对应于模型子包中的一个类（较低级别的数据结构是其所属 01 级结构的属性）。有关如何对 01 数据结构进行现代化的更多信息，请参阅BI AWS u Age 中的数据简化器是什么。

DiscgrpFile.java

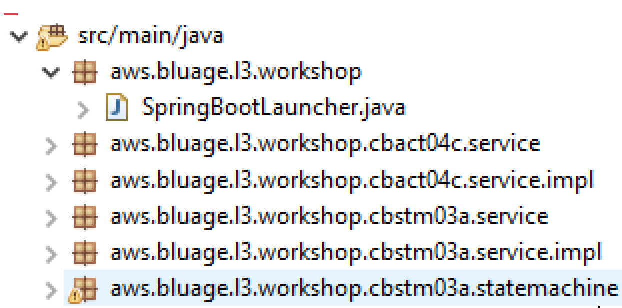
```java
package aws.bluage.l3.workshop.cbact04c.business.model;

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));
```

所有类都扩展了 RecordEntity 类，该类表示对业务记录表示的访问权限。有些记录有特殊用途，因为它们绑定到 File。a Record 和 a 之间的绑定File是在创建文件对象时在上下文类中找到的相应的 * FileHandler 方法中进行的。例如，以下清单显示了如何绑定到 TransactFileRecord（来自模型子包）。 TransactfileFile File



服务项目内容

每个服务项目都附带一个专用的 Springboot 应用程序，该应用程序用作架构的主干。具体是通过名为 `SpringBootLauncher` 的类实现的，该类位于服务 java sources 的基础包中：



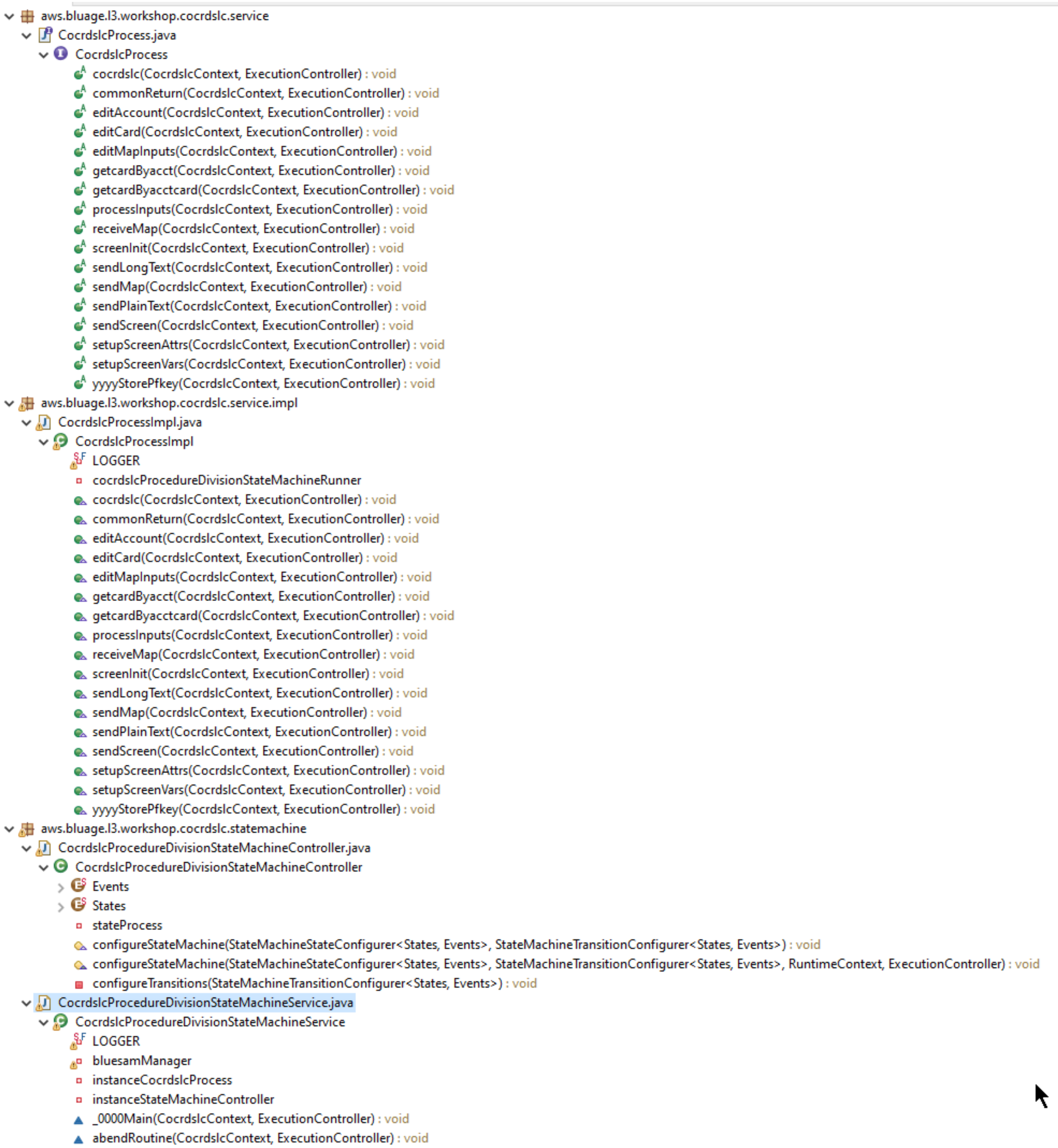
该类主要用于：

- 为程序类和托管资源（数据源/事务管理器/数据集映射/等...）之间建立联系。
- 为程序提供 `ConfigurableApplicationContext`。
- 发现所有标记为 spring 组件 (`@Component`) 的类。
- 确保程序在 `ProgramRegistry` 中正确注册。请参阅用于进行此注册的 initialize 方法。

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

程序相关构件

在未进行事先重构的情况下，业务逻辑现代化输出将按每个遗留程序两个或三个包进行组织：



最详尽的情况下将提供三个包：

- *base package.program*.service：包含一个名为 ProgramProcess 的接口，该接口具有处理业务逻辑的业务方法，保留了遗留的执行控制流。

- *base package*.*program*.service.impl: 包含一个名为 Prog ra ProcessImpl m 的类，它是前面描述的 Process 接口的实现。这是依赖 AWS Blu Age 框架将遗留语句 “翻译” 为 java 语句的地方：

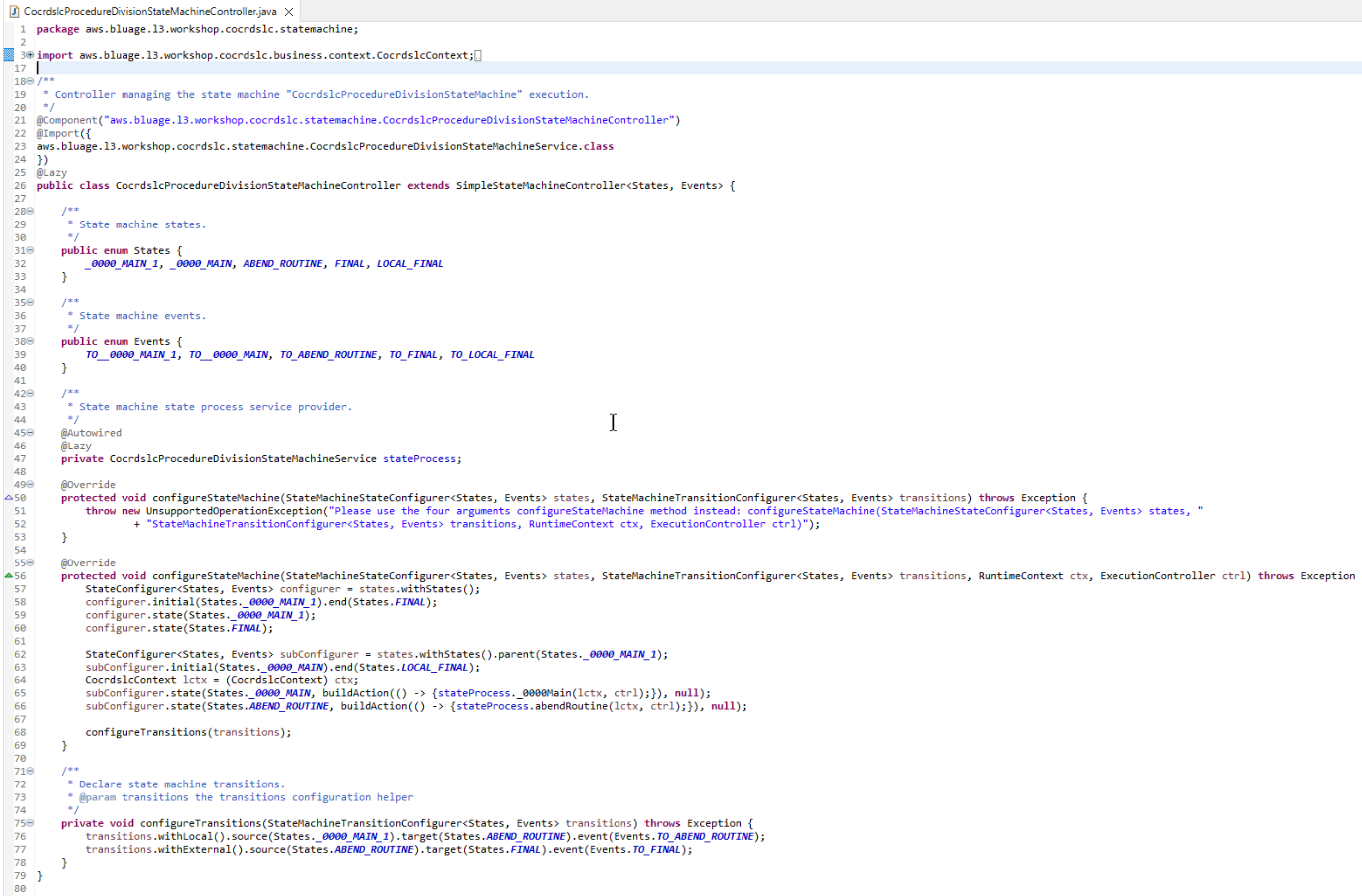
CocrdslcProcessImpl.java

```
    /**
     * 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：该包有时可能不存在。当传统控制流的现代化必须使用状态机方法（即使用 [Spring StateMachine 框架](https://spring.io/projects/spring-statemachine)）来正确覆盖遗留的执行流程时，这是必需的。

  在这种情况下，状态机子包包含两个类：

  - *Program*ProcedureDivisionStateMachineController：该类扩展实现 StateMachineController（定义控制状态机执行所需的操作）和 StateMachineRunner（定义运行状态机所需的操作）接口的类，用于驱动 Spring 状态机机制；例如，示例用例中的 SimpleStateMachineController。



CocrdslcProcedureDivisionStateMachineController.java

```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);
    }
}
```

状态机控制器定义了可能的不同状态以及状态之间的转换，可重现给定程序的遗留执行控制流。

在构建状态机时，控制器会引用状态机包的关联服务类中定义的方法，如下所述：

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

- *Program*ProcedureDivisionStateMachineService：该服务类代表一些业务逻辑，这些逻辑需要与状态机控制器创建的状态机绑定，如前所述。

  该类方法中的代码使用状态机控制器中定义的事件：

CocrdslcProcedureDivisionStateMachineService.java

```java
    /**
     * State process operation _0000Main.
     *
     * @param ctx
     * @param ctrl
     */
    void _0000Main(CocrdslcContext ctx, ExecutionController ctrl) {
        ctx.getDfheiblk().bind(ArgUtils.get(ctx, 0));
        ctx.getDfhcommarea().bind(ArgUtils.get(ctx, 1));

        /*
        ***************************************************************
        Program:     COCRDSLC.CBL                                     *
        Layer:       Business logic                                   *
        Function:    Accept and process credit card detail request    *
        ****************************************************************
        Copyright Amazon.com, Inc. or its affiliates.
        All Rights Reserved.
        Licensed under the Apache License, Version 2.0 (the \"License\").
        You may not use this file except in compliance with the License.
        You may obtain a copy of the License at
        http://www.apache.org/licenses/LICENSE-2.0
        Unless required by applicable law or agreed to in writing,
        software distributed under the License is distributed on an
        \"AS IS\" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND,
        either express or implied. See the License for the specific
        language governing permissions and limitations under the License
        ****************************************************************
        Ver: CardDemo_v1.0-15-g27d6c6f-68 Date: 2022-07-19 23:16:00 CDT */
        instanceStateMachineController.registerSignalHandler(Events.TO_ABEND_ROUTINE, "!ABEND");
        HandleAbendBuilder.newInstance(ctx.getDfheiblk(), ctx).execute().handleException();
        ctx.getCcWorkAreas().getCcWorkAreaReference().getField().initialize();
        ctx.getWsMiscStorage().getField().initialize();
        DataUtils.initialize(ctx.getWsCommarea().getWsCommareaReference());

```

CocrdslcProcedureDivisionStateMachineService.java

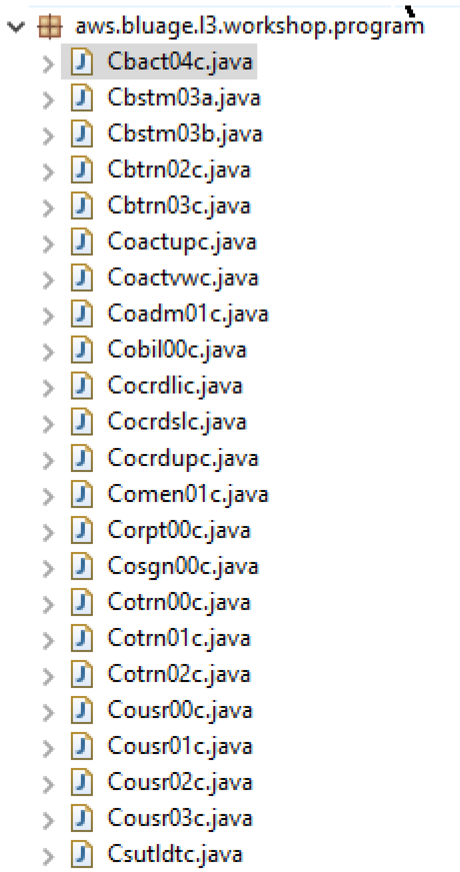
```java
     *
     * @param ctx
     * @param ctrl
     */
    void abendRoutine(CocrdslcContext ctx, ExecutionController ctrl) {
        if (DataUtils.isLowValue(ctx.getAbendData().getAbendMsgReference())) {
            ctx.getAbendData().setAbendMsg("UNEXPECTED ABEND OCCURRED.");
        }
        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);

    }
}

```

状态机服务还会调用前面所述的过程服务实现：

CocrdslcProcedureDivisionStateMachineService.java

```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);
```

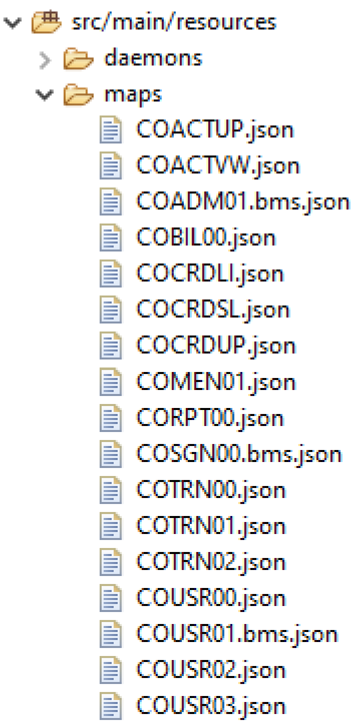
除此之外，名为 *base package*.program 的包也发挥重要作用，因为该包为每个程序收集一个类，该类将作为程序的入口点（稍后会详细介绍）。每个类都实现 Program 接口，即程序入口点的标记。



其他构件

- BMS MAPs 同伴

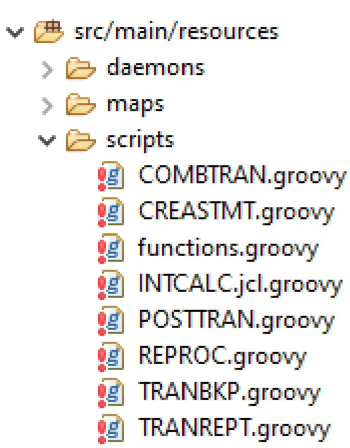
  除了与程序相关的构件外，服务项目还可以包含用于各种用途的其他构件。对于 CICS 在线应用程序的现代化，现代化过程会生成一个 json 文件并放入/文件夹的地图src/main/resources文件夹中：



Blu Age运行时使用这些 json 文件将 SEND MAP 语句使用的记录与屏幕字段绑定。

- Groovy 脚本

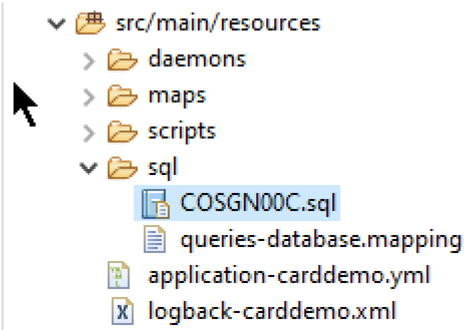
  如果旧版应用程序有 JCL 脚本，则这些脚本已现代化为 groovy 脚本，存储在/src/main/resources/scripts文件夹中（稍后会详细介绍该特定位置）：



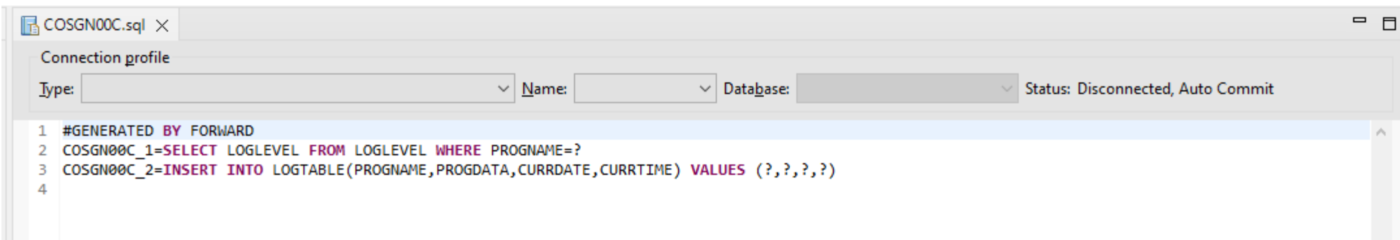
  这些脚本用于启动批处理作业（专用、非交互式、CPU 密集型数据处理工作负载）。

- SQL 文件

  如果遗留应用程序使用 SQL 查询，则相应的现代化 SQL 查询已收集在专用属性文件中，使用命名模式 program.sql，其中 program 是使用这些查询的程序的名称。



这些 sql 文件的内容是 (key=query) 条目的集合，其中每个查询都与一个唯一的键相关联，现代化程序使用该键来运行给定查询：

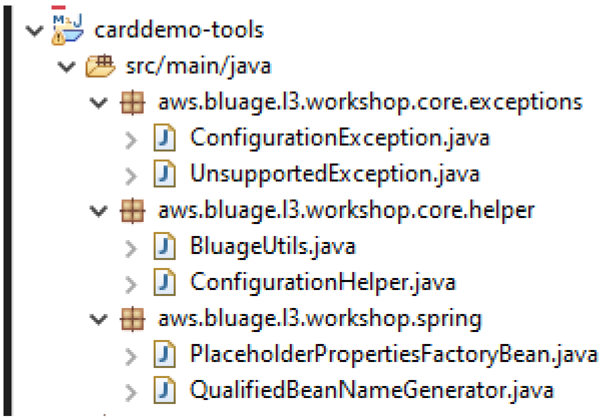


例如，COSGN00C 程序正在使用键 “COSGN00C_1”（sql 文件中的第一个条目）执行查询：

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

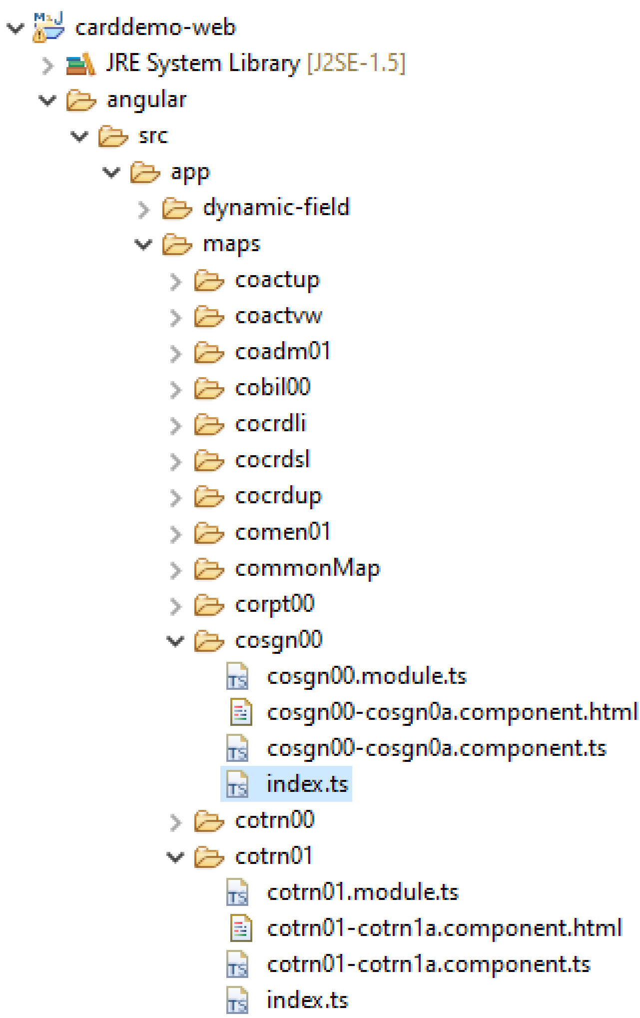
实体项目内容

名字以“-tools”结尾的实用程序项目包含一组技术实用工具，这些工具可用于所有其他项目。



Web 项目内容

Web 项目仅在对遗留 UI 元素进行现代化时才存在。用于构建现代化应用程序前端的现代 UI 元素基于 Angular 用于显示现代化构件的示例 COBOL/CICS 应用程序是在大型机上运行的应用程序。CICS 系统 MAPs 用来表示用户界面屏幕。每个映射相应的现代元素为一个 html 文件和多个 Typescript 文件：



Web 项目仅处理应用程序的前端方面。依赖于实用程序和实体项目的服务项目提供后端服务。前端和后端之间的链接是通过名为 Gapwalk-Application 的 Web 应用程序建立的，该应用程序是标准 AWS Blu Age 运行时发行版的一部分。

运行和调用程序

在遗留系统上，程序被编译为独立的可执行文件，可以通过 CALL 机制（例如 COBOL CALL 语句）进行自我调用，并根据需要传递参数。现代化应用程序提供了相同的功能，但使用了不同的方法，因为所涉及的构件的性质与遗留构件的性质不同。

在现代化系统上，程序入口点是实现 `Program` 接口的特定类，是 Spring 组件 (@Component)，位于服务项目中名为 `*base package*.program` 的包中。

程序注册

每次启动托管现代化应用程序的 Tomcat 服务器时，也会启动服务 Springboot 应用程序，从而触发程序注册。名为 `ProgramRegistry` 的专用注册表中填充了程序条目，每个程序都使用其标识符进行注册，每个已知的程序标识符对应一个条目，这意味着如果一个程序具有多个不同的已知标识符，则注册表包含的条目与标识符的数量相等。

给定程序的注册依赖于 getProgramIdentifiers () 方法返回的标识符集合：

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;
    }
```

在此示例中，该程序注册了一次，名为“CBACT04C”（查看 programIdentifiers 集合的内容）。Tomcat 日志显示每个程序的注册情况。程序注册仅取决于声明的程序标识符，而不取决于程序类名词本身（尽管通常程序标识符和程序类名称是一致的）。

同样的注册机制适用于各种实用程序 AWS Blu Age Web 应用程序带来的实用程序，这些应用程序是 AWS Blu Age 运行时发行版的一部分。例如， Gapwalk-Utility-PgmWeb应用程序提供了 z/OS 系统实用程序（IDCAMS、ICEGENER、SORT等）的等效功能，并且可以通过现代化的程序或脚本进行调用。在 Tomcat 启动时注册的所有可用实用程序都记录在 Tomcat 日志中。

脚本和进程守护程序注册

在 Tomcat 启动时，位于/ src/main/resources/scripts 文件夹层次结构中的 groovy 脚本也会发生类似的注册过程。遍历脚本文件夹层次结构后，所有发现的 groovy 脚本（特殊的 functions.groovy 保留脚本除外）都将注册到 `ScriptRegistry` 中，并使用其短名称（脚本文件名中位于第一个点字符之前的部分）作为密钥进行检索。

> **Note**
>
> - 如果多个脚本的文件名会导致生成相同的注册密钥，则仅注册最新的脚本，并覆盖该给定密钥之前的任何注册。
> - 考虑到上述注意事项，使用子文件夹时需引起注意，因为注册机制会使层次结构扁平化并可能导致意外覆盖。层次结构不计入注册过程：通常为/ scripts/A/myscript.groovy and / scripts/B/myscript.groovy will lead to /scripts/B/myscript.groovy overwriting /scripts/A/ myscript .groovy。

/src/main/resources/daemons文件夹中的 groovy 脚本的处理方式略有不同。它们仍被注册为常规脚本，但此外，它们仅在 Tomcat 启动时以异步方式启动一次。

在 `ScriptRegistry` 中注册脚本后，即可使用 Gapwalk-Application 公开的专用端点进行 REST 调用来启动这些脚本。有关更多信息，请参阅相应文档。

程序调用程序

每个程序都可以将另一个程序作为子程序进行调用，并向其传递参数。程序使用 `ExecutionController` 接口实现来进行调用（大多数情况下，这是一个 `ExecutionControllerImpl` 实例），并使用名为 `CallBuilder` 的 fluent API 机制来构建程序调用参数。

所有程序的方法都将 `RuntimeContext` 和 `ExecutionController` 作为方法参数，因此 `ExecutionController` 始终可用于调用其他程序。

例如，下图显示了 CBST03A 程序如何将 CBST03B 程序作为子程序调用，并向其传递参数：

Cbstm03aProcessImpl.java

```java
    /**
     * Process operation xreffileGetNext.
     *
     * ---------------------------------------------------------------*
     *
     * @param ctx
     * @param ctrl
     */
    @Override
    public void xreffileGetNext(final Cbstm03aContext ctx, final ExecutionController ctrl) {
        ctx.getWsM03bArea().setWsM03bDd("XREFFILE");
        ctx.getWsM03bArea().setM03bRead(true);
        DataUtils.setToZeroes(ctx.getWsM03bArea().getWsM03bRcReference());
        DataUtils.setToBlank(ctx.getWsM03bArea().getWsM03bFldtReference());
        ctrl.callSubProgram("CBSTM03B", CallBuilder.newInstance()
            .byReference(ctx.getWsM03bArea())
            .getArguments(), ctx);
        if (DataUtils.compare(ctx.getWsM03bArea().getWsM03bRcReference(), "00") == 0) {

            /*
            Do nothing */
        } else if (DataUtils.compare(ctx.getWsM03bArea().getWsM03bRcReference(), "10") == 0) {
            ctx.getMiscVariables().setEndOfFile("Y");
        } else {
            if (LOGGER.isInfoEnabled()) LOGGER.info("ERROR READING XREFFILE");
            if (LOGGER.isInfoEnabled()) LOGGER.info("{}{}" , "RETURN CODE: " , ctx.getWsM03bArea().getWsM03bRc());
            abendProgram(ctx, ctrl);
        }
        ctx.getCardXrefRecord().setBytes(ctx.getWsM03bArea().getWsM03bFldtReference().getBytes());
    }
```

- `ExecutionController.callSubProgram` 的第一个参数是要调用的程序的标识符（即用于程序注册的标识符之一，请参阅以上段落）。
- 第二个参数是在 `CallBuilder` 上构建的结果，是一个 `Record` 数组，对应于从调用方传递给被调用方的数据。
- 第三个也是最后一个参数是调用者 `RuntimeContext` 实例。

所有三个参数都是必需的，不能为空，但第二个参数可以是空数组。

被调用方仅在最初设计为能够处理传递的参数时才能处理这些参数。对于遗留的 COBOL 程序，这意味着要有一个 LINKAGE 部分和一个 USING 子句，以便过程部分使用 LINKAGE 元素。

例如，请参阅相应的 CBSTM03B.CBL COBOL 源文件：

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

```

       LINKAGE SECTION.
       01  LK-M03B-AREA.
           05  LK-M03B-DD            PIC X(08).
           05  LK-M03B-OPER          PIC X(01).
             88  M03B-OPEN         VALUE 'O'.
             88  M03B-CLOSE        VALUE 'C'.
             88  M03B-READ         VALUE 'R'.
             88  M03B-READ-K       VALUE 'K'.
             88  M03B-WRITE        VALUE 'W'.
             88  M03B-REWRITE      VALUE 'Z'.
           05  LK-M03B-RC            PIC X(02).
           05  LK-M03B-KEY           PIC X(25).
           05  LK-M03B-KEY-LN        PIC S9(4).
           05  LK-M03B-FLDT          PIC X(1000).

       PROCEDURE DIVISION USING LK-M03B-AREA.

```

因此，CBSTM03B 程序采用单个 `Record` 作为参数（大小为 1 的数组）。这是 `CallBuilder` 使用 byReference() 和 getArguments() 方法链接构建的内容。

`CallBuilder` fluent API 类有几种方法可用于填充要传递给被调用方的参数数组：

- asPointer (RecordAdaptable)：通过引用添加指针类型的参数。指针表示目标数据结构的地址。
- byReference (RecordAdaptable)：通过引用添加参数。调用方将看到被调用方执行的修改。
- byReference (RecordAdaptable)：先前方法的可变参数变体。
- byValue(Object)：添加一个按值转换为 `Record` 的参数。调用方将看到被调用方执行的修改。
- byValue (RecordAdaptable)：与前面的方法相同，但参数可以直接作为`RecordAdaptable`.
- byValueWith边界（Object、int、int）：添加一个参数，将其转换为 a`Record`，按值提取由给定边界定义的字节数组部分。

最后，getArguments 方法将收集所有添加的参数，并将其作为 `Record` 数组返回。

> **Note**
>
> 调用方需确保参数数组具有所需的大小，在内存布局与 linkage 元素的预期布局方面，项目顺序正确且兼容。

调用程序的脚本

从 groovy 脚本中调用注册程序需要使用用于实现 `MainProgramRunner` 接口的类实例。通常，获得这样的实例是通过 Spring 的 ApplicationContext 使用来实现的：

REPROC.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 = [:]
```

`MainProgramRunner` 接口可用后，使用 runProgram 方法调用程序并将目标程序的标识符作为参数传递：

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")
                })
        }
    }
}
```

在上述示例中，作业步骤调用 IDCAMS（文件处理实用程序），提供实际数据集定义与其逻辑标识符之间的映射。

在处理数据集时，遗留程序大多使用逻辑名称来标识数据集。当从脚本调用程序时，脚本必须将逻辑名称与实际的物理数据集进行映射。这些数据集可能位于文件系统中、Blusam 存储中，甚至可以由内联流、多个数据集的串联或生成 GDG 来定义。

使用该 withFileConfiguration 方法构建数据集的逻辑到物理映射，并将其提供给被调用的程序。

自行编写程序

自行编写程序供脚本或其他现代化程序调用是一项常见任务。通常，在现代化项目中，当可执行的遗留程序是使用现代化过程不支持的语言编写的，或者源代码丢失（是的，可能会发生这种情况），或者该程序是一个源代码不可用的实用程序时，您需要自行编写程序。

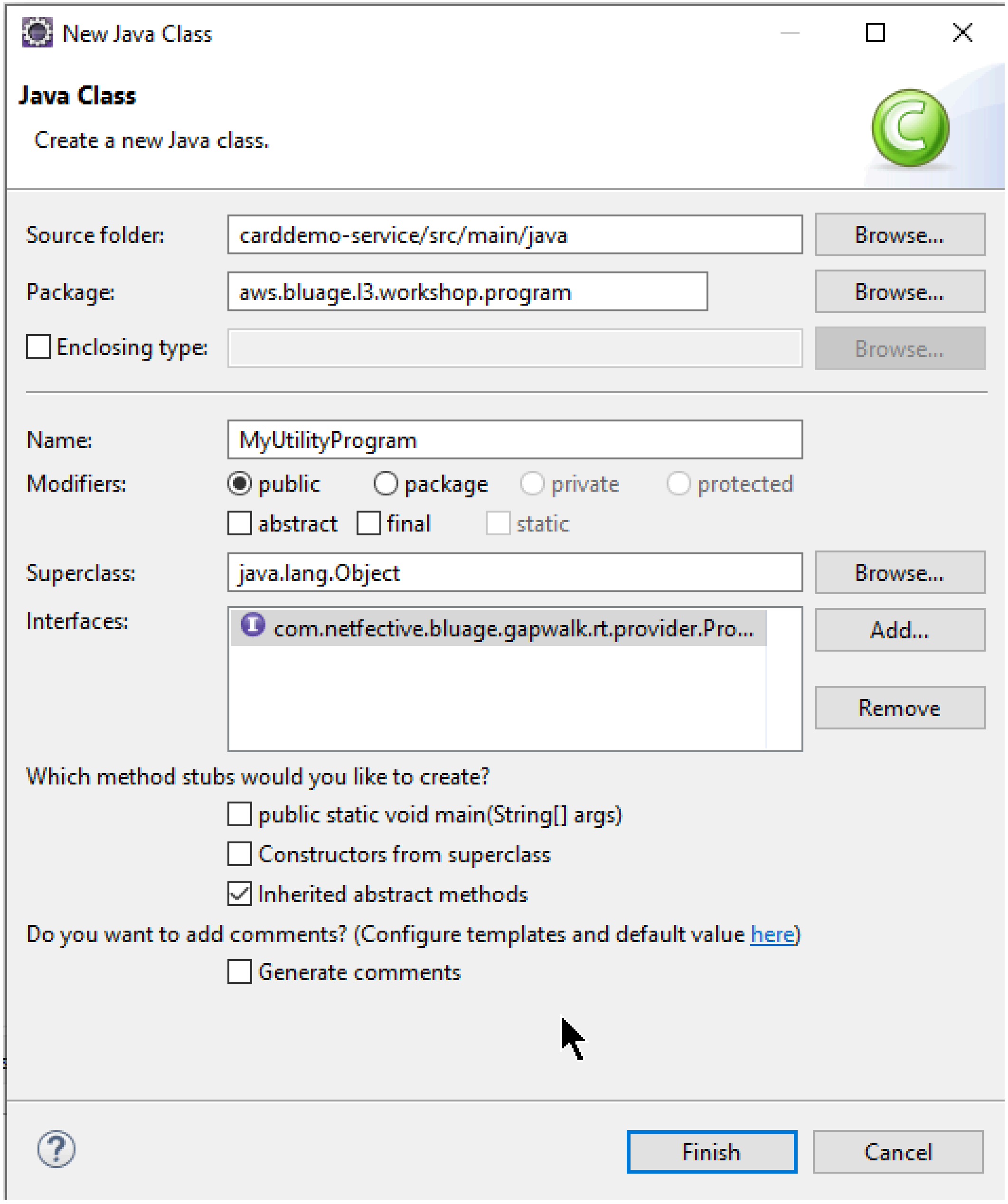
在这种情况下，您可能必须自己使用 java 编写缺失的程序（前提是您对程序的预期行为、程序参数的内存布局（如果有）等有足够的了解。）您的 java 程序必须符合本文档中描述的程序机制，以便其他程序和脚本可以运行它。

要确保该程序可用，您必须完成两个必需的步骤：

- 编写一个能正确实现 `Program` 接口的类，以便可以注册和调用该接口。
- 确保您的程序已正确注册，以便其他程序/脚本可以看到该程序。

编写程序实现

使用 IDE 创建用于实现 `Program` 接口的新 java 类：



下图显示了 Eclipse IDE，用于创建所有要实现的必需的方法：

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

    }

}
```

Spring 集成

首先，必须将该类声明为 Spring 组件。为该类添加 @Component 注释：

```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 {
```

接下来，正确实现所需的方法。在本示例中，我们将 MyUtilityProgram 添加到已包含所有现代化程序的包中，该位置允许程序使用现有的 Springboot 应用程序来提供 getSpringApplication 方法实现ConfigurableApplicationContext所需的内容：

```
public class MyUtilityProgram implements Program {

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

您可以为自己的程序选择不同的位置。例如，您可以将给定的程序放在另一个专门的服务项目中。确保给定的服务项目有自己的 Springboot 应用程序，这样就可以检索 ApplicationContext（应该是`ConfigurableApplicationContext`）。

为程序添加标识

要使程序能被其他程序和脚本调用，必须为其提供至少一个标识符，该标识符不得与系统中任何现有的其他已注册程序发生冲突。选择标识符可以根据要涵盖现有遗留程序替换的需求来确定；在这种情况下，您必须使用所需的标识符，例如在整个遗留程序中 CALL 次数中所出现的标识符。在遗留系统中，大多数程序标识符的长度为 8 个字符。

在程序中创建一组不可修改的标识符是实现此目的的一种方法。以下示例显示了如何选择“MYUTILPG”作为单一标识符：

```
@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;
    }
```

将程序与上下文关联

程序需要一个配套 `RuntimeContext` 实例。对于现代化程序， AWS Blu Age 会使用传统程序中的数据结构自动生成配套上下文。

如果您正在编写自己的程序，则还必须编写配套上下文。

请参阅项目相关类，即可以看到一个程序至少需要两个配套类：

- 配置类。
- 使用配置的上下文类。

如果实用程序使用任何额外的数据结构，则还应编写相应的数据结构并使其被上下文使用。

这些类应位于包层次结构中的包中，应用程序启动时会扫描该包层次结构，从而确保上下文组件和配置由 Spring 框架处理。

我们在实体项目中新创建的 *base package*`.myutilityprogram.business.context` 包中编写一个最小配置和上下文：

- aws.bluage.l3.workshop.csutldtc.business.model
  - FeedbackCode.java
  - LsDate.java
  - LsDateFormat.java
  - LsResult.java
  - OutputLillian.java
  - WsDateFormat.java
  - WsDateToTest.java
  - WsMessage.java
- aws.bluage.l3.workshop.myutilityprogram.business.context
  - MyUtilityProgramConfiguration.java
  - MyUtilityProgramContext.java

以下是配置内容，其使用的配置构建与附近其他（现代化）程序类似。您可能需要根据自己的特定需求对其进行自定义。

MyUtilityProgramConfiguration.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();
    }
}

```

注意：

- 一般命名约定是ProgramName配置。
- 必须使用 @org.springframework.context.annotation.Configuration 和 @Lazy 注释。
- bean 名称通常遵循ProgramNameContextConfiguration 惯例，但这不是强制性的。确保在整个项目中避免 Bean 名称冲突。
- 要实现的单个方法必须返回一个 `Configuration` 对象。使用 `ConfigurationBuilder` fluent API 来帮助您构建此方法。

相关的上下文：

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
    }

}
```

备注

- 上下文类应扩展现有的 `Context` 接口实现（`RuntimeContext` 或者 `JicsRuntimeContext`（即具有 JICS 特定项的增强 `RuntimeContext`））。
- 一般命名约定是ProgramName上下文。
- 必须将其声明为 Prototype 组件，并使用 @Lazy 注解。
- 构造函数引用关联的配置，使用 @Qualifier 注释来定位正确的配置类。
- 如果实用程序使用一些额外的数据结构，这些数据结构应：

- 已编写并添加到 *base package*.business.model 包中。
- 在上下文中引用。查看其他现有的上下文类，了解如何引用数据结构类并根据需要调整上下文方法（构造函数/清理/重置）。

专用上下文可用后，新程序即可使用该上下文：

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);
    }
```

注意:

- getContext 方法必须严格如图所示，使用对`ProgramContextStore`类的 getOrCreate 方法和自动连线 Spring `BeanFactory` 的委托。单个程序标识符用于在 `ProgramContextStore` 中存储程序上下文；该标识符被称为“程序主标识符”。
- 必须使用 `@Import` spring 注释来引用配套配置和上下文类。

实现业务逻辑

完成程序框架后，实现新实用程序的业务逻辑。

在程序的 `run` 方法中执行此操作。该方法会在程序被其他程序或脚本调用时执行。

祝您编码顺利！

处理程序注册

最后，请确保将新程序在 `ProgramRegistry` 中正确注册。如果您将新程序添加到已包含其他程序的包中，则无需执行任何其他操作。在应用程序启动时，新程序会被选中并与其所有邻居程序一并注册。

如果您为程序选择了其他位置，则必须确保该程序在 Tomcat 启动时能够正确注册。有关如何执行此操作的一些灵感，请查看服务项目中生成 SpringbootLauncher 类的初始化方法（请参阅服务项目内容）。

查看 Tomcat 启动日志。每个程序注册都会被记录在日志中。如果您的程序成功注册，可找到匹配的日志条目。

确定程序已正确注册后，即可开始迭代业务逻辑编码。

完全限定名称映射

本节包含用于现代化应用程序的 AWS Blu Age 和第三方完全限定的名称映射列表。

AWS Blu Age 完全限定名称映射

短名称	完全限定名称
`CallBuilder`	`com.netfective.bluage.gapwalk.runtime.statements.CallBuilder`
`Configuration`	`com.netfective.bluage.gapwalk.datasimplifier.configuration.Configuration`
`ConfigurationBuilder`	`com.netfective.bluage.gapwalk.datasimplifier.configuration.ConfigurationBuilder`

短名称	完全限定名称
ExecutionController	com.netfective.bluage.gapwalk.rt.call.ExecutionController
ExecutionControllerImpl	com.netfective.bluage.gapwalk.rt.call.internal.ExecutionControllerImpl
File	com.netfective.bluage.gapwalk.rt.io.File
MainProgramRunner	com.netfective.bluage.gapwalk.rt.call.MainProgramRunner
Program	com.netfective.bluage.gapwalk.rt.provider.Program
ProgramContextStore	com.netfective.bluage.gapwalk.rt.context.ProgramContextStore
ProgramRegistry	com.netfective.bluage.gapwalk.rt.provider.ProgramRegistry
Record	com.netfective.bluage.gapwalk.datasimplifier.data.Record
RecordEntity	com.netfective.bluage.gapwalk.datasimplifier.entity.RecordEntity
RuntimeContext	com.netfective.bluage.gapwalk.rt.context.RuntimeContext
SimpleStateMachineController	com.netfective.bluage.gapwalk.rt.statemachine.SimpleStateMachineController

短名称	完全限定名称
StateMachineController	com.netfective.bluage.gapwalk.rt.statemachine.StateMachineController
StateMachineRunner	com.netfective.bluage.gapwalk.rt.statemachine.StateMachineRunner

第三方完全限定名称映射

短名称	完全限定名称
@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

Bl AWS u Age 中的数据简化器是什么

在大型机和中型机系统（以下主题中称为“遗留”系统）上，常用的编程语言（例如 COBOL、PL/I 或 RPG）提供对内存的低级别访问。这种访问侧重于通过本机类型（例如分区、打包或字母数字，也可能包括组或数组聚合）访问内存布局。

在给定程序中，通过类型化字段和直接访问字节（原始内存）来访问给定内存的两种方式共存。例如，COBOL 程序会将参数作为连续的字节集（LINKAGE）传递给调用方，或者以相同的方式从文件（记录）中读取/写入数据，同时使用在 copybook 中组织的类型化字段来解析此类内存范围。

这种对内存的原始访问和结构化访问的组合、对精确的字节级内存布局的依赖以及遗留类型（例如分区或打包）是在 Java 编程环境中不容易实现的非本机特征。

作为将传统程序现代化为 Java 的 AWS Blu Age 解决方案的一部分，Data Simplifier 库为现代化的 Java 程序提供了此类结构，并以 Java 开发人员尽可能熟悉的方式公开这些结构（获取器/设置器、字节数组、基于类的字节数组）。数据简化器是从这些程序生成的现代化 Java 代码的核心依赖项。

为简单起见，以下大部分解释都基于 COBOL 结构，但您可以对两者使用相同的 API PL1 和 RPG 数据布局现代化，因为大多数概念都是相似的。

主题

- 主要的类
- 数据绑定和访问
- 所讨论的 Java 类型的 FQN

主要的类

为了便于阅读，本文档使用了 AWS Blu Age API 接口和类的 Java 简称。有关更多信息，请参阅 所讨论的 Java 类型的 FQN。

低级别内存表示

在最低级别上，内存（以快速、随机方式访问的连续字节范围）由 `Record` 接口表示。该接口本质上是固定大小字节数组的抽象。因此，它提供了能够访问或修改底层字节的 setter 和 getter。

结构化数据表示

要表示结构化数据，例如 COBOL DATA DIVISION 中的“01 data item”或“01 copybook”，则使用 `RecordEntity` 类的子类。它们通常不是手写的，而是由 AWS 蓝光时代的现代化工具从相应的遗留

结构中生成的。了解它们的主要结构和 API 也仍有用，可方便您了解现代化程序中的代码如何使用它们。就 COBOL 而言，该代码是从 PROCEDURE DIVISION 生成的 Java。

生成的代码使用 `RecordEntity` 子类表示每个“01 data item”；组成它的每个基本字段或聚合都表示为一个私有 Java 字段，采用树状结构进行组织（每个项都有一个父项，根项除外）。

为便于说明，以下是一个 COBOL 数据项示例，后面是相应的 B AWS lu Age 生成的对其进行现代化改造的代码：

```
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);
    }
}
```

基本字段

类 Elementary（或者 Filler，如果未命名）的字段代表遗留数据结构的“叶子”。这些字段与底层字节的连续跨度（“范围”）相关联，并且通常具有表示如何解析和修改这些字节（通过分别对字节数组中的值“解码”和“编码”）的类型（可能已参数化）。

所有基本类型都是 RangeType 的子类。常见的类型如下：

COBOL 类型	数据简化器类型
PIC X(n)	AlphanumericType
PIC 9(n)	ZonedType
PIC 9(n) COMP-3	PackedType
PIC 9(n) COMP-5	BinaryType

聚合字段

聚合字段组织其内容（其他聚合或基本字段）的内存布局，本身没有基本类型。

Group 字段表示内存中的连续字段。其中包含的每个字段在内存中的布局顺序相同，第一个字段位于偏移 0（相对于组字段在内存中的位置），第二个字段处于偏移 0 + (size in bytes of first field)，依此类推。它们用于表示同一包含字段下的 COBOL 字段序列。

Union 字段表示访问同一内存的多个字段。其中包含的每个字段都布局在偏移 0（相对于内存中的 Union 字段位置）。例如，它们用于表示 COBOL“REDEFINES”结构（第一个 Union 子项是重新定义的数据项，第二个子项是对第一个的重定义，以此类推）。

数组字段（Repetition 的子类）表示其子字段（无论本身是聚合还是基本项目）在内存中的布局重复。它们在内存中布局了给定数量的此类子布局，每个字段都位于偏移 index * (size in bytes of child)，用于表示 COBOL“OCCURS”结构。

基本数据类型

在某些现代化案例中，“基本数据类型”也可以单独用来表示“根”数据项。它们的使用方法与 RecordEntity 非常相似，但并非来自于它，也不是基于生成的代码。相反，它们由 AWS Blu Age 运行时作为Primitive接口的子类直接提供。提供的此类的示例包括 Alphanumeric 或 ZonedDecimal。

数据绑定和访问

结构化数据和底层数据之间的关联可以通过多种方式实现。

用于实现关联的一个重要接口是 RecordAdaptable，该接口用于获取提供 RecordAdaptable 底层数据的“可写视图”的 Record。如下所示，多个类实现了 RecordAdaptable。反过来， AWS Blu Age APIs 和操纵低级内存（例如程序参数、文件 I/O 记录、CICS 通信区域、分配的内存...）的代码通常会期望 a RecordAdaptable 作为该内存的句柄。

在 COBOL 现代化案例中，大多数数据项都与内存相关联，内存在相应程序执行的生命周期内固定。为此，RecordEntity 子类在生成的父对象（程序上下文）中实例化一次，并根据 RecordEntity 字节大小负责实例化其底层 Record。

在其他 COBOL 情况下，例如将 LINKAGE 元素与程序参数相关联，或者对 SET ADDRESS OF 结构进行现代化，必须将 RecordEntity 实例与提供的 RecordAdaptable 关联。为此，可使用两种机制：

- 如果 RecordEntity 实例已经存在，则可以使用 RecordEntity.bind(RecordAdaptable) 方法（继承自 Bindable）使此实例“指向”此 RecordAdaptable。然后，在 RecordEntity 上调用的任何 getter 或 setter 都将由底层 RecordAdaptable 字节支持（字节读取或写入）。
- 如果要将 RecordEntity 实例化，生成的接受 RecordAdaptable 的结构可用。

相反，可以访问结构化数据当前绑定的 Record。为此，RecordEntity 实现了 RecordAdaptable，因此可以在任何此类实例上调用 getRecord()。

最后，许多 COBOL 或 CICS 动词都需要访问单个字段来进行读写。RangeReference 类用于表示此类访问。它的实例可以从 RecordEntity 生成的 getXXXReference() 方法（XXX 即访问的字段）中获取，然后传递给运行时方法。RangeReference 通常用于访问整个 RecordEntity 或Group，而其子类 ElementaryRangeReference 表示对 Elementary 字段的访问。

请注意，上面的大多数观察结果都适用于Primitive子类，因为它们努力实现与 AWS Blu Age 运行RecordEntity时提供的行为相似的行为（而不是生成的代码）。为此，Primitive 的所有

子类实现 RecordAdaptable、ElementaryRangeReference 和 Bindable，以便可以代替 RecordEntity 子类和基本字段。

所讨论的 Java 类型的 FQN

下表显示了本节中所讨论 Java 类型的完全限定名称。

短名称	完全限定名称
Alphanumeric	com.netfective.bluage.gapwalk.datasimplifier.elementary.Alphanumeric
AlphanumericType	com.netfective.bluage.gapwalk.datasimplifier.metadata.type.AlphanumericType
BinaryType	com.netfective.bluage.gapwalk.datasimplifier.metadata.type.BinaryType
Bindable	com.netfective.bluage.gapwalk.datasimplifier.data.Bindable
Elementary	com.netfective.bluage.gapwalk.datasimplifier.data.structure.Elementary
ElementaryRangeReference	com.netfective.bluage.gapwalk.datasimplifier.entity.ElementaryRangeReference
Filler	com.netfective.bluage.gapwalk.datasimplifier.data.structure.Filler
Group	com.netfective.bluage.gapwalk.datasimplifier.data.structure.Group

短名称	完全限定名称
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
Union	com.netfective.bluage.gapwalk.datasimplifier.data.structure.Union

短名称	完全限定名称
ZonedDecimal	com.netfective.bluage.gapwalk.datasimplifier.elementary.ZonedDecimal
ZonedType	com.netfective.bluage.gapwalk.datasimplifier.metadata.type.ZonedType

AWS Blu Age Blusam

在大型机系统（以下主题中称为“遗留系统”）上，业务数据通常使用 VSAM（虚拟存储访问方法）存储。数据存储在“记录”（字节数组）中，属于“数据集”。

有四个数据集组织：

- KSDS：按键排序的数据集：按主键（不支持重复的键）和可选的其他“备用”键为记录编制索引。所有键值都是记录字节数组的子集，每个键由以下内容定义：
  - 偏移量 [基于 0（0 是记录字节数组内容的开头），以字节为单位]
  - 长度（以字节为单位）
  - 是否支持重复的值
- ESDS：按条目排序的数据集：记录主要按顺序访问（基于它们在数据集中的插入顺序），但可以使用其他备用键进行访问；
- RRDS：相对记录数据集：使用“跳跃”访问记录，使用相对记录号；可以向前跳跃，也可以向后跳跃；
- LDS：线性数据集：这里没有记录，只有一个按页组织的字节流。主要用于遗留平台上的内部用途。

在使用 AWS Blu Age 重构方法对传统应用程序进行现代化改造时，现代化的应用程序不再打算访问 VSAM 存储的数据，同时保留数据访问逻辑。该Blusam组件就是答案：它允许从传统 VSAM 数据集导出中导入数据，为现代化应用程序提供操作它们的 API 以及专用的管理 Web 应用程序。请参阅the section called “Blusam管理控制台”。

> **Note**
>
> Blusam仅支持 KSDS、ESDS 和 RRDS。

BlusamAPI 使保留数据访问逻辑（顺序、随机和相对读取；插入、更新和删除记录）成为可能，而组件架构依赖于缓存策略和基于 RDBMS 的存储的组合，允许在资源有限的情况下进行高吞吐量 I/O 操作。

Blusam 基础设施

Blusam依赖 PostgreSQL RDBMS 存储数据集，包括原始记录数据和密钥索引（如果适用）。最喜欢的选择是使用兼容亚马逊 Aurora PostgreSQL 的引擎。本主题中的示例和插图均基于该引擎。

> **Note**
>
> 服务器启动时，Blusam运行时会检查是否存在一些必需的技术表，如果找不到这些表，则创建它们。因此，必须向配置中使用的访问Blusam数据库的角色授予创建、更新和删除数据库表（行和表定义本身）的权限。有关如何禁用的信息Blusam，请参阅the section called “Blusam 配置”。

缓存

除了存储本身之外，当与缓存实现相结合时，Blusam运行速度更快。

目前支持两个缓存引擎， EhCache 以及 Redis，每个引擎都有自己的用例：

- EhCache : 独立嵌入式易失性本地缓存
  - 不符合 AWS 大型机现代化托管环境部署的资格。
  - 通常在使用单个节点（例如单个 Apache Tomcat 服务器）运行现代化应用程序时使用。例如，该节点可能专用于托管批处理作业任务。
  - V@@ ol atile： EhCache 缓存实例是易失的；其内容将在服务器关闭时丢失。
  - 嵌入式： EhCache 和服务器共享相同的 JVM 内存空间（在定义托管计算机规格时要考虑这个问题）。
- Redis：共享永久缓存
  - 有资格部署 AWS 大型机现代化托管环境。

- 通常用于多节点情况，尤其是当多个服务器共用一个负载均衡器时。缓存内容在所有节点之间共享。
- Redis 是永久性的，不受节点生命周期的约束。它在自己的专用计算机或服务（例如 Amazon ElastiCache）上运行。缓存对所有节点都是远程的。

锁定

要处理对数据集和记录的并行访问，Blusam需要使用可配置的锁定系统。锁定可以应用于两个级别：数据集和记录：

- 针对写入目的锁定数据集将阻止所有其他客户端在任何级别（数据集或记录）对其执行写入操作。
- 在记录级别针对写入进行锁定只会阻止其他客户端对给定记录执行写入操作。

应根据缓存配置相应地配置Blusam锁定系统：

- 如果选择作为缓存实现， EhCache 则无需进行进一步的锁定配置，因为应使用默认的内存中锁定系统。
- 如果选择 Redis 作为缓存实现，则需要进行基于 Redis 的锁定配置，以支持从多个节点进行并发访问。用于锁的 Redis 缓存不必与用于数据集的 Redis 缓存相同。有关配置基于 Redis 的锁定系统的信息，请参见 the section called “Blusam 配置”。

Blusam内在函数和从旧版迁移数据

存储数据集：记录和索引

每个传统数据集在导入到Blusam后都将存储到一个专用表中；该表的每一行使用两列代表一条记录：

- 数字 ID 列，大整数类型，即表的主键，用于存储记录的相对字节地址（RBA）。RBA 表示自数据集开头的偏移量（以字节为单位），从 0 开始。
- 字节数组记录列，用于存储原始记录的内容。

例如，请参阅 CardDemo 应用程序中使用的 KSDS 数据集的内容：

```
SELECT * FROM public.aws_m2_carddemo_acctdata_vsam_ksds
ORDER BY id ASC
```

Data output Messages Notifications

	id [PK] bigint	record bytea
1	0	[binary data]
2	300	[binary data]
3	600	[binary data]
4	900	[binary data]
5	1200	[binary data]
6	1500	[binary data]
7	1800	[binary data]
8	2100	[binary data]
9	2400	[binary data]
10	2700	[binary data]
11	3000	[binary data]
12	3300	[binary data]
13	3600	[binary data]

- 这个特定的数据集包含固定长度的记录，长度为 300 字节（因此 id 的集合是 300 的倍数）。
- 默认情况下，用于查询 PostgreSQL 数据库的 pgadmin 工具不显示字节数组列内容，而是打印 [二进制数据] 标签。
- 原始记录内容与从遗留平台导出的原始数据集相匹配，无需进行任何转换。特别是，不会发生字符集转换；这意味着记录的字母数字部分必须由使用遗留字符集（很可能是 EBCDIC 变体）的现代化应用程序进行解码。

关于数据集元数据和键索引：每个数据集都与名为 `metadata` 的表中的两行相关联。这是默认的命名约定。若要了解如何对其进行自定义，请参阅 the section called “Blusam 配置”。

	name [PK] text	metadata oid
1	AWS_M2_CARDDEMO_ACCTDATA_VSAM_KSDS	66320
2	AWS_M2_CARDDEMO_ACCTDATA_VSAM_KSDS__internal	66321

- 第一行将数据集名称作为 name 列的值。metadata 列是一个二进制列，其中包含给定数据集的一般元数据的二进制序列化。有关更多信息，请参阅 the section called “常规数据集元数据属性”。
- 第二行将数据集名称加上 `__internal'` 后缀作为 name 列的值。metadata 列的二进制内容取决于数据集的“大小”。
  - 对于 small/medium 数据集，内容是以下内容的压缩序列化：
    - 数据集使用的键的定义；主键定义（对于 KSDS）和备用键定义（如果适用）（对于 KSDS/ESDS）
    - 键索引（如果适用）（带有备用键定义的 KSDS/ESDS）：用于记录的索引浏览；键索引将键值映射到记录的 RBA；
    - 记录长度映射：用于记录的顺序/相对浏览；
  - 对于 Large/Very 大型数据集，内容是以下内容的压缩序列化：
    - 数据集使用的键的定义；主键定义（对于 KSDS）和备用键定义（如果适用）（对于 KSDS/ESDS）

此外， large/very 大型数据集索引（如果适用）使用分页机制存储；索引页的二进制序列化存储为专用表的行（每个数据集键一个表）。每个索引页存储在一行中，其中包含以下几列：

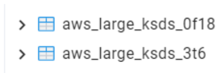
- id：索引页的技术标识符（数字主键）；
- firstkey：存储在索引页中的第一个（最低）键值的二进制值；
- lastkey：存储在索引页中的最后一个（最高）键值的二进制值；
- metadata：索引页的二进制压缩序列化（将键值映射到记录 RBAs）。

	id [PK] bigint	firstkey bytea	lastkey bytea	metadata oid
1	1	[binary data]	[binary da...	6458928
2	2	[binary data]	[binary da...	6458929
3	3	[binary data]	[binary da...	6458930
4	4	[binary data]	[binary da...	6458931
5	5	[binary data]	[binary da...	6458932
6	6	[binary data]	[binary da...	6458933
7	7	[binary data]	[binary da...	6458934
8	8	[binary data]	[binary da...	6458935
9	9	[binary data]	[binary da...	6458936

表名称是数据集名称和键内部名称的串联，键内部名称包含键的相关信息，例如键偏移量、键是否接受重复项（设置为 true 将支持重复项）以及键长度。例如，假设一个名为 “AWS_LARGE_KSDS” 的数据集具有以下两个已定义的密钥：

- 主键 [偏移量：0，重复项：false，长度：18]
- 备用键 [偏移量：3，重复项：true，长度：6]

在本例中，下表存储了与这两个键相关的索引。



使用写入 I/O 机制优化吞吐量

为了优化插入/更新/删除操作的性能，该Blusam引擎依赖于可配置的写入机制。该机制建立在一个专用线程池之上，这些线程使用批量更新查询处理持久性操作，以最大限度地提高Blusam存储 I/O 吞吐量。

该Blusam引擎收集应用程序对记录执行的所有更新操作，并生成发送到专用线程进行处理的记录批次。然后，使用批量更新查询将这些批次持久化到Blusam存储中，避免使用原子持久性操作，从而确保网络带宽得到最佳利用。

该机制使用可配置的延迟（默认为一秒）和可配置的批大小（默认为 10000 条记录）。只要满足以下两个条件中的第一个，就会立即执行构建持久性查询：

- 配置的延迟已过且批不为空
- 批中要处理的记录数达到配置的限制

若要了解如何配置回写机制，请参阅the section called “可选属性”。

选择合适的存储方案

如上一节所述，数据集的存储方式取决于它们的“大小”。但是对于一个数据集而言，什么是小、中、大呢？ 何时应该选择分页存储策略而不是常规存储策略？

该问题的答案取决于以下几点。

- 托管将使用这些数据集的现代化应用程序的每个服务器上的可用内存容量。
- 缓存基础设施（如果有）上的可用内存容量。

使用非分页索引存储方案时，每个数据集的完整键索引和记录大小信息集合都会在数据集打开时加载到服务器内存中。此外，如果涉及缓存，则可能会使用常规方法将所有数据集记录预加载到缓存中，这可能会导致缓存基础设施端的内存资源耗尽。

根据定义的键的数量、键值的长度、记录数量和同时打开的数据集数量，可以粗略地估算出给定的已知用例将消耗的内存容量。

要了解更多信息，请参阅the section called “估算给定数据集的内存占用”。

Blusam 迁移

为给定数据集选择了正确的存储方案后，必须通过迁移传统数据集来填充Blusam存储空间。

为此，必须使用遗留数据集的原始二进制导出，在导出过程中不使用任何字符集转换。从遗留系统传输数据集导出时，请确保不要破坏二进制格式。例如，使用 FTP 时强制使用二进制模式。

原始二进制导出仅包含记录。导入机制不需要导 keys/indexes 出，因为所有导出 keys/indexes 都是通过导入机制即时重新计算的。

数据集二进制导出可用后，有几种方法可以将其迁移到Blusam存在：

在 AWS 大型机现代化托管环境中：

- 使用专用特征导入数据集。请参阅the section called “为 应用程序导入数据集”。

或者

- 使用数据集批量导入工具。请参阅 the section called “数据集定义参考”和the section called “VSAM 数据集请求格式示例”。

或者

- 利用专用的加载服务，使用 groovy 脚本导入数据集。

> **Note**
>
> 目前，只能使用 groovy 脚本在 Mainframe Modernization 托管环境中导入 LargeKSDS 和 LargeESDS。

在亚马逊上的 AWS Blu Age Runtime 上 EC2：

- 使用导入数据集the section called “Blusam管理控制台”。

或者

- 利用专用的加载服务，使用 groovy 脚本导入数据集。

使用 Groovy 脚本导入数据集

本节将帮助您编写用于将旧数据集导入的groovy脚本。Blusam

首先要进行一些必需的导入：

```
import com.netfective.bluage.gapwalk.bluesam.BluesamManager
import com.netfective.bluage.gapwalk.bluesam.metadata.Key;
import com.netfective.bluage.gapwalk.rt.provider.ServiceRegistry
import java.util.ArrayList; //used for alternate keys if any
```

之后，对于要导入的每个数据集，根据给定的模式构建代码：

1. 创建或清除映射对象
2. 在映射中填充所需的属性（这因数据集类型而异，详见下文）
3. 在服务注册表中检索要用于数据集类型的适当加载服务
4. 使用映射作为参数运行服务

可以使用以下标识符从服务注册表中检索 5 种服务实现：

- `"BluesamKSDSFileLoader"`: 适用于 small/medium 大小的 KSDS
- `"BluesamESDSFileLoader"`适用于 small/medium 大小 ESDS
- `"BluesamRRDSFileLoader"`：适用于 RRDS
- `"BluesamLargeKSDSFileLoader"`：适用于大型 KSDS
- `"BluesamLargeESDSFileLoader"`：适用于大型 ESDS

为 KSDS/ESDS 选择常规版还是大型版服务取决于数据集的大小以及要应用于数据集的存储策略。要了解如何选择适当的存储策略，请参阅the section called “选择合适的存储方案”。

为了能够成功地将数据集导入Blusam，必须向加载服务提供正确的属性。

常用属性：

- 必需（适用于所有类型的数据集）
  - “bluesamManager”：预期值为 `applicationContext.getBean(BluesamManager.class)`
  - “datasetName”：数据集名称，为字符串
  - “inFilePath"：旧数据集导出的路径，为字符串
  - “recordLength”：固定记录长度（对于可变记录长度数据集，为 0），为整数
- 可选
  - 不支持大型数据集：
    - “isAppend”：一个布尔标志，表示导入是在追加模式下进行的（将记录追加到现有Blusam数据集）。
    - “useCompression”：一个布尔标志，指示将使用压缩来存储元数据。
  - 仅适用于大型数据集：
    - “indexingPageSizeInMb"：数据集的每个键的每个索引页的大小（以兆字节为单位），严格为正整数

依赖于数据集类型的属性：

- KSDS/大型 KSDS：
  - 必需：
    - “primaryKey”：使用 `com.netfective.bluage.gapwalk.bluesam.metadata.Key` 构造函数调用构建的主键定义。
  - 可选：
    - “alternateKeys”：使用 `com.netfective.bluage.gapwalk.bluesam.metadata.Key` 构造函数调用构建的备用键定义列表（`java.util.List`）。
- ESDS/大型 ESDS：
  - 可选：
    - “alternateKeys”：使用 `com.netfective.bluage.gapwalk.bluesam.metadata.Key` 构造函数调用构建的备用键定义列表（`java.util.List`）。
- RRDS：
  - 无。

键构造函数调用：

- `new Key(int offset, int length)`：创建一个键对象，该对象具有给定的键属性（偏移量和长度），并且不支持重复项。此变体应该用于定义主键。
- `new Key(boolean allowDuplicates, int offset, int length)`：创建一个键对象，该对象具有给定的键属性（偏移量和长度），并带有支持重复项标志。

以下 Groovy 示例展示了各种加载场景。

加载具有两个备用键的大型 KSDS：

```
import com.netfective.bluage.gapwalk.bluesam.BluesamManager
import com.netfective.bluage.gapwalk.bluesam.metadata.Key;
import com.netfective.bluage.gapwalk.rt.provider.ServiceRegistry
import java.util.ArrayList;

// Loading a large KSDS into Blusam
def map = [:]
map.put("bluesamManager", applicationContext.getBean(BluesamManager.class));
map.put("datasetName", "largeKsdsSample");
map.put("inFilePath", "/work/samples/largeKsdsSampleExport");
map.put("recordLength", 49);
map.put("primaryKey", new Key(0, 18));
ArrayList altKeys = [new Key(true, 10, 8), new Key(false, 0, 9)]
map.put("alternateKeys", altKeys);
map.put("indexingPageSizeInMb", 25);
def service = ServiceRegistry.getService("BluesamLargeKSDSFileLoader");
service.runService(map);
```

加载没有备用键的可变记录长度 ESDS：

```
import com.netfective.bluage.gapwalk.bluesam.BluesamManager
import com.netfective.bluage.gapwalk.bluesam.metadata.Key;
import com.netfective.bluage.gapwalk.rt.provider.ServiceRegistry

// Loading an ESDS into Blusam
def map = [:]
map.put("bluesamManager", applicationContext.getBean(BluesamManager.class));
map.put("datasetName", "esdsSample");
map.put("inFilePath", "/work/samples/esdsSampleExport");
map.put("recordLength", 0);
```

```
def service = ServiceRegistry.getService("BluesamESDSFileLoader");
service.runService(map);
```

可变记录长度数据集导出将包含必需的记录描述字（RDW）信息，以支持在读取时对记录进行拆分。

加载固定记录长度 RRDS：

```
import com.netfective.bluage.gapwalk.bluesam.BluesamManager
import com.netfective.bluage.gapwalk.bluesam.metadata.Key;
import com.netfective.bluage.gapwalk.rt.provider.ServiceRegistry

// Loading a RRDS into Blusam
def map = [:]
map.put("bluesamManager", applicationContext.getBean(BluesamManager.class));
map.put("datasetName", "rrdsSample");
map.put("inFilePath", "/work/samples/rrdsSampleExport");
map.put("recordLength", 180);
def service = ServiceRegistry.getService("BluesamRRDSFileLoader");
service.runService(map);
```

在多架构模式下加载数据集：

多架构模式：在某些传统系统中，VSAM 文件被组织成文件集，允许程序访问和修改指定分区内的数据。现代系统将每个文件集视为一个架构，从而实现类似的数据分区和访问控制。

要在`application-main.yml`文件中启用多架构模式，请参阅。the section called “Blusam 配置”在此模式下，可以使用共享上下文将数据集加载到特定架构中，共享上下文是用于存储运行时信息的内存注册表。要将数据集加载到特定架构中，请在数据集名称前加上相关的架构名称。

将 KSDS 文件加载到多架构模式的特定架构中：

```
import com.netfective.bluage.gapwalk.bluesam.BluesamManager
import com.netfective.bluage.gapwalk.bluesam.metadata.Key;
import com.netfective.bluage.gapwalk.rt.provider.ServiceRegistry
import java.util.ArrayList;
import com.netfective.bluage.gapwalk.rt.shared.SharedContext;

// Loading a KSDS into Blusam
def map = [:]
String schema = "schema1";
String datasetName = schema+"|"+"ksdsSample";
SharedContext.get().setCurrentBlusamSchema(schema);
```

```
schema = SharedContext.get().getCurrentBlusamSchema();
map.put("bluesamManager", applicationContext.getBean(BluesamManager.class));
map.put("datasetName", datasetName);
map.put("inFilePath", "/work/samples/ksdsSampleExport");
map.put("recordLength", 49);
map.put("primaryKey", new Key(0, 18));
map.put("indexingPageSizeInMb", 25);
def service = ServiceRegistry.getService("BluesamKSDSFileLoader");
service.runService(map);
```

将大型 KSDS 文件加载到多架构模式的特定架构中：

```
import com.netfective.bluage.gapwalk.bluesam.BluesamManager
import com.netfective.bluage.gapwalk.bluesam.metadata.Key;
import com.netfective.bluage.gapwalk.rt.provider.ServiceRegistry
import java.util.ArrayList;
import com.netfective.bluage.gapwalk.rt.shared.SharedContext;

// Loading a Large KSDS into Blusam
def map = [:]
String schema = "schema1";
String datasetName = schema+"|"+"largeKsdsSample";
SharedContext.get().setCurrentBlusamSchema(schema);
schema = SharedContext.get().getCurrentBlusamSchema();
map.put("bluesamManager", applicationContext.getBean(BluesamManager.class));
map.put("datasetName", datasetName);
map.put("inFilePath", "/work/samples/LargeKsdsSampleExport");
map.put("recordLength", 49);
map.put("primaryKey", new Key(0, 18));
map.put("indexingPageSizeInMb", 25);
def service = ServiceRegistry.getService("BluesamLargeKSDSFileLoader");
service.runService(map);
```

此外，配置条目（将在`application-main.yml`配置文件中设置）可用于微调导入过程：

- `bluesam.fileLoading.commitInterval`: 一个严格的正整数，定义常规ESDS/KSDS/RRDS导入机制的提交间隔。不适用于大型数据集的导入。默认值为 100000。

Blusam 配置

配置Blusam发生在`application-main.yml`配置文件中（或Blusam管理控制台--BAC--应用程序的独立部署的`application-bac.yml`配置文件中）。

Blusam必须从两个方面进行配置：

- Blusam存储和缓存访问配置
- Blusam发动机配置

Blusam存储和缓存访问配置

有关如何使用密钥管理器或数据源配置对Blusam存储和缓存的访问权限的信息，请参阅。the section called “AWS Blu Age 运行时配置”

> **Note**
>
> 关于对Blusam存储的访问权限，使用的凭据将指向具有相应权限的连接角色。为了使Blusam引擎能够按预期运行，连接角色必须具有以下权限：

- 连接到数据库
- 创建/删除/更改/截断表和视图
- 选择/插入/删除/更新表格和视图中的行
- 执行函数或过程

Blusam发动机配置

禁用 Blusam 支持

首先，让我们提一下，通过将`bluesam.disabled`属性设置为，可以完全禁用Blusam支持`true`。应用程序启动时，服务器日志中将显示一条信息消息，提醒Blusam禁用：

```
BLUESAM is disabled. No operations allowed.
```

在这种情况下，无需进一步配置，任何尝试使用Blusam相关功能（无论Blusam是以编程方式还是通过 REST 调用）都将引发 Java 代码执行`UnsupportedOperationException`中的问题，并附上有关Blusam被禁用的相关解释消息。

Blusam发动机属性

Blusam引擎配置属性在 bluesam key 前缀下重新分组：

必需属性

- `cache`：将以所选择的缓存实现赋值。有效值为：
  - `ehcache`：使用本地嵌入式 ehcache。请参阅上面的相关用例限制。
  - `redis`：使用共享远程 redis 缓存。这是 AWS 大型机现代化托管用例的首选选项。
  - none：禁用存储缓存
- `persistence`: 待用 pgsql 进行估值（PostgreSQL 引擎：最低版本 10.0 — 推荐版本 >=14.0
- 数据源参考：`<persistence engine>.dataSource`将指向用于Blusam存储连接的 DataSource 定义，该定义在配置文件的其他地方定义。这通常被命名为 `bluesamDs`。

> **Note**
>
> 每当使用 Redis 作为缓存机制时，无论是用于数据还是锁（请见下文），都需要配置对 Redis 实例的访问权限。有关详细信息，请参阅 the section called “可用的 Redis 缓存属性”。

可选属性

Blusam锁：属性前缀为 `locks`

- `cache`: 唯一可用的值是`redis`，指定将使用基于 redis 的锁定机制（当Blusam存储缓存也基于 redis 时使用）。如果该属性缺失或未设置为 `redis`，则将使用默认的内存内锁定机制。
- `lockTimeOut`：一个正的长整数值，用于指定尝试锁定已锁定的元素被标记为失败之前的超时时间（以毫秒为单位）。默认值为 `500`。
- `locksDeadTime`：一个正的长整数值，表示应用程序可以持有锁的最长时间（以毫秒为单位）。在该时间过后，锁会自动标记为已过期并被释放。默认值为 `1000`；
- `locksCheck`: 一个字符串，用于定义当前锁管理器使用的关于移除过期Blusam锁的锁定检查策略。将从以下值中选择：
  - `off`：不执行任何检查。不建议，因为可能会发生死锁。
  - `reboot`：在重启或应用程序启动时执行检查。届时所有过期的锁都会被释放。这是默认值。
  - `timeout`：在重启或应用程序启动时执行检查，或者在尝试锁定数据集超时时执行检查。过期的锁将立即释放。

回写机制：这些属性以 `write-behind` 键为前缀：

- enabled：true（默认值和建议值）或 false，用于启用或禁用回写机制。禁用该机制会极大地影响写入性能，因此不建议这样做。
- maxDelay：要触发的线程的最大持续时间。默认值为 "1s"（1 秒）。除非特殊情况需要调整该值，否则保持默认值通常是个不错的选择。无论如何，该值都应保持在较低水平（低于 3 秒）。该延迟字符串的格式为：<integer value><optional whitespace><time unit>，其中 <time unit> 将从以下值中选择：
  - "ns"：纳秒
  - "µs"：微秒
  - "ms"：毫秒
  - "s"：秒
- threads：专用回写线程的数量。默认为 5。您需要根据运行Blusam引擎的主机的计算能力调整此值。使用更高的值并不意味着就能获得更高的性能，因为限制因素将变成存储 RDBMS 处理大量并发批处理查询的能力。建议值通常在 4-8 之间。
- batchSize：一个正整数，表示批中将分配给线程进行批量处理的最大记录数。该值必须在 1 到 32767 之间。默认值为 10000。使用 1 作为值有违该机制的目的（即避免使用单个更新查询）；适合使用的最小值为 1000 左右。

嵌入式 EhCache 微调：属性以 ehcache key 为前缀：

- resource-pool:
  - size：为嵌入式缓存分配的内存大小，用字符串表示。默认为 "1024MB"（1 GB）。将根据托管Blusam引擎的计算机的可用内存和应用程序使用的数据集的大小进行调整。该大小字符串的格式为：<integer value><optional whitespace><memory unit>，其中 <memory-unit> 将从以下值中选择：
    - B：字节
    - KB：千字节
    - MB：兆字节
    - GB：千兆字节
    - TB：兆兆字节
  - heap：true 或 false，用于指示缓存是否会占用 JVM 堆内存。默认为 true（获得最快缓存性能的选项，但缓存存储会占用 JVM 堆内 RAM 内存）。将此属性设置为 false 将切换到堆外内存，由于需要与 JVM 堆进行交换，会使速度变慢。

- `timeToLiveMillis`：缓存条目在被视为过期并被删除之前在缓存中停留的持续时间（以毫秒为单位）。如果未指定此属性，则默认情况下，缓存条目不会自动过期。

大型数据集的可选属性：

分页索引的本地化内存缓存：

- `indexesPrefetchWindowSize`：此属性适用于使用启用 Redis 缓存的操作的 Blusam 大型数据集。它指定了可用于存储分页索引的最大内存缓存大小（以 MB 为单位）。默认值为 0。该值可以根据可用的系统内存和正在处理的数据集的大小进行调整。

示例配置片段：

```
dataSource:
  bluesamDs:
    driver-class-name: org.postgresql.Driver
    ...
    ...
bluesam:
  locks:
    lockTimeOut: 700
  cache: ehcache
  persistence: pgsql
  ehcache:
    resource-pool:
      size: 8GB
  write-behind:
    enabled: true
    threads: 8
    batchsize: 5000
    indexesPrefetchWindowSize: 25
  pgsql:
    dataSource : bluesamDs
```

多架构配置属性

- `multiSchema`: false（默认值）或 true，禁用或启用多架构模式 Blusam-从 4.4.0 版本开始可用。
- `pgsql`:
  - `schemas`：应用程序将在多架构模式下使用的架构名称列表。Blusam

- `fallbackSchema`：用于多架构模式的备用架构名称。如果在当前架构上下文中找不到数据集，则该架构将用于对该数据集进行Blusam相关操作。

示例配置片段（启用了多架构模式）：Blusam

```
dataSource:
  bluesamDs:
    driver-class-name: org.postgresql.Driver
    ...
    ...
bluesam:
  locks:
    lockTimeOut: 700
  cache: ehcache
  persistence: pgsql
  ehcache:
    resource-pool:
      size: 8GB
  write-behind:
    enabled: true
    threads: 8
    batchsize: 5000
  multiSchema: true
  pgsql:
    dataSource : bluesamDs
    schemas:
      - "schema1"
      - "schema2"
      - "schema3"
    fallbackSchema: schema3
```

> **Note**
> Blusam如果元数据架构不存在且用户有足够的权限，则会在Blusam数据库中创建这些架构，包括`application-main.yml`文件中列出的多架构模式。

BlusamDD 文件配置

这些选项可用于 groovy 中的Blusam文件配置（JCL 中的数据定义）：

- `largeKSDS()`: 表示大型 KSDS 性质。使用时，如果指定的文件是缺失文件并且应该创建该文件（取决于打开模式和可选文件选项），则将创建一个大型 KSDS 文件及其索引表。

```
.bluesam("TESTFILE").dataset("TESTFILE").largeKSDS().build()
```

- `indexPageSize(Integer param)`：指定要创建的索引表的页面大小（以 mb 为单位）。适用于带`largeKSDS()`选项的文件。的值`param`应严格为正数。对于无效值，将使用默认`param`值 15。

```
.bluesam("TESTFILE").dataset("TESTFILE").largeKSDS().indexPageSize(15).build()
```

可以在文件中设置默认文件配置，该`ds-config.yml`文件位于`entities/src/main/resources/ds-config.yml`（或配置文件的其他位置）。当没有向运行单元提供特定的文件配置时，这些设置将用作备用配置。中支持所有 Bluesam 文件配置选项。`ds-config.yml`

```
TESTFILE:
 provider: Bluesam
 dataset: TESTFILE
 largeKSDS: true
 indexPageSize: 15
```

Blusam管理控制台

Blusam管理控制台 (BAC) 是一个 Web 应用程序，用于管理存储。Blusam有关 BAC 的信息，请参阅 [the section called “Blusam管理控制台”]。

附录

常规数据集元数据属性

常规数据集元数据序列化属性列表：

- 数据集名称
- 类型（KSDS、LargeKSDS、ESDS、LargeESDS 或 RRDS）
- 缓存预热标志（是否在服务器启动时将数据集预加载到缓存中）
- 压缩使用标志（以压缩格式还是原始格式存储记录）
- 创建日期
- 上次修改日期

- 固定长度记录标志（数据集记录是否都具有相同的长度）
- 记录长度：只对固定记录长度有意义
- 页面大小（用于自定义在需要时用于预加载缓存的分页 sql 查询）
- 大小（数据集的大小：记录的累积长度）
- 最后偏移量 [添加到数据集的最新记录的偏移量（即 RBA）]
- 下一个偏移量（向数据集中添加新记录的下一个可用偏移量）
- （如果有意义）数据集使用的键的定义；每个键由其类型（主键或备用键集合的一部分）和三个属性定义：
  - 偏移量：键值起始字节在记录中的位置；
  - 长度：键值的长度（以字节为单位）。因此，键值是字节数组，该数组是从 `key offset` 位置开始，在 `key offset + length - 1` 位置结束的记录的子集；
  - 支持重复项标志：指示键是否接受重复项（设置为 true 表示支持重复项）。

估算给定数据集的内存占用

对于中小型数据集，元数据（各种键的大小和索引）将完全加载到内存中。为托管用于运行现代化应用程序的服务器的计算机分配适当的资源需要弄清楚Blusam数据集引起的内存消耗，尤其是元数据消耗。本节为有关的操作人员提供了切实可行的答案。

给定的公式仅适用于中Blusam小型数据集，不使用 “大” 存储策略。

Blusam数据集元数据

对于Blusam数据集，元数据分为两部分：

- 核心元数据：保存有关数据集的全局信息。与内部元数据相比，其内存占用量可以忽略不计。
- 内部元数据：保存有关记录大小和键索引的信息；当数据集不为空时，这就是加载到托管现代化应用程序的应用程序服务器时占用内存的内容。以下各节详细说明了内存占用量如何随着记录数量的增加而增长。

计算内部元数据的空间占用

记录大小映射

首先，内部元数据存储了一个映射来保存每个记录的大小（为一个整数），给定其 RBA（相对字节地址，作为一个长数字存储）。

该数据结构的内存占用量为 `80 * number of records`（以字节为单位）

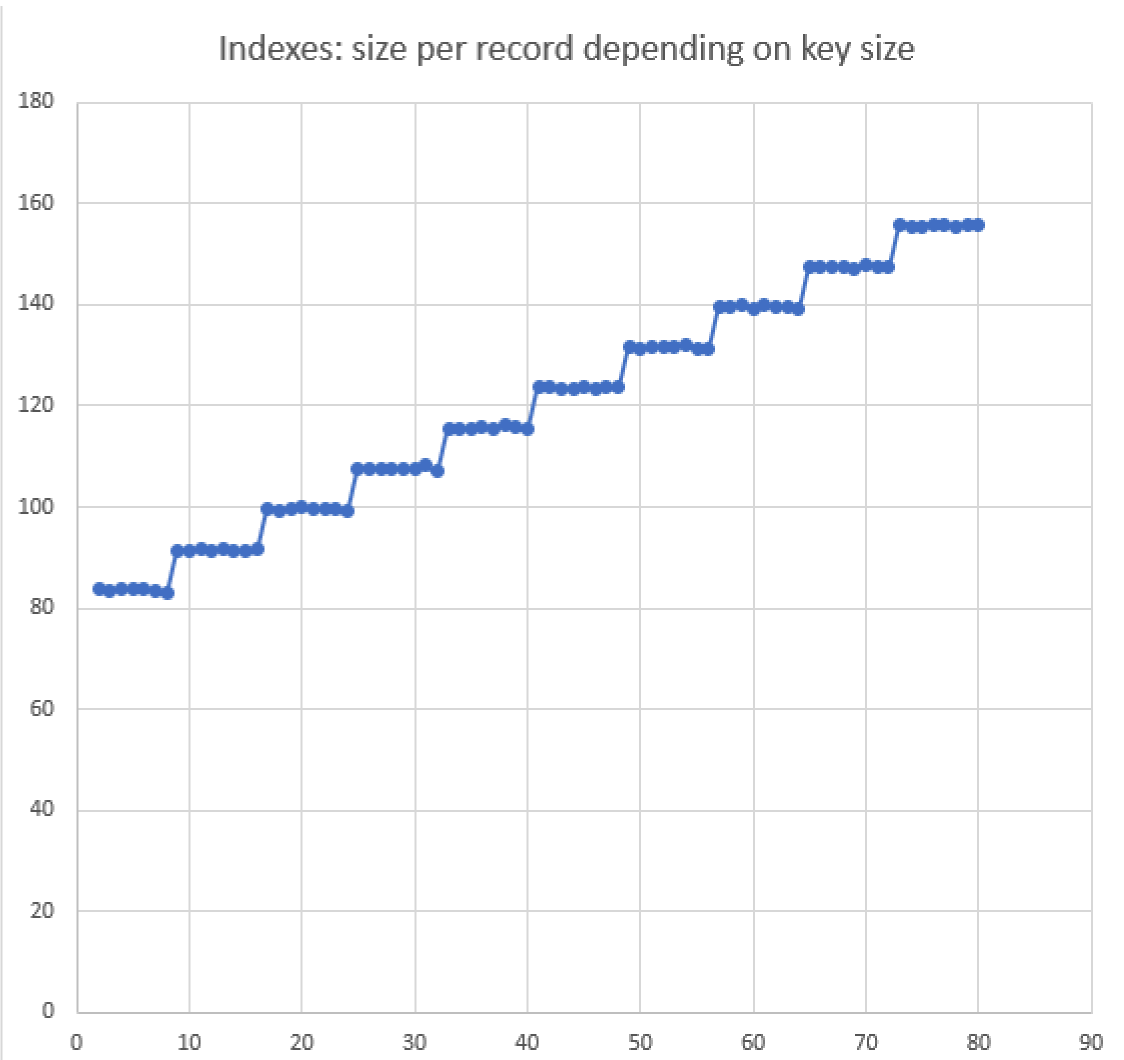
这适用于所有数据集类型。

索引

关于 KSDS 的主键或 ESDS 和 KSDS 上的备用键的索引，空间占用量的计算取决于两个因素：

- 数据集中的记录数；
- 键的大小（以字节为单位）。

下图显示了基于键大小（x 轴）的每个记录的键索引大小（y 轴）。



用于估算数据集给定键索引的空间占用的相应公式为：

```
index footprint = number of records * ( 83 + 8 (key length / 8))
```

其中“/”表示整数除。

示例：

- 数据集 1：
  - 记录数 = 459 996
  - 键长度 = 15，因此，(键长度 / 8) = 1
  - 索引占用空间 = 459 996 * (83 + (8*1)) = 41 859 636 字节（约等于 39 MB）
- 数据集 2：
  - 记录数 = 13 095 783
  - 键长度 = 18，因此，(键长度 / 8) = 2
  - 索引占用空间 = 13 095 783 * (83 + (8*2)) = 1 296 482 517 字节（约等于 1.2 GB）

给定数据集的总占用空间是所有键索引的总占用空间与记录大小映射的占用空间之和。

例如，以只有一个键的数据集 2 为例，总占用空间为：

- 记录大小映射：13 095 783 * 80 = 1 047 662 640 字节
- 键索引：1 296 482 517 字节（见上文）
- 总占用空间 = 2 344 145 157 字节（约等于 2.18 GB）

AWS Blu Age Blusam 管理控制台

Blusam管理控制台 (BAC) 是一款用于处理Blusam数据集的安全 Web 应用程序。本指南涵盖了 BAC 用户界面。有关通过 REST 端点进行远程管理的信息，请参阅 the section called “Blusam应用程序控制台 REST 端点”。

主题

部署 BAC

BAC 可用作单个安全 Web 应用程序，使用 Web 归档格式（.war）。它打算与 BluAge Gapwalk-Application 一起部署在 Apache Tomcat 应用程序服务器中，但也可以作为独立应用程序部署。如果存在 Gapwalk-Application 配置，BAC 将继承对Blusam存储的访问权限。

BAC 具有自己的专用配置文件，名为 `application-bac.yml`。有关配置详细信息，请参阅 the section called “BAC 专用配置文件”。

BAC 受到安全保护。有关安全配置的详细信息，请参阅the section called “为 BAC 配置安全性”。

BAC 专用配置文件

独立部署：如果 BAC 单独部署 Gapwalk-Application，则必须在 application-bac.yml 配置文件中配置与Blusam存储的连接。

必须在配置文件中设置用于浏览数据集记录的数据集配置的默认值。请参阅the section called “浏览数据集中的记录”。记录浏览页面可以使用可选的掩码机制，从而可以显示记录内容的结构化视图。使用掩码时，某些属性会影响记录视图。

以下可配置属性必须在配置文件中设置。BAC 应用程序不为这些属性假设任何默认值。

键	类型	描述
`bac.crud.limit`	整数	正整数值，提供浏览记录时返回的最大记录数。使用 `0` 意味着无限制。建议值：`10`（然后根据浏览页面上的数据集调整该值，以满足您的需求）。
`bac.crud.encoding`	字符串	默认字符集名称，用于将记录字节解码为字母数字内容。提供的字符集名称必须与 java 兼容（有关受支持的字符集，请参阅 java 文档）。建议值：在数据集源自的遗留平台上使用的遗留字符集；大多数情况下，这将是 EBCDIC 变体。

键	类型	描述
bac.crud.initCharacter	字符串	用于初始化数据项的默认字符（字节）。可以使用两个特殊值："LOW-VALUE" ，0x00 字节（建议值）和 "HI-VALUE" ，0xFF 字节。在应用掩码时使用。
bac.crud.defaultCharacter	字符串	默认字符（字节），作为单字符串，用于填充记录（在右侧）。建议值：" "（空白）。在应用掩码时使用。
bac.crud.blankCharacter	字符串	默认字符（字节），作为单字符串，用于表示记录中的空白。建议值：" "（空白）。在应用掩码时使用。
bac.crud.strictZoned	布尔值	一个标志，用于指示记录使用哪种分区模式。如果为 true，则使用严格区域模式；如果为 false，则使用修改后的分区模式。建议值：true。在应用掩码时使用。
bac.crud.decimalSeparator	字符串	在数值编辑字段中用作小数分隔符的字符（应用掩码时使用）。
bac.crud.currencySign	字符串	默认字符，作为单字符串，在应用格式设置时用于表示数字编辑字段中的货币（应用掩码时使用）。
bac.crud.pictureCurrencySign	字符串	默认字符，作为单字符串，用于表示数字编辑字段图片中的货币（应用掩码时使用）。

以下示例是一个配置文件片段。

```
bac.crud.limit: 10
bac.crud.encoding: ascii
bac.crud.initCharacter: "LOW-VALUE"
bac.crud.defaultCharacter: " "
bac.crud.blankCharacter: " "
bac.crud.strictZoned: true
bac.crud.decimalSeparator: "."
bac.crud.currencySign: "$"
bac.crud.pictureCurrencySign: "$"
```

为 BAC 配置安全性

为 BAC 配置安全性依赖于本文档页面中详述的机制。身份验证方案是 OAuth2，并提供了 Amazon Cognito 或 Keycloak 的配置详细信息。

虽然可以应用一般设置，但这里需要详细说明有关 BAC 的一些细节。对 BAC 特征的访问使用基于角色的策略进行保护，并且依赖于以下角色。

- ROLE_USER：
  - 基本用户角色
  - 不支持导入、导出、创建或删除数据集
  - 无法控制缓存策略
  - 不支持使用管理特征
- ROLE_ADMIN：
  - 继承 ROLE_USER 权限
  - 支持所有数据集操作
  - 支持使用管理缓存策略

安装掩码

在Blusam存储中，出于多功能性和性能考虑，数据集记录存储在数据库的字节数组列中。基于应用的视角，使用字段访问业务记录的结构化视图是 BAC 的一项便捷特征。这依赖于在 BluAge 驱动的现代化过程中生成的 SQL 掩码。

要生成要生成的 SQL 掩码，请确保将 BluInsights 转换中心配置中的相关选项 (`export.SQL.masks`) 设置为 true：

面具是现代化文物的一部分，可以从中下载 BluInsights给定项目。掩码是 SQL 脚本，由现代化程序组织，提供了数据集记录的应用视角。

例如，使用 AWS CardDemo 示例应用程序，您可以从该应用程序的现代化结果中下载的项目中找到 CBACT04C .cbl 程序的以下 SQL 掩码：

- cbact04c_fd_acctfile_rec.sql
- cbact04c_fd_discgrp_rec.sql
- cbact04c_fd_tran_cat_bal_record.sql
- cbact04c_fd_tranfile_rec.sql
- cbact04c_fd_xreffile_rec.sql

每个 SQL 掩码名称都是程序名称和程序中给定数据集的记录结构名称的串联。

以 [CBACT04C.cbl] 程序为例，给定的文件控制条目：

```
FILE-CONTROL.
    SELECT TCATBAL-FILE ASSIGN TO TCATBALF
           ORGANIZATION IS INDEXED
           ACCESS MODE  IS SEQUENTIAL
           RECORD KEY   IS FD-TRAN-CAT-KEY
           FILE STATUS  IS TCATBALF-STATUS.
```

与给定的 FD 记录定义关联

```
       FILE SECTION.
```

```
        FD  TCATBAL-FILE.
        01  FD-TRAN-CAT-BAL-RECORD.
            05 FD-TRAN-CAT-KEY.
               10 FD-TRANCAT-ACCT-ID             PIC 9(11).
               10 FD-TRANCAT-TYPE-CD             PIC X(02).
               10 FD-TRANCAT-CD                  PIC 9(04).
            05 FD-FD-TRAN-CAT-DATA               PIC X(33).
```

名为 `cbact04c_fd_tran_cat_bal_record.SQL` 的匹配 SQL 掩码给出了 CBACT04C.cbl 程序在名为 `FD-TRAN-CAT-BAL-RECORD` 的 FD 记录上的视角。

其内容是：

```
-- Generated by Blu Age Velocity
-- Mask : cbact04c_fd_tran_cat_bal_record

INSERT INTO mask (name, length) VALUES ('cbact04c_fd_tran_cat_bal_record', 50);
  INSERT INTO mask_item (name, c_offset, length, skip, type, options, mask_fk) VALUES
 ('fd_trancat_acct_id', 1, 11, false, 'zoned', 'integerSize=11!fractionalSize=0!
signed=false', (SELECT MAX(id) FROM mask));
  INSERT INTO mask_item (name, c_offset, length, skip, type, options, mask_fk) VALUES
 ('fd_trancat_type_cd', 12, 2, false, 'alphanumeric', 'length=2', (SELECT MAX(id) FROM
 mask));
  INSERT INTO mask_item (name, c_offset, length, skip, type, options, mask_fk)
 VALUES ('fd_trancat_cd', 14, 4, false, 'zoned', 'integerSize=4!fractionalSize=0!
signed=false', (SELECT MAX(id) FROM mask));
  INSERT INTO mask_item (name, c_offset, length, skip, type, options, mask_fk) VALUES
 ('fd_fd_tran_cat_data', 18, 33, false, 'alphanumeric', 'length=33', (SELECT MAX(id)
 FROM mask));
```

口罩使用两个表Blusam存储在存储器中：

- mask：用于识别掩码。mask 表包含以下列：
  - name：用于存储掩码标识（用作主键，因此必须是唯一的）
  - length：记录掩码的大小（以字节为单位）
- mask_item：用于存储掩码详细信息。FD 记录定义中的每个基本字段都将在 mask_item 表中生成一行，其中包含了有关如何解读给定记录部分的详细信息。mask_item 表包含以下列：
  - name：记录字段的名称，以基本名称为基础，使用小写并用下划线取代连字符
  - c_offset：记录子部分基于 1 的偏移量，用于字段内容
  - length：记录子部分的长度（以字节为单位），用于字段内容

- skip：一个标志，用于在视图表示中指示是否应该跳过给定记录部分
- type：字段类型（基于其遗留 picture 子句）
- options：其他类型选项，取决于 type 列
- mask_fk：指代要将此项目附加到的掩码标识符

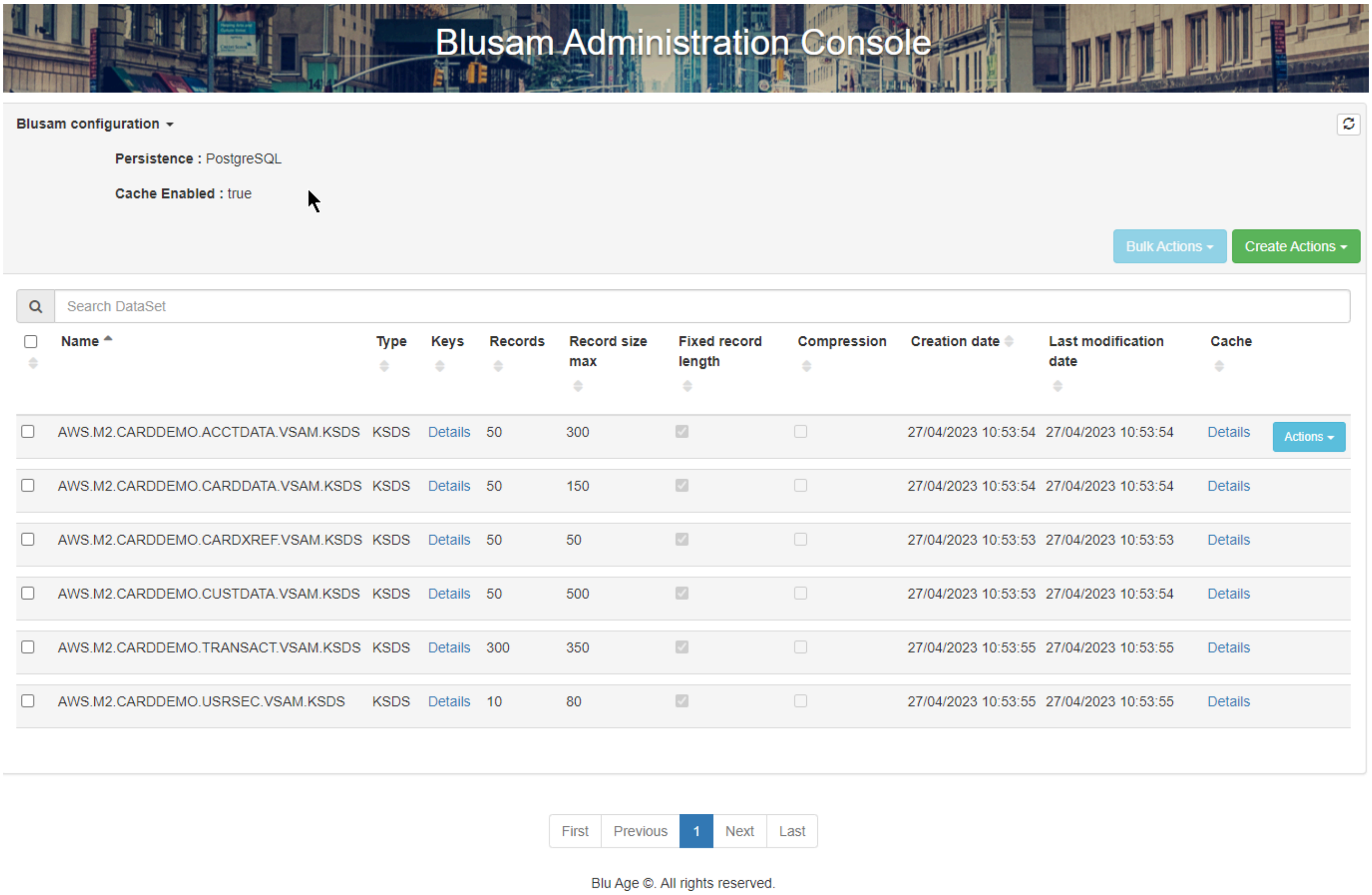
请注意以下几点：

- SQL 掩码代表了程序对数据集记录的视角：几个程序可能对给定的数据集有不同的视角；请仅安装您认为与您的目的相关的掩码。
- 除 FD 记录以外，SQL 掩码还可以代表程序基于 01 数据结构（来自 WORKING STORAGE 部分）的视角。SQL 掩码根据其性质组织成子文件夹：
  - 基于 FD 记录的掩码将位于名为 `file` 的子文件夹中
  - 基于 01 数据结构的掩码将位于名为 `working` 的子文件夹中

  虽然 FD 记录定义始终与数据集中的记录内容相匹配，但 01 数据结构可能并不如此，或者可能仅代表数据集记录中的子集。在使用它们之前，请检查代码并了解可能存在的缺点。

使用 BAC

由于 BAC 受到安全保护，并且根据用户角色提供使用特征的权限，因此访问应用程序的第一步就是对自己进行身份验证。完成身份验证步骤后，您将被重定向到主页。主页显示了Blusam存储中找到的数据集的分页列表：



要返回包含数据集列表的主页，请选择应用程序任意页面左上角的Blu Age徽标。下图显示了该徽标。

BLU AGE

标有 “Blusam配置” 的可折叠标题包含有关所用Blusam存储配置的信息：

- `Persistence`: 永久存储引擎 (PostgreSQL)
- `Cache Enabled`：是否启用了存储缓存

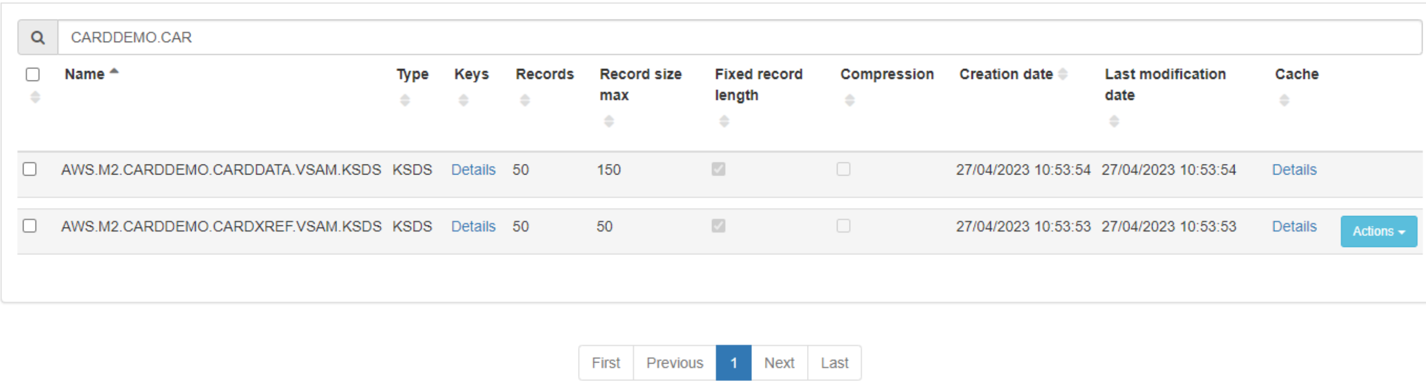
标题右侧有两个下拉列表，每个下拉列表列出了与数据集相关的操作：

- 批量操作
- 创建操作

要了解这些列表的详细内容，请参阅the section called “现有的数据集操作”。

如果未选择任何数据集，则批量操作按钮将处于禁用状态。

您可以使用搜索字段根据数据集名称对列表进行筛选：



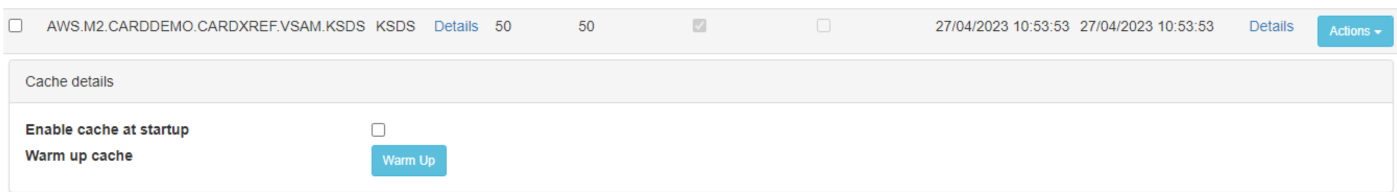
下面的分页列表显示表格的每一行对应一个数据集，其中包含以下几列：

- 选择复选框：此复选框用于选择当前数据集。
- 名称：数据集的名称。
- 类型：数据集的类型，为以下类型之一：
  - KSDS
  - ESDS
  - RRDS
- 键：用于显示或隐藏有关键的详细信息（如果有）的链接。例如，给定的 KSDS 具有必需的主键和一个备用键。

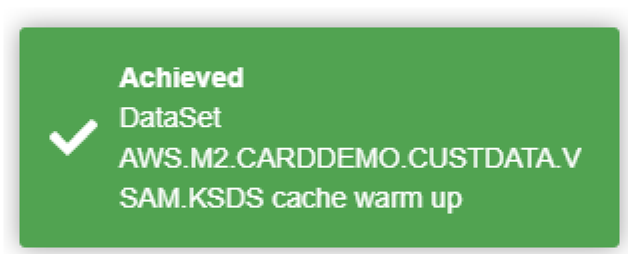
每个键对应一行，包含以下几列。所有字段都不可编辑。

- 键性质：要么为主键，要么为备用键
- 名称：键的名称
- 唯一性：键是否接受重复条目
- 偏移量：自键在记录中的开头处的偏移量
- 长度：记录中键部分的长度（以字节为单位）

- 记录数：数据集中的记录总数。
- 最大记录大小：记录的最大大小，以字节表示。
- 固定记录长度：一个复选框，用于指示记录是固定长度（选中）还是可变长度（未选中）。
- 压缩：一个复选框，用于指示是（选中）否（未选中）对存储的索引应用压缩。
- 创建日期：在Blusam存储中创建数据集的日期。
- 上次修改日期：上次在Blusam存储中更新数据集的日期。
- 缓存：用于显示或隐藏有关应用于此数据集的缓存策略的详细信息的链接。



  - 启动时启用缓存：此复选框用于为此数据集指定启动缓存策略。如果选中此项，则数据集将在启动时加载到缓存中。
  - 预热缓存：一个按钮，用于将给定的数据集加载到缓存中，会立即开始加载（但是注入缓存需要一些时间，具体取决于数据集的大小和键的数量）。将数据集加载到缓存中后，会出现如下所示的通知。

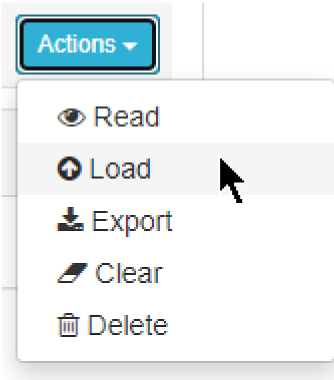


- 操作：可能的数据集操作的下拉列表。有关更多信息，请参阅 the section called "现有的数据集操作"。

在页面底部，有一个常规的分页导航小组件，用于浏览数据集列表的页面。

现有的数据集操作

对于分页列表中的每个数据集，都有一个操作下拉列表，其中包含以下内容：



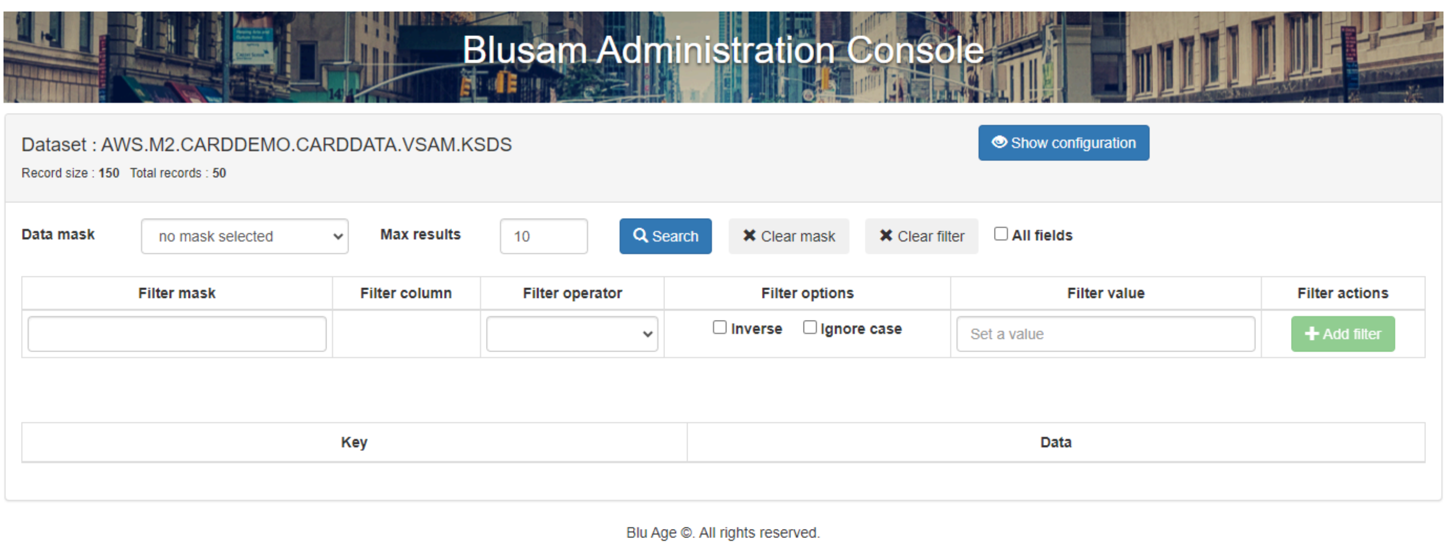
列表中的每个项目都是一个活动链接，可用于对数据集执行指定操作：

- 读取：浏览数据集中的记录
- 加载：从遗留数据集文件导入记录
- 导出：将记录导出到平面文件（与遗留系统兼容）
- 清除：从数据集中删除所有记录
- 删除：从存储中删除数据集

以下各节介绍了每个操作的详细信息。

浏览数据集中的记录

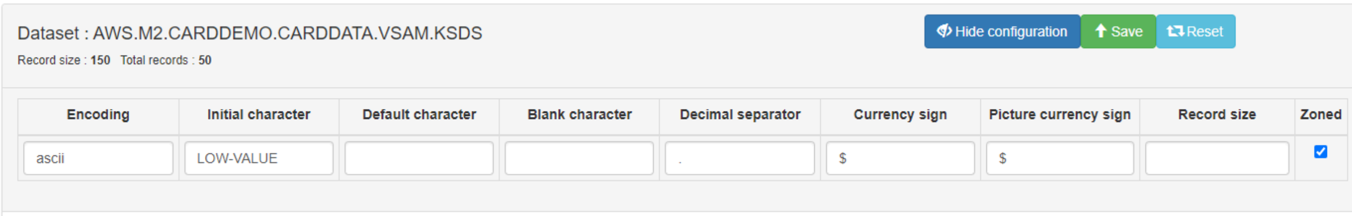
当您为给定数据集选择读取操作时，您会看到以下页面。



该页面由以下部分组成：

- 标题，包含：

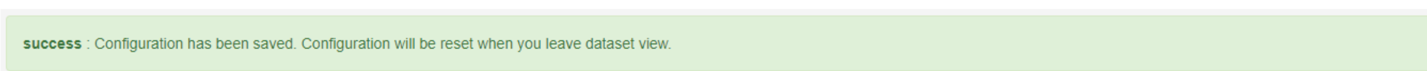
- 数据集：数据集名称
- 记录大小：固定记录长度，以字节为单位
- 记录总数：为此数据集存储的记录总数
- 显示配置按钮（右侧）：数据集配置 show/hide 的切换按钮。开始，配置是隐藏的。使用该按钮时，您将看到配置，如下图中所示。

  显示配置时，有两个新按钮："保存"和"重置"，分别用于：

  - 保存此数据集和当前工作会话的配置
  - 将所有字段的配置重置为默认值。
- 可配置属性的列表，这些属性用于为给定数据集量身定制浏览体验。

这些可配置属性与 the section called "BAC 专用配置文件"中所述的配置属性相匹配。要了解每列的含义和适用的值，请参阅该部分内容。可以在此处为数据集重新定义每个值，并保存用于工作会话（使用"保存"按钮）。保存配置后，会出现类似于下图所示的横幅。



该横幅说明工作会话会在您离开当前页面时结束。

还有一个额外的可配置属性没有在配置部分中载明：记录大小。该属性用于指定给定的记录大小（以字节为单位），该大小将用于筛选适用于该数据集的掩码：只有总长度与给定记录大小匹配的掩码才会列在"数据掩码"下拉列表中。

由"搜索"按钮触发从数据集中检索记录，会使用附近的所有选项和筛选条件。

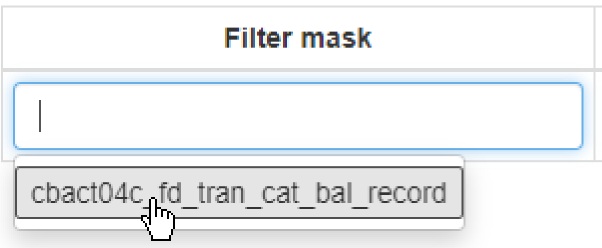
选项的第一行：

- "数据掩码"下拉列表显示适用的掩码（根据记录大小）。请注意，光记录大小匹配还不足以使其成为有效的适用掩码。掩码定义还必须与记录内容兼容。这里选择的数据掩码具有
- 最大结果数：限制搜索检索的记录数。设置为 0，表示对于从数据集获得的（分页）结果数没有限制。
- "搜索"按钮：使用筛选条件和选项启动记录检索

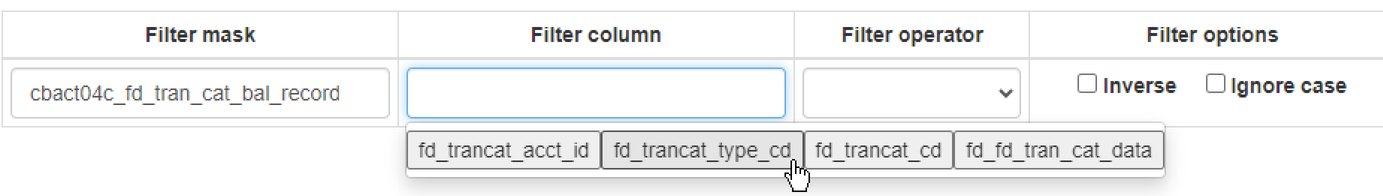
- “清除蒙版” 按钮：将清除使用的蒙版（如果有），然后将结果页面切换回原始 key/data 演示文稿。
- “清除筛选条件”按钮：将清除所使用的筛选条件（如果有），并相应地更新结果页面。
- 切换所有字段：选中后，用 `skip = true` 定义的掩码项会始终显示，否则用 `skip = true` 定义的掩码项将被隐藏。

下面几行筛选条件：可以根据应用于给定掩码的字段（列）的筛选条件的使用，定义筛选条件列表，如下图所示。

- 筛选掩码：要从中选取筛选列的掩码的名称。当您选择该字段时，将显示适用掩码的列表。您可以从该列表中选择想要的掩码。

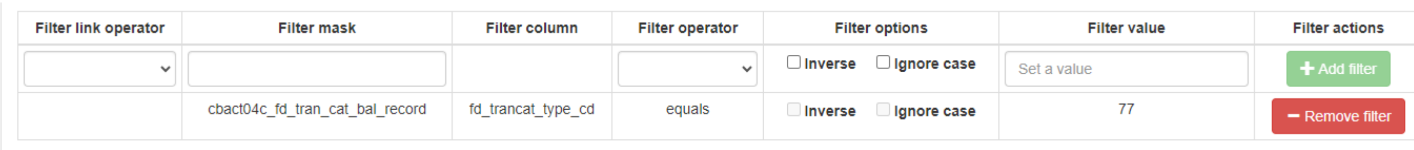


- 筛选列：掩码中用于筛选记录的字段（列）的名称。当您选择该字段时，将显示掩码列的列表。要填写筛选列字段，请选择所需的单元格。



- 筛选运算符：要应用于所选列的运算符。可用的运算符如下：
  - equals to：记录的列值必须等于筛选条件值
  - starts with：记录的列值必须以筛选条件值开头
  - ends with：记录的列值必须以筛选条件值结尾
  - contains：记录的列值必须包含筛选条件值
- 筛选条件选项：
  - 反向：应用筛选运算符的反向条件；例如，将“等于”替换为“不等于”；
  - 忽略大小写：在筛选运算符的字母数字比较中忽略大小写
- 筛选条件值：筛选运算符用于与筛选列进行比较的值。

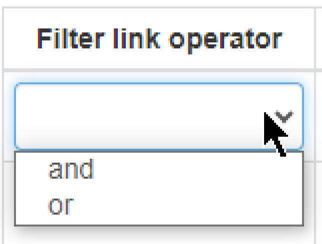
设置了最少数量的筛选项（至少必须设置筛选掩码、筛选列、筛选运算符和筛选条件值）后，“添加筛选条件”按钮就会启用，单击该按钮会对检索到的记录创建新的筛选条件。在顶部会增加另一个空的筛选条件行，添加的筛选条件带有一个“删除筛选条件”按钮，可用于抑制给定的筛选条件：



启动搜索，筛选后的结果便会显示在分页表中。

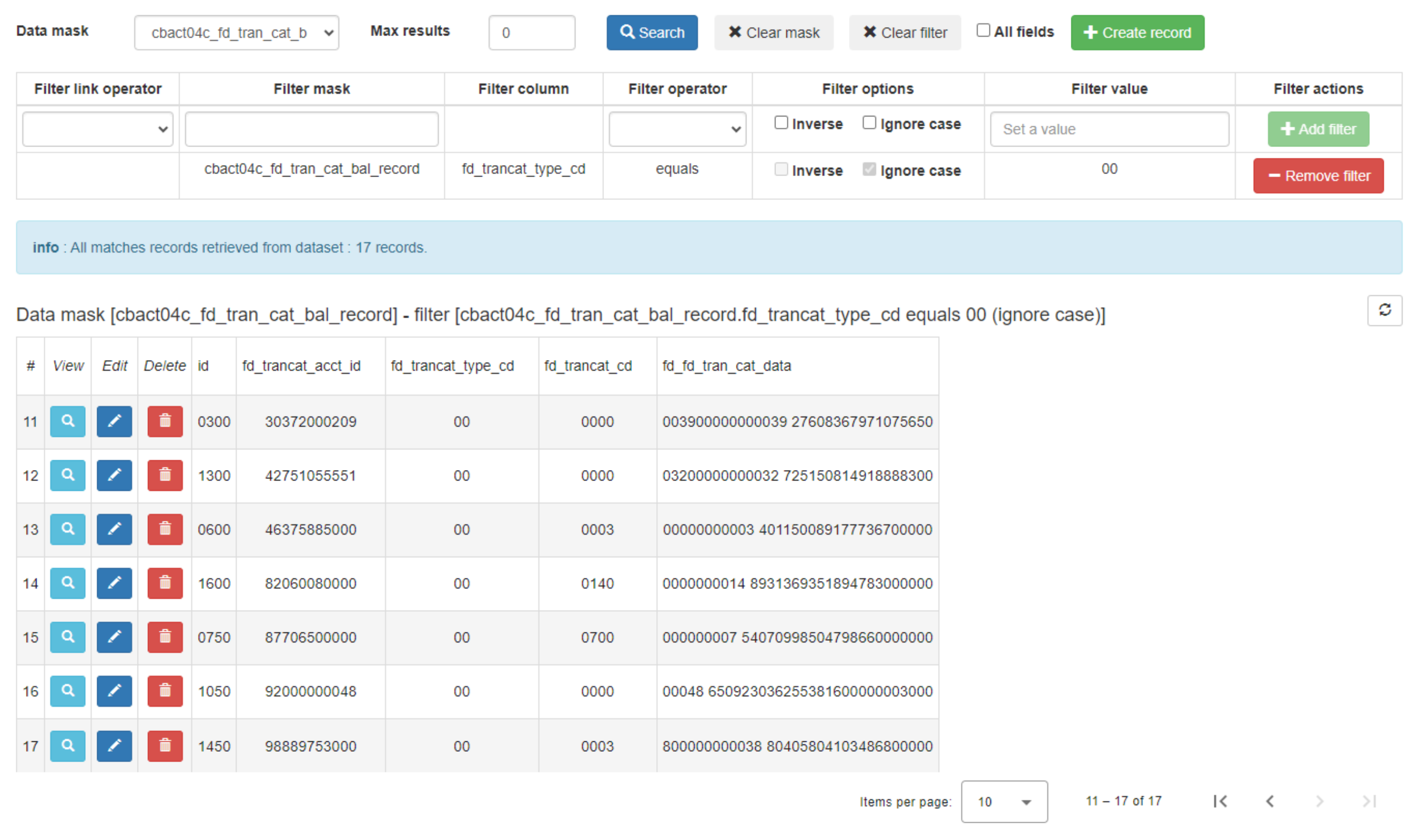
注意

- 连续的筛选条件通过 and 或 or 进行连接。每个新的筛选条件定义都从设置连接运算符开始，如下图中所示。



- 可能没有任何记录与给定的筛选条件相匹配。

否则，结果表将如下图所示。



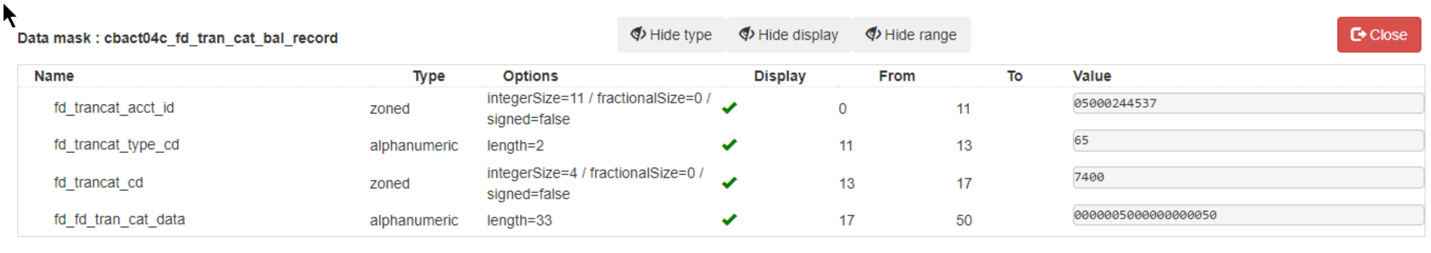
标题会指示符合筛选条件的记录总数。在标题后面，您将看到以下内容。

- 所使用的数据掩码（如果有）和筛选条件的提醒。
- 一个刷新按钮，您可以使用它来触发整个结果表的刷新，其中包含Blusam存储中的最新值（例如，它可能已被其他用户更新）。

对于检索到的每条记录，表中都有一行显示对记录内容应用数据掩码的结果。每列都是根据列类型（并使用所选编码）对记录子部分的解读。每行左侧有三个按钮：

- 放大镜按钮：指向显示详细记录内容的专用页面
- 钢笔按钮：指向记录内容的专用编辑页面：
- 垃圾桶按钮：用于从 blusam 存储空间中删除给定记录

查看记录的详细内容：



- 用于隐藏或显示某些列的三个切换按钮：
  - 隐藏/显示类型
  - 隐藏/显示“显示标志”
  - 隐藏/显示范围
- 要离开此专用页面并返回结果表，请选择关闭。
- 每行代表数据掩码中的一列，其中包含以下几列：
  - 名称：列的名称
  - 类型：列的类型
  - 显示：显示指示器；如果匹配的掩码项使用 `skip = false` 定义，则会显示绿色的对勾符号，否则将会显示红色叉号
  - 从和到：记录子部分基于 0 的范围
  - 值：记录子部分的解读值，使用类型和编码

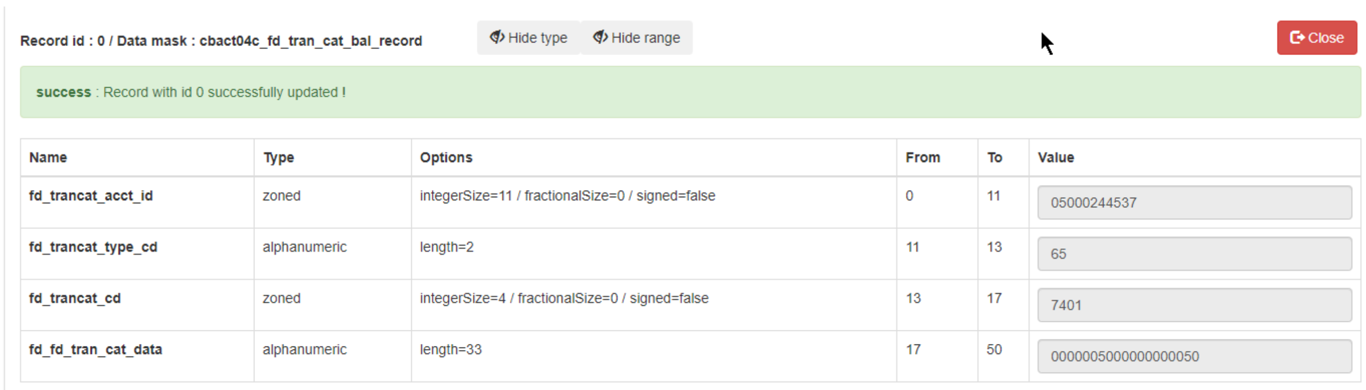
编辑记录的内容：

编辑页类似于上面所述的视图页，不同之处在于掩码项值是可编辑的。三个按钮控制更新过程：

- 重置：将可编辑值重置为初始记录值（在进行任何编辑之前）；
- 验证：根据掩码项类型验证输入。对于每个掩码项，验证结果都将使用可视标签打印出来（如果验证成功，则显示 OK 和复选框；如果验证失败，则显示 ERROR 和红色叉号，同时将显示一条提示验证失败的错误消息）。如果验证成功，将显示两个新按钮：

- 保存：尝试将现有记录更新到Blusam存储中
- 保存副本：尝试在Blusam存储中创建新记录

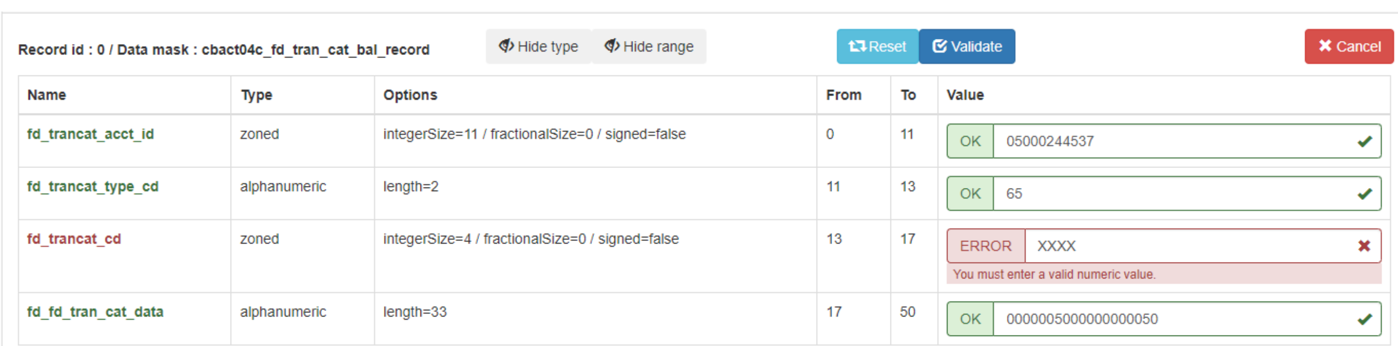
- 如果成功将记录保存到存储空间中，则会显示一条消息，并且页面将切换到只读模式（无法再编辑掩码项的值）：



- 如果出于任何原因将记录保留到存储空间失败，则会以红色显示一条错误消息，提示失败原因。最常见的失败原因是存储记录会导致键损坏（键无效或重复）。有关说明，请见以下“注意”。
- 要退出，请选择关闭按钮。
- 取消：结束编辑会话，关闭页面，然后返回到记录列表页面。

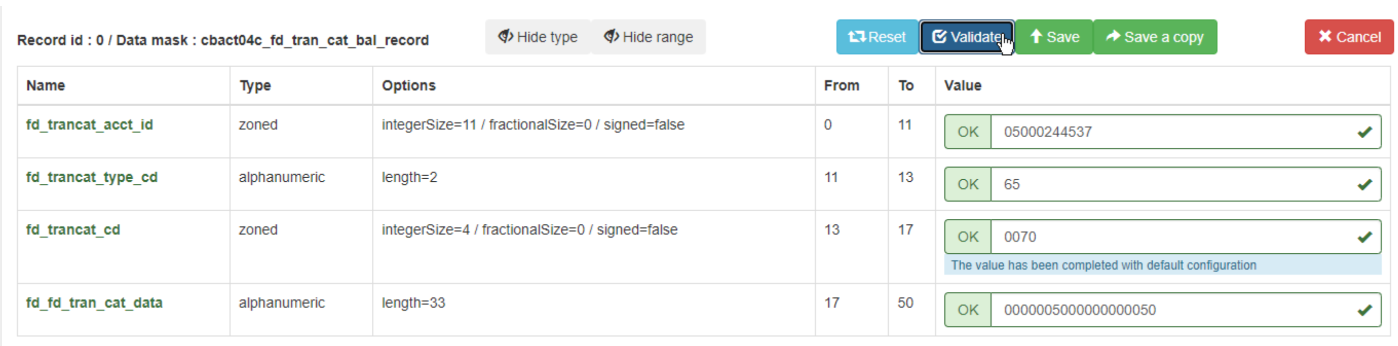
注意：

- 验证机制仅检查掩码项值是否与掩码项类型在形式上兼容。例如，请查看以下数值掩码项验证失败：

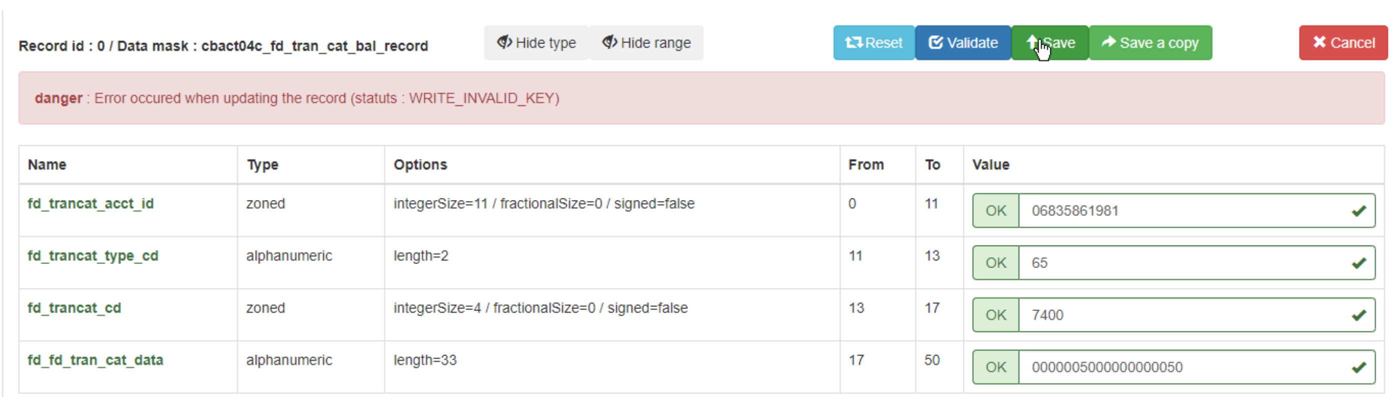


- 验证机制可能会尝试自动更正无效的输入，并显示一条蓝色的信息性消息，指出该值已根据其类型自动更正。例如，在 `fd_trncat_cd` 数值掩码项中输入 7XX0 作为数值：

调用验证会导致以下结果：

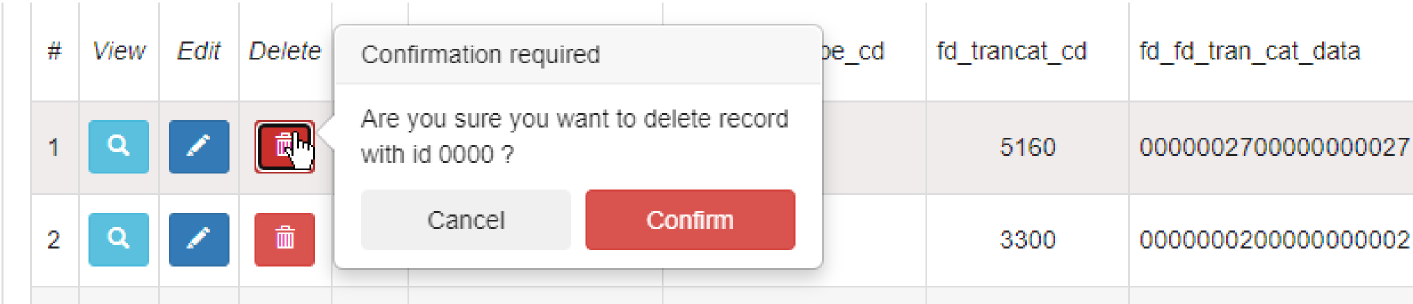


- 验证机制不检查给定值在键完整性方面是否有效（如果涉及给定数据集的任何唯一键）。例如，尽管验证成功，但如果提供的值导致键无效或重复的情况，则持久化将失败并显示错误消息：



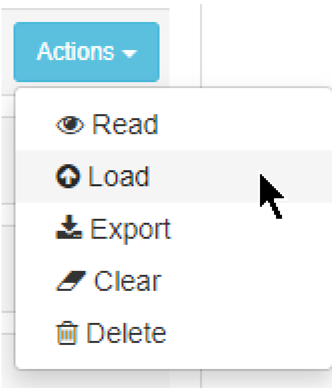
删除记录：

要删除记录，请选择垃圾桶按钮：

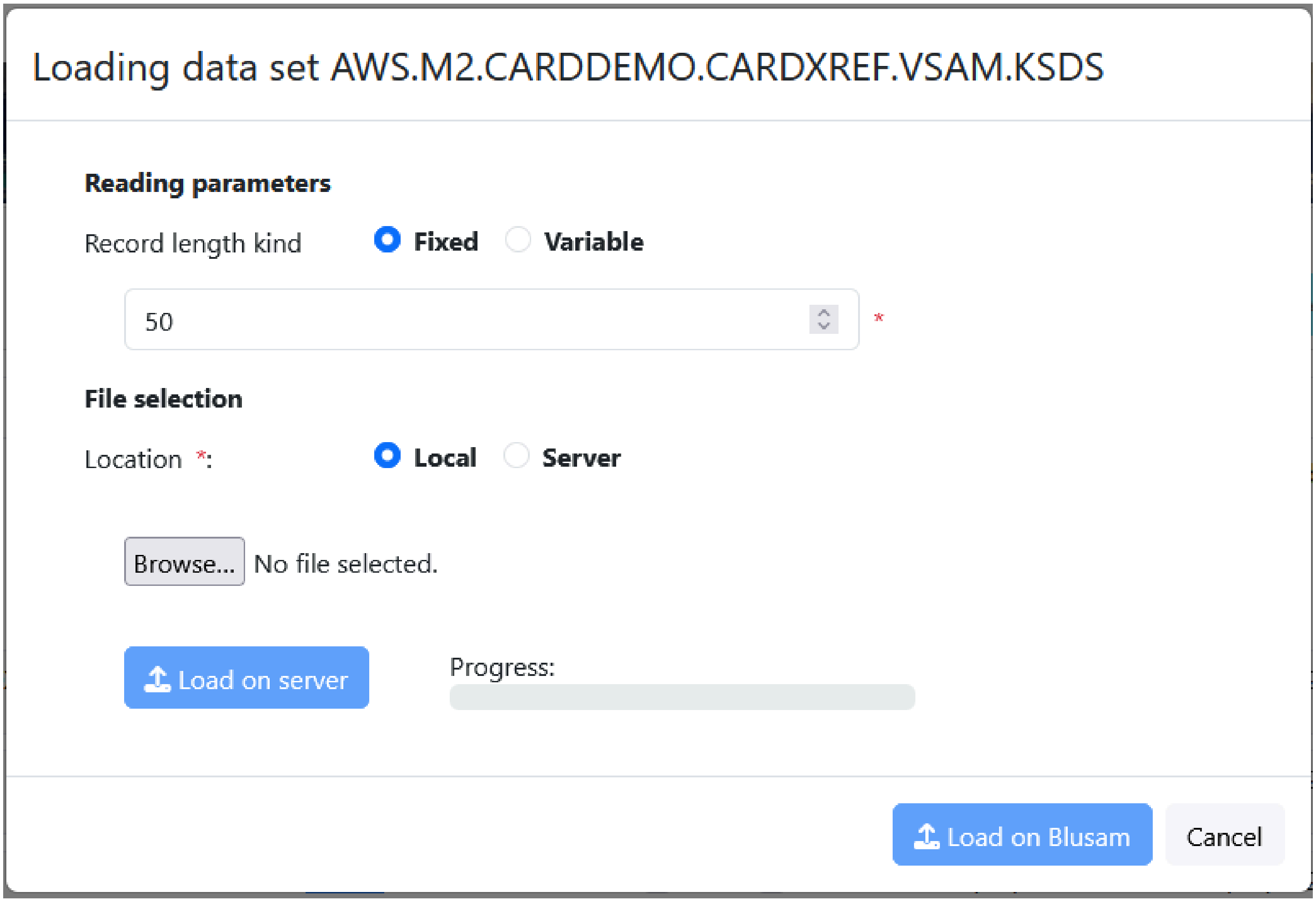


将记录加载到数据集中

要将记录加载到数据集中，请选择操作，然后选择加载。



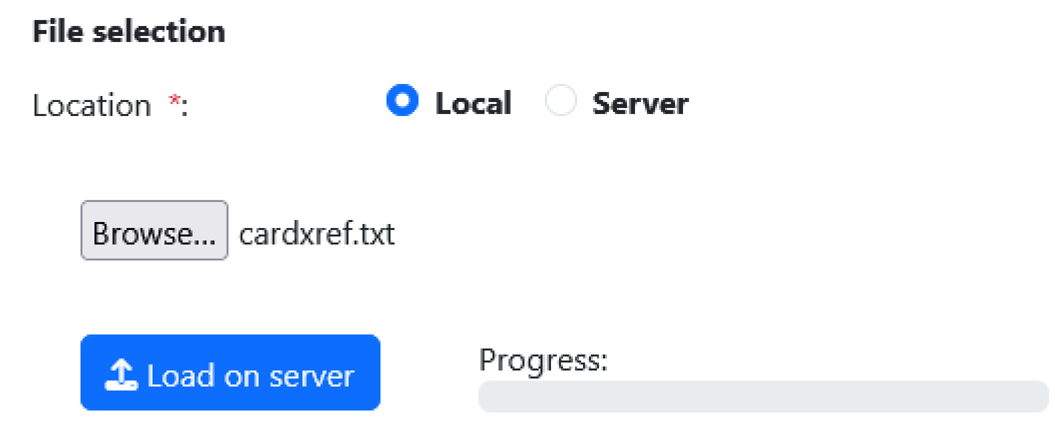
此时会出现一个带有加载选项的窗口。



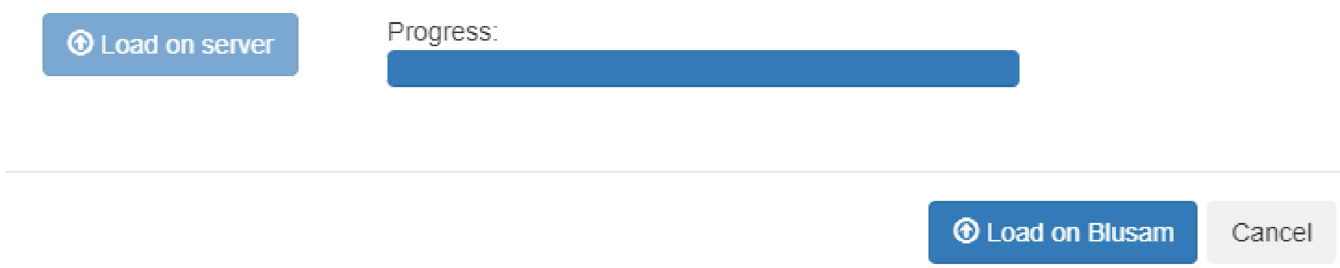
一开始，加载到服务器和加载到 Blusam 按钮都处于禁用状态。

读取参数：

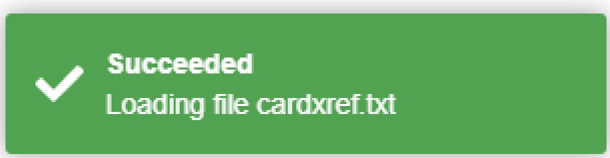
- 记录长度类型：
  - 固定或可变记录长度：使用单选按钮指定遗留数据集导出使用固定长度记录还是可变长度记录（记录应以 RDW 字节开头）。如果选择“固定”，则必须在输入字段中将记录长度（以字节为单位）指定为正整数值。该值应由来自数据集的信息预填充。如果选择“可变”，则给定的输入字段将消失。
  - 文件选择：
    - 本地：使用下面的文件选择器从本地计算机中选择数据集文件（注意：文件选择器使用浏览器的区域设置来显示其消息——这里为法语，但在您本地可能会不同，这是正常情况）。进行选择后，将使用数据文件名更新窗口，并且加载到服务器按钮会启用：



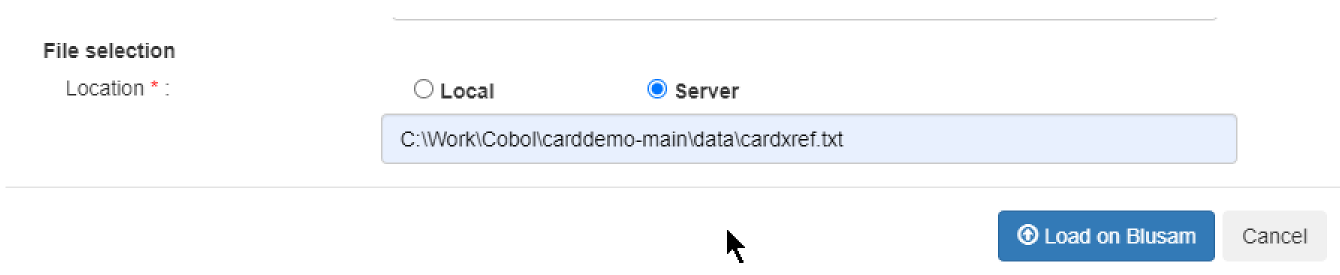
选择加载到服务器。待进度条结束后，加载到 Blusam 按钮会启用：



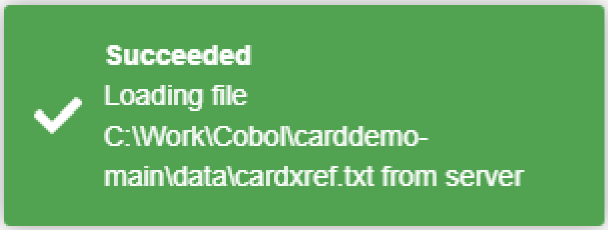
要完成Blusam存储的加载过程，请选择“在 Blusam 上加载”。否则，选择取消。如果选择继续进行加载过程，则在加载过程完成后，右下角会显示一条通知：



- 服务器：如果选择此选项，待加载到服务器按钮消失时，会显示一个输入字段。在输入字段中，您必须指定Blusam服务器上数据集文件的路径（这假设您已先将给定文件传输到Blusam服务器）。指定该路径后，加载到 Blusam 会启用。

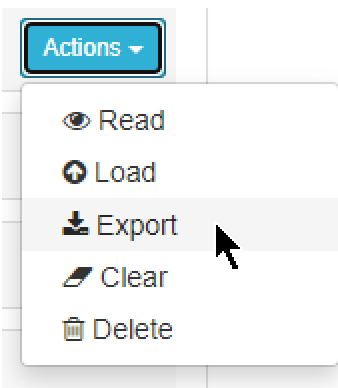


要完成加载过程，请选择加载到 Blusam。否则，选择取消。如果选择继续进行加载，则在加载过程完成后会显示一条通知。该通知与从浏览器加载的通知不同，因为它显示的是数据文件在服务器上的路径，后跟 from server：

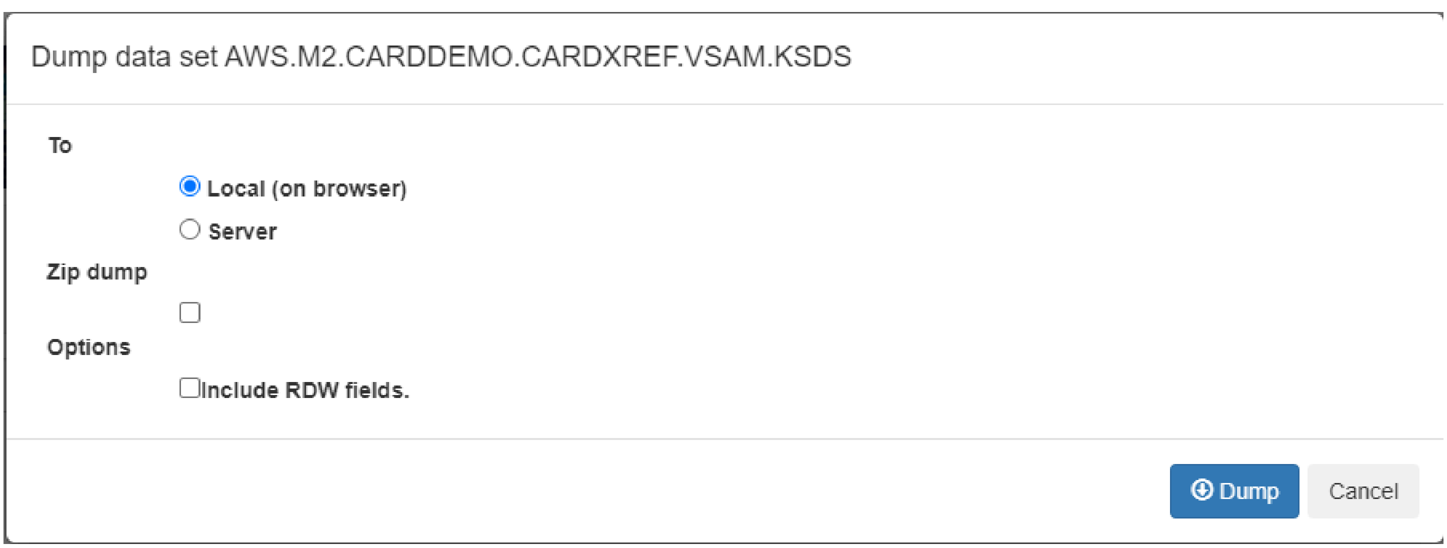


从数据集导出记录

要导出数据集记录，请在当前数据集行中选择操作，然后选择导出：



此时会出现以下弹出窗口。



选项：

到：一个单选按钮选项，用于选择导出目的地，可以在浏览器中下载 [本地（在浏览器上）]，也可以下载到托管 BAC 应用程序的服务器上的给定文件夹。如果选择使用服务器选项进行导出，则会显示一个新的输入字段：

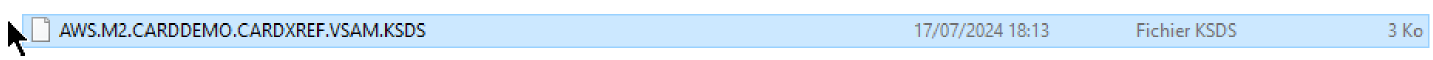
正如该输入字段右侧的红色星号所指示，必须提供服务器上的有效文件夹位置（如果未提供文件夹位置，则“转储”按钮将处于非活动状态）。

如果要导出到服务器，并且打算在导出后操作导出的数据集文件，则必须对服务器文件系统拥有足够的访问权限。

Zip 转储：一个复选框，用于生成压缩归档文件而不是原始文件。

选项：对于可变长度记录数据集，如果要在导出数据集中的每条记录的开头包含记录描述字（RDW），请选择包含 RDW 字段。

要启动数据集导出过程，请选择转储。如果选择导出到浏览器，请检查导出数据集文件的下载文件夹。此文件将与数据集同名：

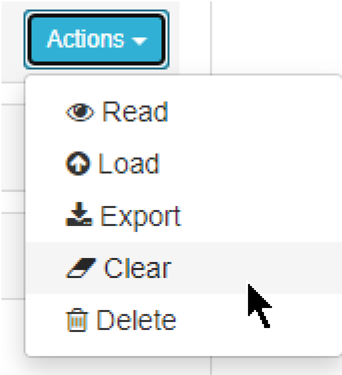


注意：

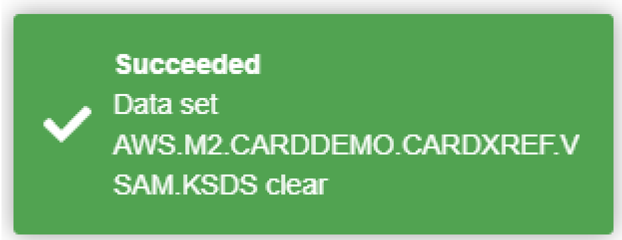
- 对于 KSDS，记录将按照主键顺序导出。
- 对于 ESDS 和 RRDS，记录将按照 RBA（相对字节地址）顺序导出。
- 对于所有类型的数据集，记录都将导出为原始二进制数组（不进行任何类型的转换），从而确保与遗留平台的直接兼容。

清除数据集中的记录

要清除数据集中的所有记录，请选择操作，然后选择清除：

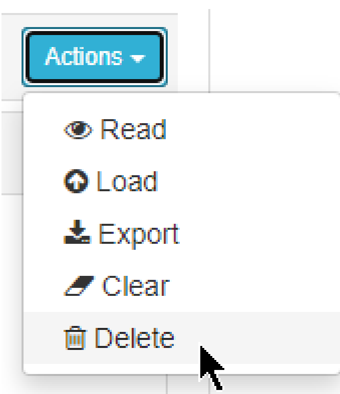


从数据集中删除所有记录后，将显示以下通知。

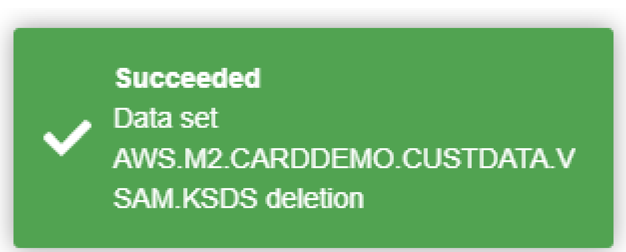


删除数据集

要删除数据集，请选择操作，然后选择删除：



删除数据集后，会显示以下通知：

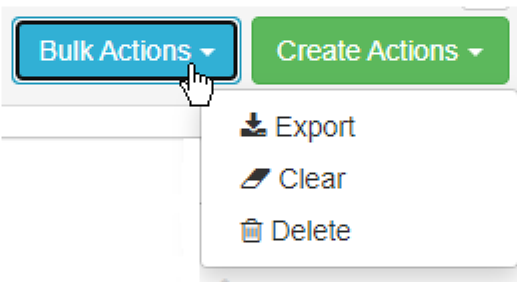


批量操作

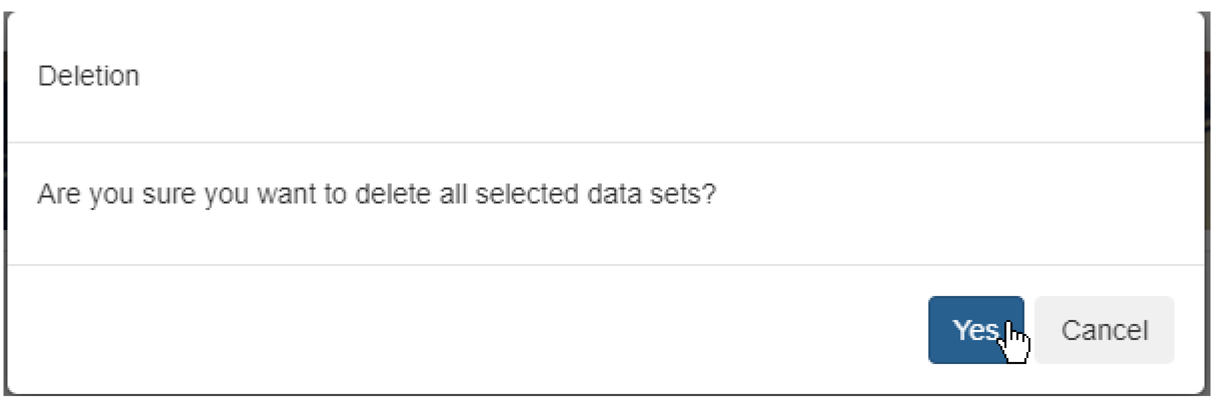
对于数据集有三种可用的批量操作：

- 导出
- 清除
- 删除

批量操作只能应用于选定的数据集（至少需要选择一个数据集）；可通过在数据集列表中勾选数据集行左侧的选择复选框来选择数据集。至少选择一个数据集后就会启用“批量操作”下拉列表：



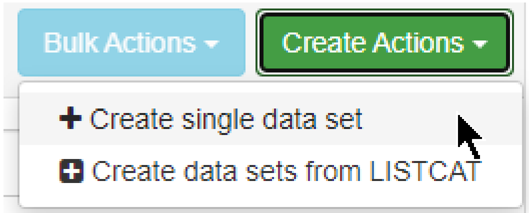
给定的操作将应用于选定的数据集而不是单个数据集，除此之外，这些操作与上面介绍的操作类似。有关详细信息，请参阅专门的操作文档。弹出窗口的文本内容将略有不同，以反映批量操作的性质。例如，当尝试删除多个数据集时，弹出窗口的内容如下：



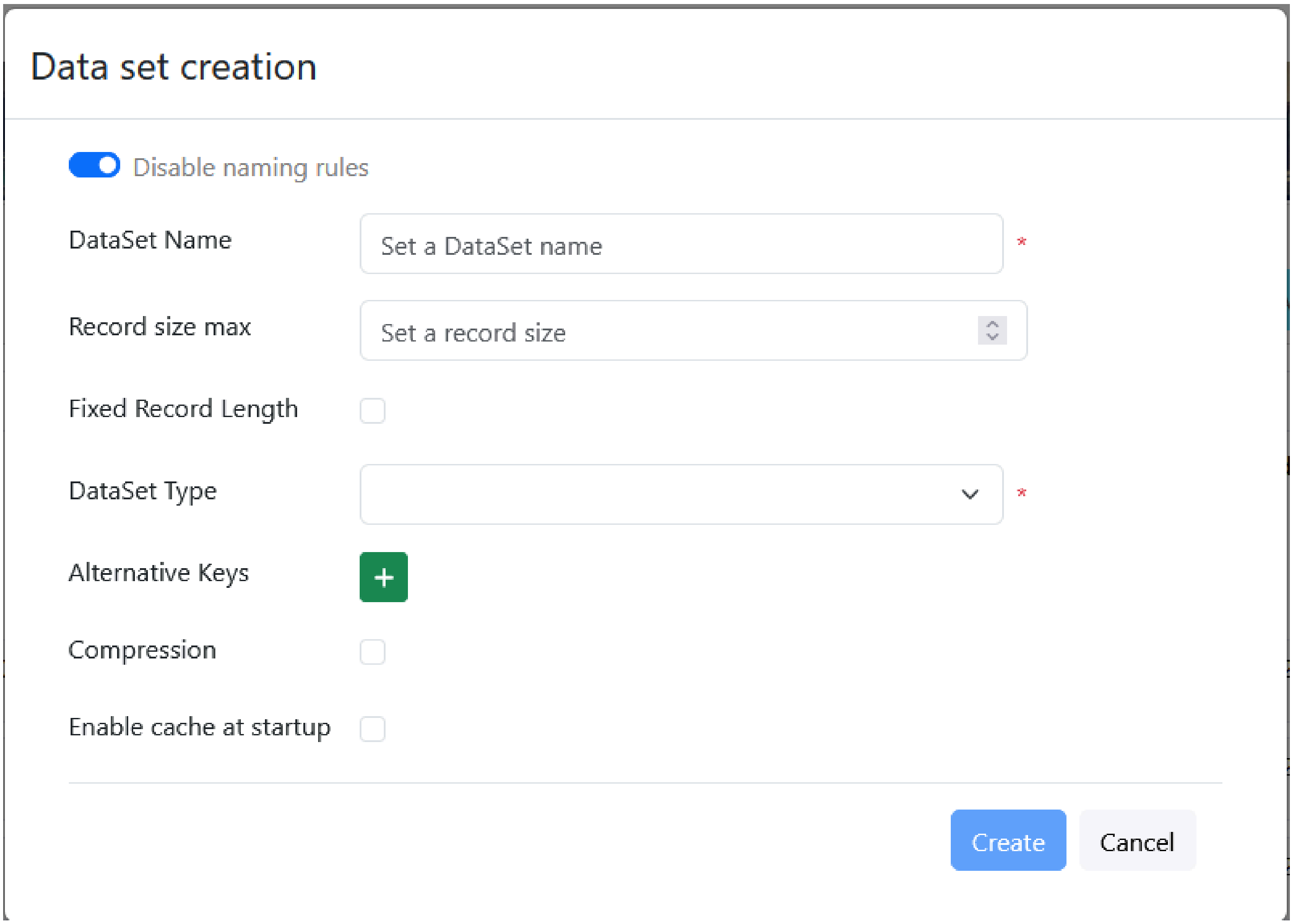
创建操作

创建单个数据集

选择操作，然后选择创建单个数据集：

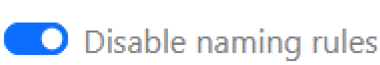


然后，数据集创建表单将以弹出窗口的形式显示：

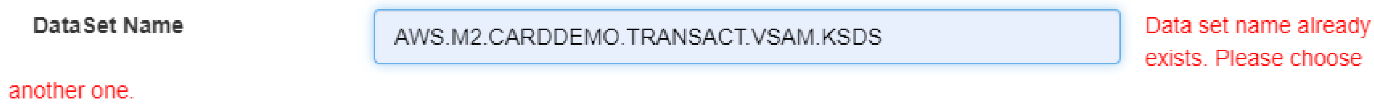


您可以为数据集定义指定以下属性：

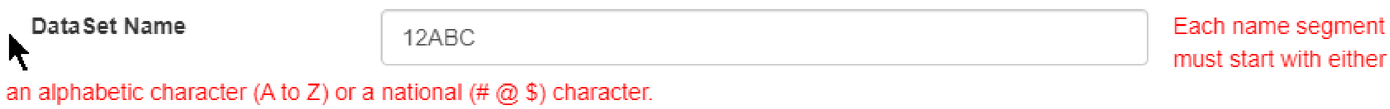
- 启用和禁用命名规则：使用"禁用命名规则/启用命名规则"切换小组件来禁用和启用数据集命名约定。建议您将切换开关保留为默认值，即处于启用数据集命名规则的状态（切换小组件应显示"禁用命名规则"）：

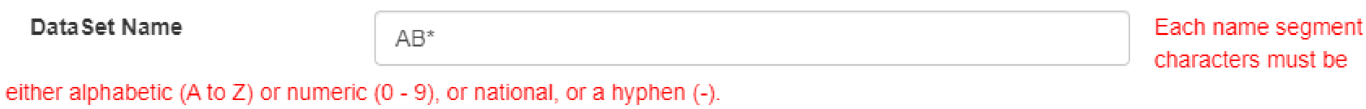
  

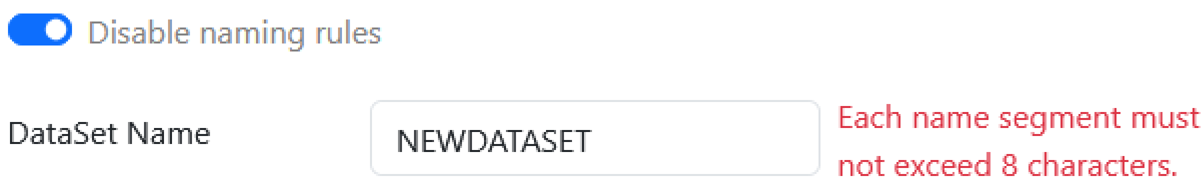
- 数据集名称：数据集的名称。如果指定的名称已在使用，则会显示以下错误消息。

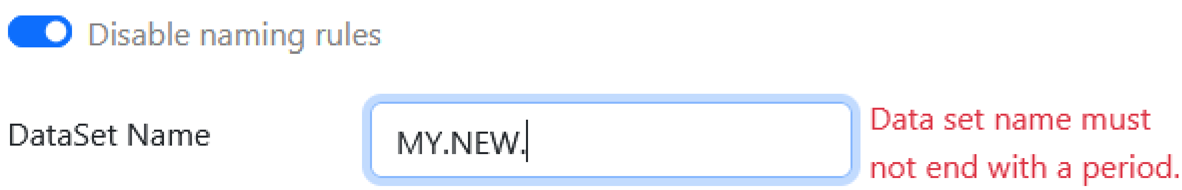


如果启用了命名约定，则名称还必须遵守命名约定：





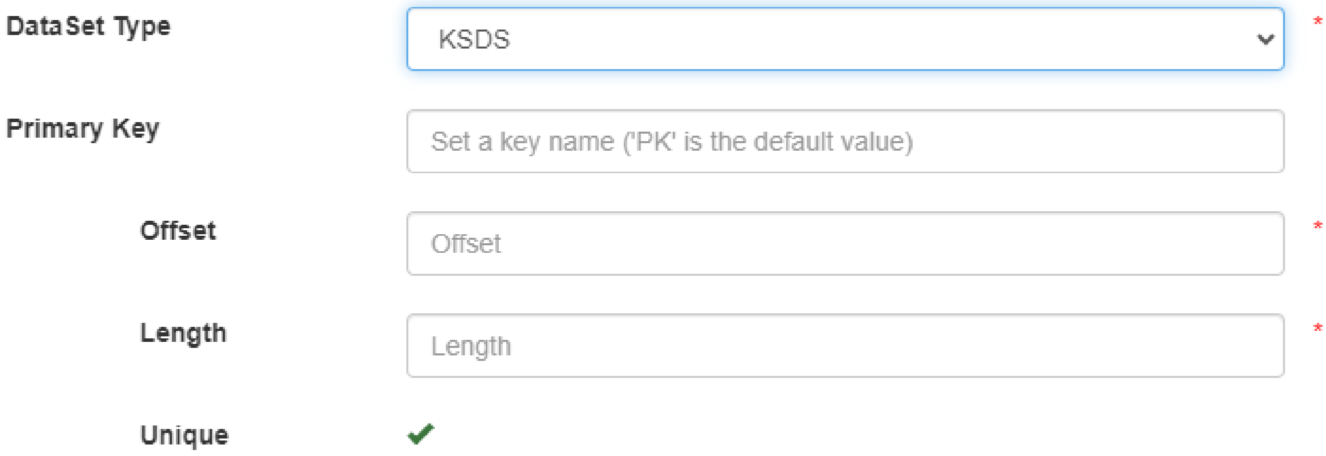




- 最大记录大小：这必须是一个正整数，表示具有固定长度记录的数据集的记录大小。对于具有可变长度记录的数据集，可以将其留空。
- 固定长度记录：此复选框用于指定记录长度为固定长度还是可变长度。如果选中，则数据集将具有固定长度记录，否则记录长度将可变。

  将遗留数据导入可变长度记录数据集时，提供的遗留记录必须包含提供每条记录的长度的记录描述字（RDW）。
- 数据集类型：用于指定当前数据集类型的下拉列表。支持以下类型：
  - ESDS
  - LargeESDS
  - KSDS

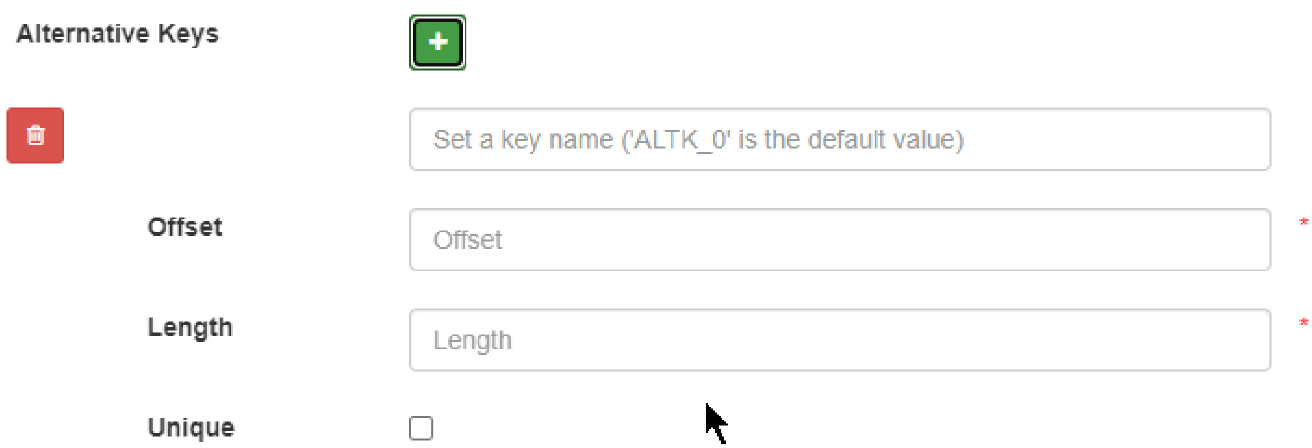
  对于 KSDS，必须指定主键：



对于主键，请指定以下内容：

- 名称：该字段是可选的。默认值为 **PK**。
- 偏移量：主键在记录中基于 0 的偏移量。该值必须为正整数。该字段为必填。
- 长度：主键的长度。此值必须为正整数。该字段为必填。

对于 KSDS 和 ESDS，您可以通过选择“备用键”标签前面的加号按钮来定义一组备用键。每次选择该按钮时，数据集创建表单中都会出现一个新的备用键定义部分：



对于每个备用键，您都需要提供：

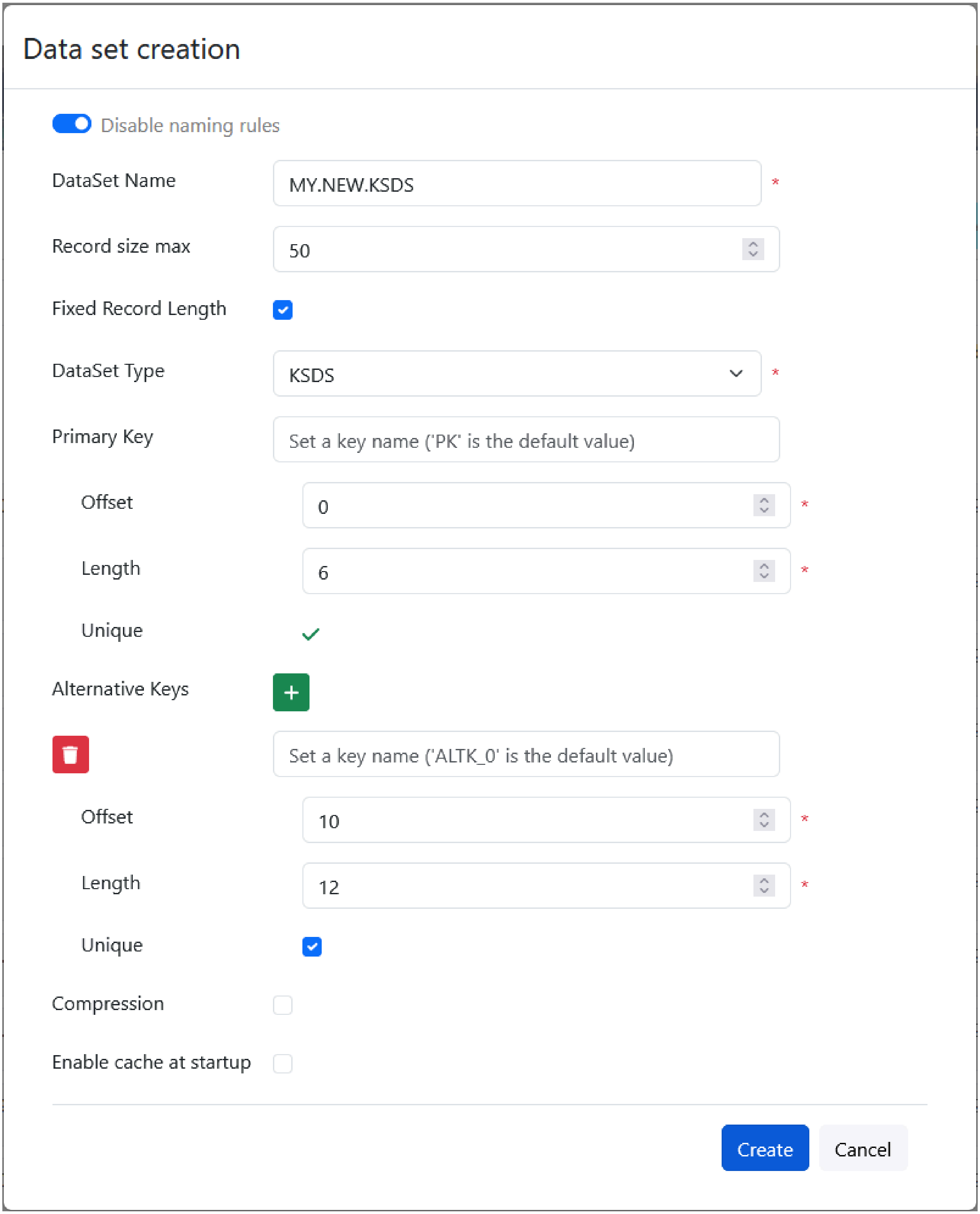
- 名称：该字段是可选的。默认值为 **ALTK_#**，其中 # 表示从 0 开始的自动递增计数器。
- 偏移量：备用键在记录中基于 0 的偏移量。此值必须为正整数。该字段为必填。
- 长度：备用键的长度。此值必须为正整数。该字段为必填。

  - 唯一：一个复选框，用于指示备用键是否接受重复的条目。如果选中，备用键将被定义为具有唯一性（不接受重复的键条目）。该字段为必填。

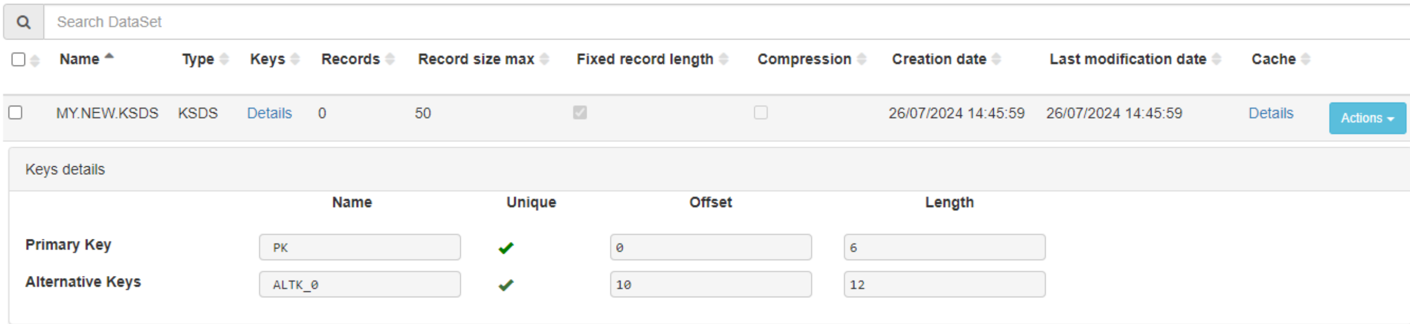
  要删除备用键定义，请使用左侧的垃圾桶按钮。

- 压缩：此复选框用于指定是否使用压缩来存储数据集。
- 启动时启用缓存：此复选框用于指定是否应在应用程序启动时将数据集加载到缓存中。

指定属性定义后，选择创建以继续：


Data set creation
Disable naming rules
DataSet Name
MY.NEW.KSDS
*
Record size max
50
Fixed Record Length
DataSet Type
KSDS
*
Primary Key
Set a key name ('PK' is the default value)
Offset
0
*
Length
6
*
Unique
Alternative Keys
Set a key name ('ALTK_0' is the default value)
Offset
10
*
Length
12
*
Unique
Compression
Enable cache at startup
Create
Cancel

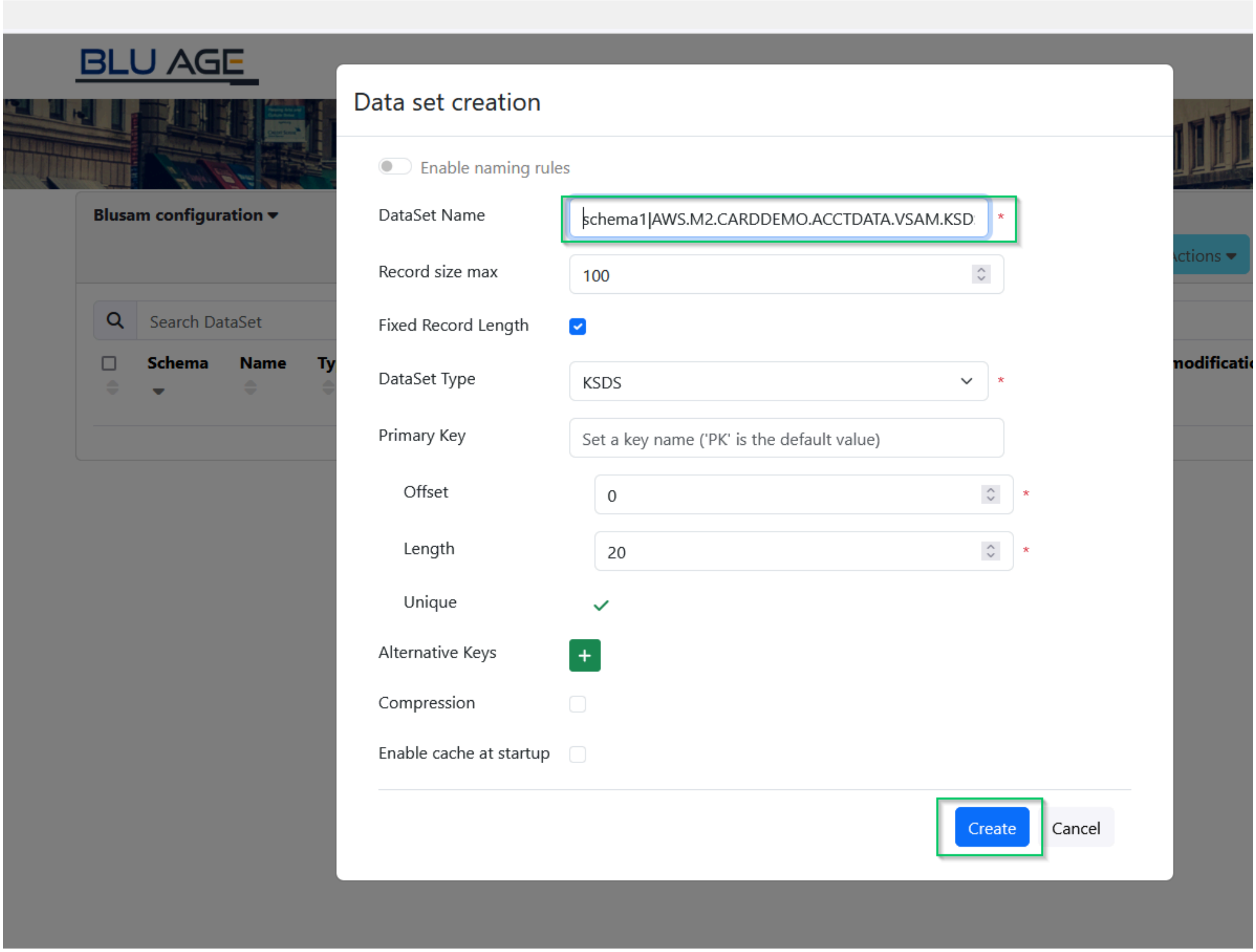
创建窗口将关闭，并显示包含数据集列表的主页。您可以查看新创建数据集的详细信息。



在多架构模式下创建单个数据集

可以在多架构模式下创建数据集，方法是在数据集名称前面加上架构名称，后面加上竖线 (|) 符号（例如）。`schema1|AWS.M2.CARDDEMO.ACCTDATA.VSAM.KSDS`

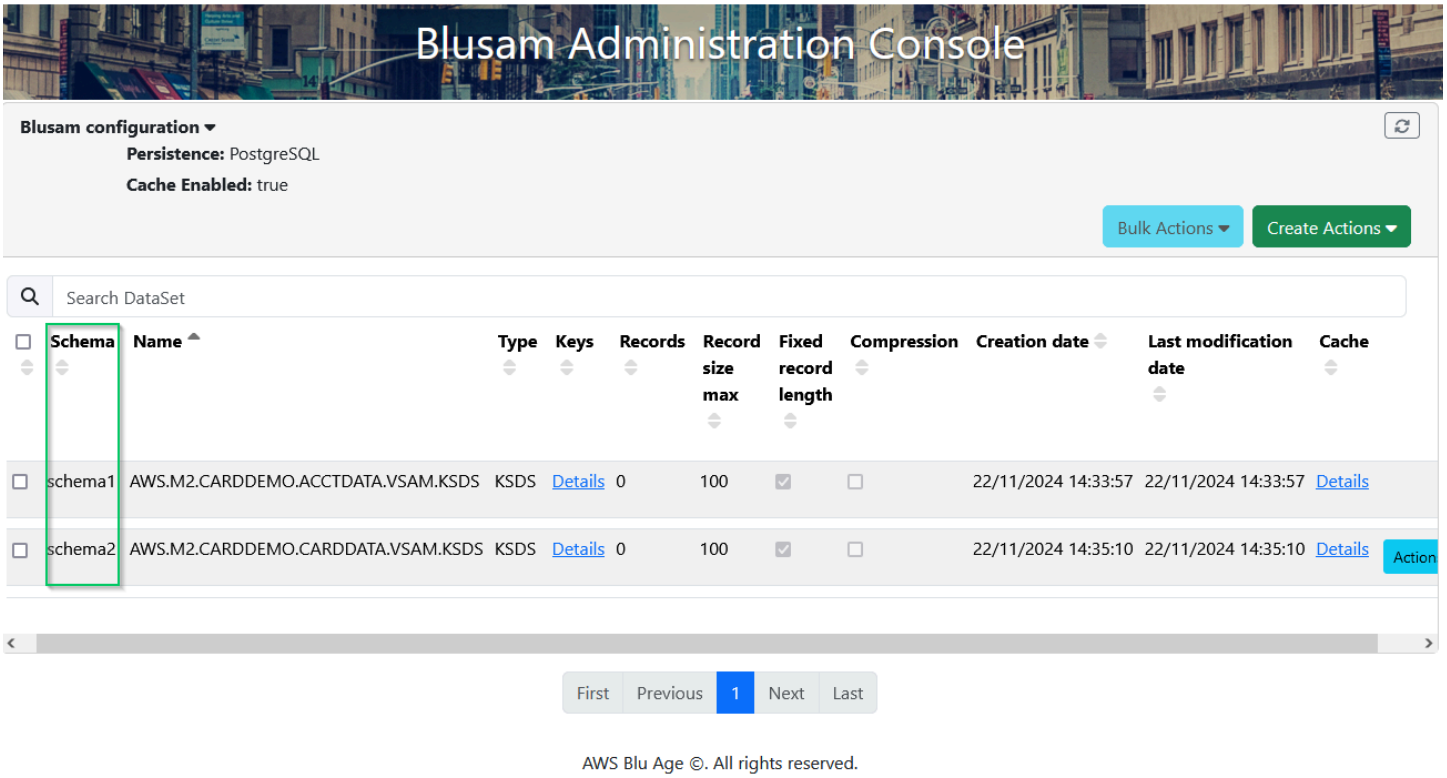
> **Note**
>
> 应在`application-main.yml`配置中指定用于创建数据集的架构。有关更多信息，请参阅 the section called "多架构配置属性"。



如果未提供架构前缀，则将在数据源配置的数据Blusam源 URL 中指定的默认架构中创建Blusam数据集。如果在Blusam数据源 URL 中未指定架构，则默认使用 “public” 架构。

Note

在多架构模式下，BAC 控制台在第一列中显示数据集的架构信息。



从 LISTCAT 创建数据集

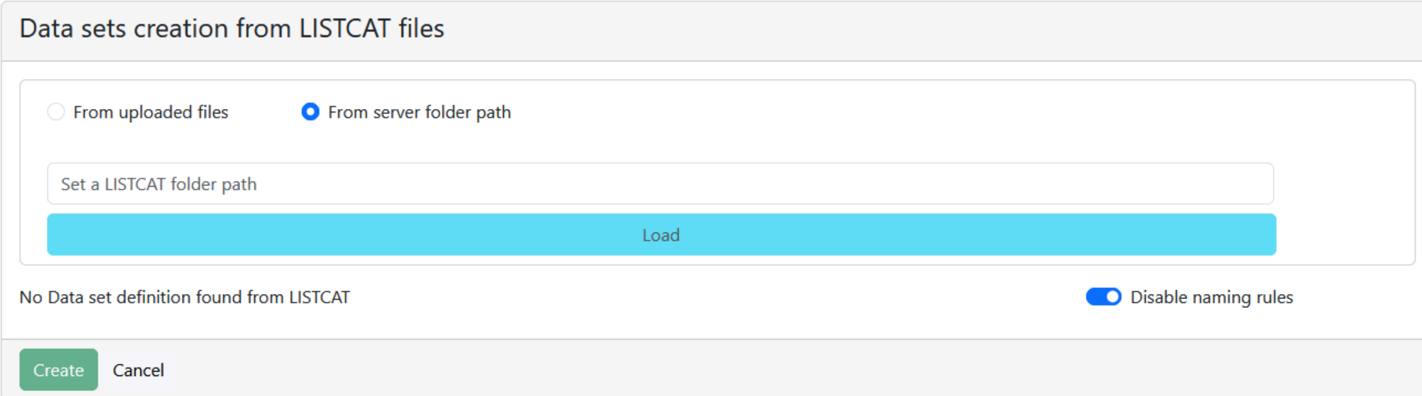
此功能可以利用 BluAge 转换过程中使用 BluInsights 转换中心创建的 LISTCAT JSON 文件，这些文件是解析从旧平台导出的 LISTCAT 的结果：LISTCAT 导出被解析并转换为保存数据集定义（名称、数据集类型、键定义以及记录长度是固定还是可变的）的 JSON 文件。

使用 LISTCAT JSON 文件可以直接创建数据集，而不必手动输入数据集所需的所有信息。您也可以直接创建数据集的集合，而不必逐个创建它们。

如果您的项目没有 LISTCAT JON 文件可用（例如，在转换时没有可用的 LISTCAT 导出文件），则只要遵循附录中详述的 LISTCAT JSON 格式，就可以随时手动创建一个 LISTCAT JON 文件。

在“创建操作”下拉列表中，选择从 LISTCAT 创建数据集。

将显示以下专用页面：



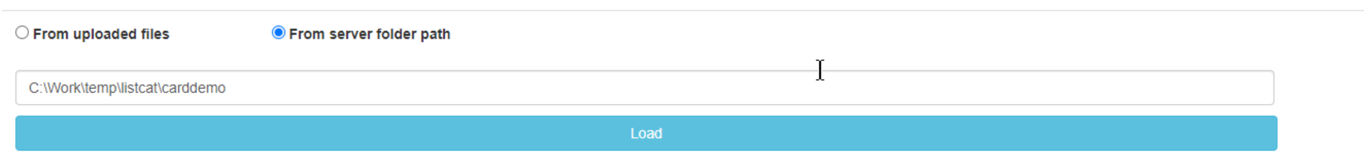
在此阶段，加载按钮处于禁用状态，这是正常的。

使用单选按钮指定要如何提供 LISTCAT JSON 文件。有两种方式：

- 可以使用浏览器上传 JSON 文件。
- 也可以从服务器上的某个文件夹位置选择 JSON 文件。要选择此方式，您必须先将 JSON 文件复制到服务器上您具有适当访问权限的给定文件夹路径。

使用服务器上的 JSON 文件

1. 设置服务器上的文件夹路径，指向包含 LISTCAT JSON 文件的文件夹：

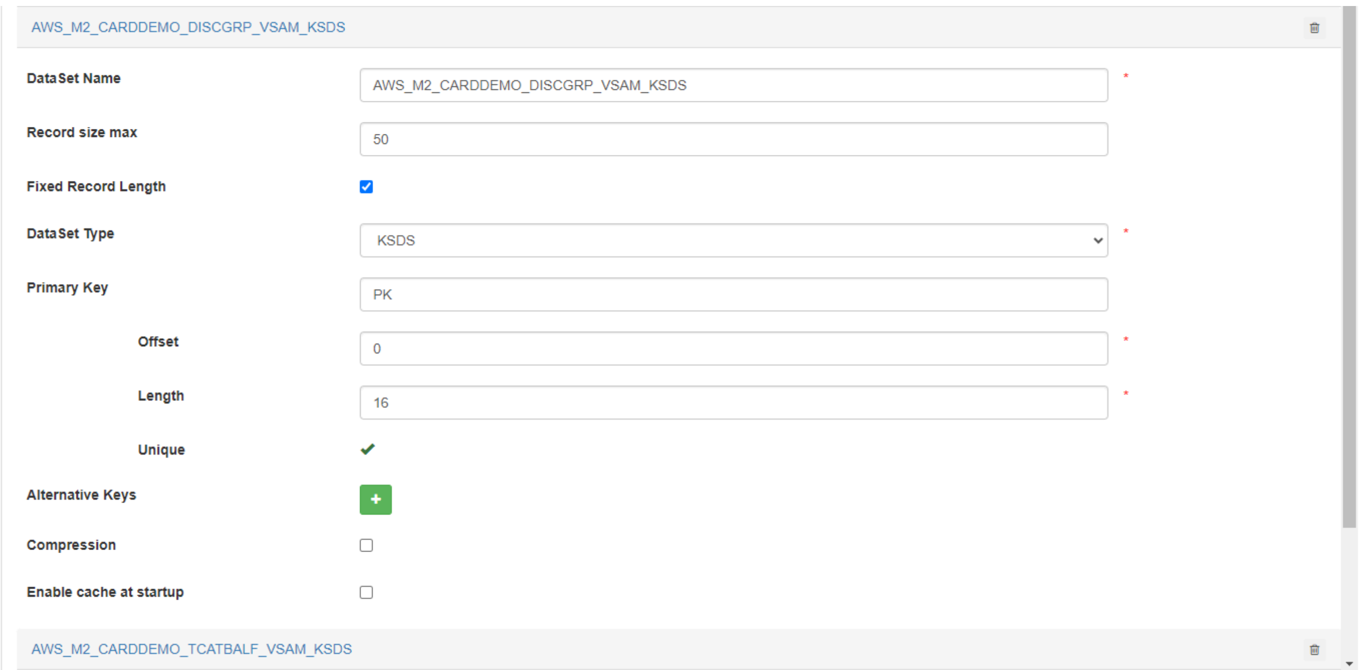


2. 选择上传按钮。所有已识别的数据集定义都将列在表格中：

每行代表一个数据集定义。您可以使用垃圾桶按钮从列表中删除数据集定义。

> **Important**
>
> 从列出中删除的操作将立即执行，不会显示警告消息。

3. 左边的名称是一个链接。您可以选择它来显示或隐藏数据集定义的详细信息，该定义是可编辑的。您可以从解析后的 JSON 文件开始，自由地修改定义。



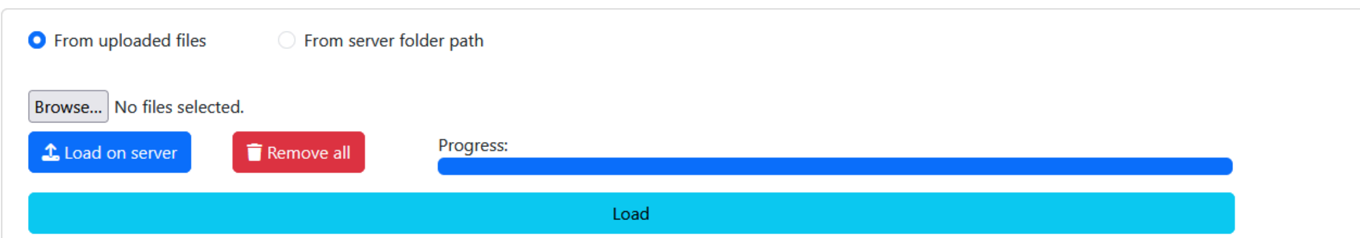
4. 要创建所有数据集，请选择创建。将创建所有数据集，创建的数据集将显示在数据集结果页面上。新创建的数据集将全部包含 0 条记录。

Search DataSet

	Name	Type	Keys	Records	Record size max	Fixed record length	Compression	Creation date	Last modification date	Cache	
☐	AWS.M2.CARDDEMO.CARDDATA.VSAM.KSDS	KSDS	Details	0	150	☑	☐	25/07/2024 15:48:26	25/07/2024 15:48:26	Details	
☐	AWS.M2.CARDDEMO.CARDXREF.VSAM.KSDS	KSDS	Details	0	50	☑	☐	25/07/2024 15:48:26	25/07/2024 15:48:26	Details	
☐	AWS.M2.CARDDEMO.CUSTDATA.VSAM.KSDS	KSDS	Details	0	500	☑	☐	25/07/2024 15:48:26	25/07/2024 15:48:26	Details	
☐	AWS.M2.CARDDEMO.DISCGRP.VSAM.KSDS	KSDS	Details	0	50	☑	☐	25/07/2024 15:48:26	25/07/2024 15:48:26	Details	
☐	AWS.M2.CARDDEMO.TCATBALF.VSAM.KSDS	KSDS	Details	0	50	☑	☐	25/07/2024 15:48:26	25/07/2024 15:48:26	Details	
☐	AWS.M2.CARDDEMO.TRANCATG.VSAM.KSDS	KSDS	Details	0	60	☑	☐	25/07/2024 15:48:26	25/07/2024 15:48:26	Details	
☐	AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS	KSDS	Details	0	350	☑	☐	25/07/2024 15:48:26	25/07/2024 15:48:26	Details	Actions
☐	AWS.M2.CARDDEMO.TRANTYPE.VSAM.KSDS	KSDS	Details	0	60	☑	☐	25/07/2024 15:48:26	25/07/2024 15:48:26	Details	
☐	AWS.M2.CARDDEMO.USRSEC.VSAM.KSDS	KSDS	Details	0	80	☑	☐	25/07/2024 15:48:27	25/07/2024 15:48:27	Details	

将文件上传到服务器

1. 此方式与使用服务器文件夹路径中的文件类似，但在这种情况下，必须先使用文件选择器上传文件。从本地计算机上选择所有要上传的文件，然后选择加载到服务器。



2. 进度条到达末尾时，表示所有文件都已成功上传到服务器，这时加载按钮会启用。选择加载按钮并使用发现的数据集定义，如上文所述。

LISTCAT JSON 格式

LISTCAT JSON 格式由以下属性定义：

- 可选的“catalogId”：遗留目录的标识符为一个字符串，默认目录的标识符为“default”。
- “identifier”：数据集名称，为字符串。
- “isIndexed”：指示 KSDS 的布尔标志：是 KSDS 时，为 true，否则为 false。
- “isLinear”：指示 ESDS 的布尔标志：是 ESDS 时，为 true，否则为 false。
- “isRelative”：指示 RRDS 的布尔标志，是 RRDS 时，为 true，否则为 false。

- 注意："isIndexed"、"isLinear"和"isRelative"是互斥的。
- "isFixedLength记录"：布尔标志：如果设置了固定长度的记录，则设置为 true，否则设置为 false。
- "avgRecordSize": 以字节为单位的平均记录大小，以正整数表示。
- "maxRecordSize": 以字节为单位的最大记录大小，以整数表示。 avgRecordSize 对于固定长度的记录大小，应等于。
- 仅针对 KSDS：必需的主键定义（作为嵌套对象）
  - 标有"primaryKey"
  - "offset"：主键在记录中基于 0 的字节偏移量。
  - "length"：主键的长度（以字节为单位）。
  - "unique"：对于主键，必须设置为 true。
- 对于 KSDS/ESDS，备用键集合（作为嵌套对象的集合）：
  - 标有"alternateKeys"
  - 对于每个备用键：
    - "offset"：备用键在记录中基于 0 的字节偏移量。
    - "length"：备用键的长度（以字节为单位）。
    - "unique"：如果备用键不接受重复条目，则必须设置为 true，否则设置为 false。
- 如果没有备用键，请提供一个空集合：

```
alternateKeys: []
```

下面是一个示例 KSDS LISTCAT JSON 文件。

```
{
  "catalogId": "default",
  "identifier": "AWS_M2_CARDDEMO_CARDXREF_VSAM_KSDS",
  "isIndexed": true,
  "isLinear": false,
  "isRelative": false,
  "isFixedLengthRecord": true,
  "avgRecordSize": 50,
  "maxRecordSize": 50,
  "primaryKey": {
    "offset": 0,
    "length": 16,
    "unique": true
```

```
  },
  "alternateKeys": [
    {
      "offset": 25,
      "length": 11,
      "unique": false
    }
  ]
}
```

为 AWS Blu Age Runtime 设置配置

AWS Blu Age 运行时和客户端代码是使用 Spring Boot 框架的 Web 应用程序。它利用 Spring 功能来提供配置，其中包含多个可能的位置和优先级规则。提供许多其他文件（例如 groovy 脚本、sql 等）时，也有类似的优先级规则。

AWS Blu Age Runtime 还包含其他可选的 Web 应用程序，如果需要，可以选择加入这些应用程序。

主题

应用程序配置基础知识

处理应用程序配置的默认方法是使用应用程序服务器 `config` 文件夹中提供的专用 YAML 文件。以下是两个主要的 YAML 配置文件：

- `application-main.yaml`
- `application-`*`profile`*`.yaml`（其中 *profile* 值是在应用程序生成期间设置的）。

第一个文件用于配置框架，即 `Gapwalk-application.war`，第二个文件用于专门针对客户端应用程序的其他选项。可以使用 Spring 配置文件来实现：Gapwalk 应用程序使用 `main` 配置文件，而客户端应用程序使用 *`profile`* 配置文件。

下面示例显示了典型的主 YAML 文件。

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

下面示例显示了典型的客户端 YAML 文件。

```
# 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
```

有关 YAML 文件内容的信息，请参阅启用 AWS Blu Age 运行时的属性。

应用程序优先级

Spring 优先规则适用于上述配置文件。请注意：

- `application-main` YAML 文件在 Gapwalk 主 war 文件中采用默认值，该默认值会被 `config` 文件夹中的值取代。
- 这点应同样适用于客户端应用程序配置
- 其他参数可以在服务器启动时在命令行中传递。这些参数会覆盖 YAML 文件中的参数。

有关更多信息，请参阅 Spring Boot 官方文档。

用于数据库的 JNDI

数据库配置可能与 Tomcat 中 context.xml 文件中的 JNDI 一并提供。任何此类配置都会覆盖 YAML 文件中的配置。不过，请注意，使用这类配置不允许将您的凭证封装在密钥管理器中（请参阅下文）。

以下示例显示了 JICS 和Blusam数据库的示例配置。

```
<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

将`jdbc/jics`用于 JICS 数据库，`jdbc/bluesam`（注意 'e'）用于Blusam数据库。

url="jdbc:postgresql://XXXX.rds.amazonaws.com:5432/XXXX" username="XXXX" password="XXXX"

数据库 URL、用户名和密码。

AWS 蓝光时代运行时的秘密

某些包含凭证的资源配置可以通过使用 AWS 密钥来获得进一步保护。这个想法是将关键数据存储在 AWS 机密中，并在YAML配置中引用该密钥，以便在Apache Tomcat启动时即时获取秘密内容。

适用于 Aurora 的密钥

Aurora 数据库配置（用于 JICS Blusam、、customer db 等）将使用内置的数据库密钥，该密钥将自动填充相应数据库中的所有相关字段。

> **Note**
>
> `dbname` 密钥是可选的，可以包含在或不包含在密钥中，具体取决于您的数据库配置。您可以将其手动添加到密钥中，也可以通过向 YAML 文件提供名称来添加。

其他密钥

其他密钥用于具有单一密码的资源（尤其是受密码保护的 redis 缓存）。在这种情况下，必须使用其他类型的密钥。

YAML 对密钥的引用

`application-main.yml` 可以引用各种资源的密钥 ARN。

JICS 数据库

JICS 数据库凭证（`spring.aws.jics.db.secret`）

```
spring:
  aws:
    jics:
      db:
```

```
        dbname: jics
        secret: arn:aws:secretsmanager:XXXX
```

支持的 JICS 数据库密钥：

密钥	密钥说明
host	主机名
端口	端口
dbname	数据库名称
username	用户名
password	密码
engine	数据库引擎：Postgres、Oracle、Db2、Microsoft SQL Server
currentSchema	要使用的特定架构（仅支持 Db2）
sslConnection	是否使用 SSL 连接（仅支持 Db2）
sslTrustStoreLocation	信任库在客户端上的位置（仅支持 Db2）
sslTrustStore密码	客户端上信任库的密码（仅支持 Db2）

> **Note**
> 数据库的名称要么在密钥中提供，要么在 yaml 引用 `spring.aws.jics.db.dbname` 中提供。

Blusam 数据库

Blusam数据库凭据 `spring.aws.client.bluesam.db.secret`

```
spring:
  aws:
```

```
      client:
        bluesam:
          db:
            dbname: bluesam
            secret: arn:aws:secretsmanager:XXXX
```

支持的Blusam数据库密钥：

密钥	密钥说明
host	主机名
端口	端口
dbname	数据库名称
username	用户名
password	密码
engine	数据库引擎：Postgres

> **Note**
> 数据库的名称要么在密钥中提供，要么在 yaml 引用 `spring.aws.client.bluesam.db.dbname` 中提供。

客户端数据库

客户端 `application-profile.yml` 可以引用客户端数据库的密钥 ARN。这需要一个额外的属性来列出数据源名称 `spring.aws.client.datasources.names`。对于每个数据源名称 ds_name，请在 `spring.aws.client.datasources.ds_name.secret` 属性中指定密钥 ARN。示例：

```
spring:
   aws:
      client:
        datasources:
          names: primary,host
          primary:
```

```
            secret: arn:aws:secretsmanager:XXXX
          host:
            dbname: hostdb
            secret: arn:aws:secretsmanager:XXXX
```

names: primary,host:

本例中包含名为 primary 和 host 的两个客户端数据源，每个数据源都有各自的数据库和凭证。

dbname: hostdb:

在此示例中，“host””数据库的名称不在密钥中，因此在此处提供，而“primary”数据库在密钥中。

支持的客户端数据库密钥：

密钥	密钥说明
host	主机名
端口	端口
dbname	数据库名称
username	用户名
password	密码
engine	数据库引擎：Postgres、Oracle、Db2、Microsoft SQL Server
currentSchema	要使用的特定架构（仅支持 Db2）
sslConnection	是否使用 SSL 连接（仅支持 Db2）
sslTrustStoreLocation	信任库在客户端上的位置（仅支持 Db2）
sslTrustStore密码	客户端上信任库的密码（仅支持 Db2）

PGM 实用程序数据库

`application-utility-pgm.yml` 可以引用各种资源的密钥 ARN。

- `spring.aws.client.datasources.primary`
  - `secret`

    应用程序数据库的密钥 ARN。

类型：字符串

- `type`

要使用的连接池实现的完全限定名称。

类型：字符串

默认：`com.zaxxer.hikari.HikariDataSource`

- `spring.aws.client.utility.pgm.datasources`
  - `names`

数据来源名称列表。

类型：字符串

- `dsname`
  - `dbname`

主机名称。

类型：字符串

- `secret`

主机数据库的密钥 ARN。

类型：字符串

- `type`

要使用的连接池实现的完全限定名称。

类型：字符串

默认：`com.zaxxer.hikari.HikariDataSource`

对于多数据源密钥：

```
spring:
  aws:
    client:
      primary:
        secret: arn:aws:secretsmanager:XXXX
        type: dataSourceType
      utility:
        pgm:
          datasources:
            names: dsname1,dsname2,dsname3
              dsname1:
                dbname: dbname1
                secret: arn:aws:secretsmanager:XXXX
                type: dataSourceType
              dsname2:
                dbname: dbname2
                secret: arn:aws:secretsmanager:XXXX
                type: dataSourceType
              dsname3:
                dbname: dbname3
                secret: arn:aws:secretsmanager:XXXX
                type: dataSourceType
```

不支持 XA 的密钥

- 发动机 (postgres/oracle/db2/mssql)
- 端口
- dbname
- currentSchema
- username
- password
- url
- sslConnection

- sslTrustStoreLocation
- sslTrustStore密码

对于 `postgres`，只有 `sslMode` 密钥值（`disable/allow/prefer/require/verify-ca/verify-full`）和 `spring.aws.rds.ssl.cert-path` YAML 属性才能使 SSL 连接成为可能。

支持 XA 的密钥

如果客户端数据库使用 XA，则通过密钥值支持子 xa 属性。

- host
- 端口
- dbname
- currentSchema
- username
- password
- url
- sslConnection (true/false)
- sslTrustStoreLocation
- sslTrustStore密码

但是，对于其他 xa 属性（例如 `maxPoolSize` 或 `driverType`），则仍必须提供常规 YAML 密钥 `spring.jta.atomikos.datasource.XXXX.unique-resource-name`。

密钥值会覆盖 YAML 属性。

默认超级管理员 BAC 和 JAC

您还可以通过指定 ARN 将 application-main.yml 配置为从 AWS Secrets Manager 检索密钥中的默认超级管理员用户的用户名和密码。以下示例说明如何在 YAML 文件中声明该密钥。

```
spring:
   aws:
     client:
       defaultSuperAdmin:
```

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

支持的默认超级管理员数据库密钥：

密钥	密钥说明
username	用户名。
password	密码。

OAuth2

您还可以通过指定提供程序和 ARN 将 “application-main.yml” 配置为从中检索 OAuth2 客户端密钥。AWS Secrets Manager 提供者属性的默认值为 Amazon Cognito。以下是 OAuth2 提供商 Keycloak 的配置示例：

```
spring:
   aws:
     client:
       provider: keycloak
       keycloak:
         secret: arn:aws:secretsmanager:XXXX
```

在此示例中， OAuth2 提供商 Keycloak 的客户端密钥是从 AWS Secrets Manager 中指定的 ARN 中检索的。此配置通过动态解析提供者名称和相应的密钥 ARN 来支持多个提供者。

支持的 OAuth2 密钥：

密钥	密钥说明
client-secret	授权服务器在应用程序注册过程中生成的密钥。

用于 Redis 缓存的密钥管理器

`application-main.yml` 文件可以引用 Redis 缓存的密钥 ARN。支持以下各项：

- Gapwalk Redis 凭证（`spring.aws.client.gapwalk.redis.secret`）
- Bluesam Redis 凭证（`spring.aws.client.bluesam.redis.secret`）

- Bluesam 锁 Redis 凭证（`spring.aws.client.bluesam.locks.redis.secret`）
- 数据集目录 Redis 凭证（`spring.aws.client.dataset.catalog.redis.secret`）
- JICS Redis 凭证（`spring.aws.client.jics.redis.secret`）
- 会话 Redis 凭证（`spring.aws.client.jics.redis.secret`）
- 会话跟踪器 Redis 凭证（`spring.aws.client.session.tracker.redis.secret`）
- JICS TS 队列 Redis 凭证（`spring.aws.client.jics.queues.ts.redis.secret`）
- JCL 检查点 Redis 凭证（`spring.aws.client.jcl.checkpoint.redis.secret`）
- Gapwalk 文件锁 Redis 凭证（`spring.aws.client.gapwalk.files.locks.redis.secret`）
- Blu4IV 锁 Redis 凭证（`spring.aws.client.blu4iv.locks.redis.secret`）

以下示例说明如何在 YAML 文件中声明这些密钥。

```
spring:
   aws:
     client:
       gapwalk:
         redis:
           secret: arn:aws:secretsmanager:XXXX
       bluesam:
         locks:
           redis:
             secret: arn:aws:secretsmanager:XXXX
         redis:
           secret: arn:aws:secretsmanager:XXXX
       dataset:
         catalog:
           redis:
             secret: arn:aws:secretsmanager:XXXX
       jics:
         redis:
           secret: arn:aws:secretsmanager:XXXX
       session:
         tracker:
           redis:
             secret: arn:aws:secretsmanager:XXXX
       jics:
         queues:
           ts:
```

```
          redis:
            secret: arn:aws:secretsmanager:XXXX
    jcl:
      checkpoint:
        redis:
          secret: arn:aws:secretsmanager:XXXX
    gapwalk:
      files:
        locks:
          redis:
            secret: arn:aws:secretsmanager:XXXX
    blu4iv:
      locks:
        redis:
          secret: arn:aws:secretsmanager:XXXX
```

支持的 Redis 密钥：

密钥	密钥说明
hostname	Redis 服务器主机名。
端口	Redis 服务器端口。
username	用户名。
password	密码。

用于 SSL 密码设置的密钥管理器

`application-main.yml` 文件可以引用 SSL 密码设置的密钥 ARN。支持以下内容。

- Gapwalk SSL 凭证（`spring.aws.client.ssl.secret`）

以下示例说明如何在 YAML 文件中声明这些密钥。

```
spring:
  aws:
    client:
      ssl:
```

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

密钥	密钥说明
trustStorePassword	信任库密码。
keyStorePassword	密钥库密码。

用于 IBM MQ 密码设置的密钥管理器

该application-main.yml文件可以引用 IBM MQ 设置的机密 ARN。支持以下内容。

- IBM MQ 连接定义为列表，凭证也如此：

  mq.queues.jmsMQQueueManagers[N].secret:

  对于第一个连接，N 从 0 开始。

以下示例说明如何在 YAML 文件中声明这些密钥。

```
mq.queues.jmsMQQueueManagers[0].secret: Secret-0-ARN
mq.queues.jmsMQQueueManagers[1].secret: Secret-1-ARN
```

有关密钥的信息 ARNs，请参阅 Secret s Manager 密钥中有什么？

密钥中定义的属性将覆盖其在 jmsMQ YAML 配置中的相应值。

如果queueManager在密钥中设置，它将覆盖 YAML 文件中的mq.queues.jmsMQQueueManagers[N].jmsMQQueueManager值。

密钥	密钥说明
队列管理器	IBM MQ 队列管理器名称。
appName	IBM MQ 应用程序的名称。
渠道	IBM MQ 频道名称。
host	IBM MQ 的主机名。

密钥	密钥说明
端口	IBM MQ 端口。
userId	IBM MQ 用户名。
password	IBM MQ 用户密码。
maxPoolSize	IBM MQ 的最大池大小。
sslCipherKey	IBM MQ SSL 密码套件。

JHDB 数据库

客户端application-jhdb.yml文件可以引用 JHDB 元数据数据库 ARNs 的密钥。每个数据库连接都需要一个唯一的名称和包含连接凭据的相应机密 ARN。数据库名称在以逗号分隔的列表中定义，每个数据库名称都 ARNs 映射了各个密钥。

```
spring:
   aws:
     jhdb:
       cnxs:
         datasources:
           names: DBD1,DBD2
           DBD1:
             secret: arn:aws:secretsmanager:XXXX
           DBD2:
             secret: arn:aws:secretsmanager:XXXX
```

支持的客户端数据库密钥：

密钥	密钥说明
host	主机名。
端口	端口。
dbname	数据库的名称。
username	用户名。

密钥	密钥说明
password	密码。
engine	数据库引擎：Postgres（现在仅支持 Postgres）。
currentSchema	要使用的特定架构。

其他文件（groovy、sql 等）

客户项目使用的其他文件使用与 Spring 配置文件相似的优先级规则。示例：

- Groovy 脚本是 `scripts` 文件夹或子文件夹中的 `.groovy` 文件。
- SQL 脚本是 `sql` 文件夹或子文件夹中的 `.sql` 文件。
- 进程守护程序脚本是 `daemons` 文件夹或子文件夹中的 `.groovy` 文件。
- 查询数据库映射文件是 `sql` 文件夹或子文件夹中名为 `queries-database.mapping` 的文件。
- Jasper 模板是 `templates` 文件夹或子文件夹中的 `.jrxml` 文件。
- 数据集目录是 `catalog` 文件夹中的 `.json` 文件。
- Lnk 文件是 `lnk` 文件夹中的 `.json` 文件。

所有这些位置都可以通过系统属性或客户端 YAML 属性进行覆盖。

- 对于 Groovy 脚本：`configuration.scripts`
- 对于 SQL 脚本：`configuration.sql`
- 对于进程守护程序脚本：`configuration.daemons`
- 对于查询数据库映射文件：`configuration.databaseMapping`
- 对于 Jasper 模板：`configuration.templates`
- 对于数据集目录：`configuration.catalog`
- 对于 Lnk 文件：`configuration.lnk`

如果找不到该属性，则文件将从上述默认位置获取。首先将使用 tomcat 工作目录作为根目录进行查找，最后在应用程序 war 文件中进行查找。

其他 Web 应用程序

AWS Blu Age Runtime 的`webapps-extra`文件夹中包含其他 Web 应用程序。默认情况下，tomcat 服务器不为这些应用程序提供服务。

是否选择加入这些 Web 应用程序取决于现代化项目，可通过将所需的 war 文件从 `webapps-extra` 文件夹移动到 `webapps` 文件夹来完成。之后，war 文件将在下次启动时由 tomcat 服务器提供服务。

对于每个增加的 war，也可以在 YAML 配置文件中添加一些特定于项目的额外配置，类似在上述 `application-main.yml` 文件中添加配置。其他 war 如下：

- `gapwalk-utility-pgm.war`：包含对 ZOS 实用程序的支持并将 `application-utility-pgm.yaml` 用作其配置。
- `gapwalk-cl-command.war`：包含对 AS/400 实用程序的支持并将 `application-cl-command.yaml` 用作其配置。
- `gapwalk-hierarchical-support.war`: 包含 IMS/MFS 事务支持并`application-jhdb.yaml`用作其配置

启用 AWS Blu Age 运行时的属性

在 Spring Boot 应用程序中，我们可以在 `application-main.yml` 配置文件中定义不同类型的属性，例如侦听端口、数据库连接等。您可以通过此页面了解 AWS Blu Age 运行时的可用属性以及如何启用它们。

主题

YML 表示法

在以下文档中，属性（例如 `parent.child1.child2=true`）写为如下 YAML 格式。

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

快速入门/用例

以下用例显示了适用的键和值的示例。

- 默认 application-main.yml

```
----
#### 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

# Secret manager configuration for global Redis cache
    aws:
      client:
        gapwalk:
          redis:
            secret: arn:aws:secretsmanager:XXXX

###############################################
##### 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
    url : jdbc:postgresql://localhost/bluesam
    username : bluesam
    password : bluesam
    type : org.postgresql.ds.PGSimpleDataSource

##########################################
##### Embedded Bluesam configuration #####
##########################################
bluesam :
  cache : ehcache
  persistence : pgsql
  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

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

####################################
#####     SSL configuration    #####
####################################
gapwalk.ssl.enabled : true
gapwalk.ssl.trustStore : "./config/clientkey.jks"
gapwalk.ssl.trustStorePassword : mysslcertifpassword

##################################
##### MQ settings #####
##################################
mq.queues: jmsmq
mq.queues.jmsMQQueueManagers[0].jmsMQQueueManager: QM1
mq.queues.jmsMQQueueManagers[0].jmsMQAppName: Gapwalk
mq.queues.jmsMQQueueManagers[0].jmsMQChannel: DEV.APP.SVRCONN
mq.queues.jmsMQQueueManagers[0].jmsMQHost: localhost
mq.queues.jmsMQQueueManagers[0].jmsMQPort: 1415
mq.queues.jmsMQQueueManagers[0].jmsMQUserid: app
mq.queues.jmsMQQueueManagers[0].jmsMQSSLCipher: "*TLS12ORHIGHER"
mq.queues.jmsMQQueueManagers[1].jmsMQQueueManager: QM2
mq.queues.jmsMQQueueManagers[1].jmsMQAppName: Gapwalk
mq.queues.jmsMQQueueManagers[1].jmsMQChannel: DEV.APP.SVRCONN
mq.queues.jmsMQQueueManagers[1].jmsMQHost: localhost
mq.queues.jmsMQQueueManagers[1].jmsMQPort: 1415
mq.queues.jmsMQQueueManagers[1].jmsMQUserid: app

#########################################################
##### Configuration properties for JMS MQ connection ####
#########################################################
mq.queues.jms.connectionfactory.borrow-connection-timeout: 31 # Timeout, in seconds,
 for borrowing connections from the pool.
mq.queues.jms.connectionfactory.ignore-session-transacted-flag: true # Whether or not
 to ignore the transacted flag when creating a session.
```

```
mq.queues.jms.connectionfactory.local-transaction-mode: false # Whether or not to
 include local transactions.
mq.queues.jms.connectionfactory.maintenance-interval: 62 # The time, in seconds,
 between runs of the pool's maintenance thread.
mq.queues.jms.connectionfactory.max-idle-time: 63 # The time, in seconds, after which
 connections are cleaned up from the pool.
mq.queues.jms.connectionfactory.max-lifetime: 0 # The time, in seconds, that a
 connection can be pooled for before being destroyed. 0 denotes no limit.
mq.queues.jms.connectionfactory.max-pool-size: 1 # The maximum size of the pool. This
 property will be overwritten by mq.queues.jmsMQQueueManagers[N].jmsMQMaxPoolSize
 property (if provided).
mq.queues.jms.connectionfactory.min-pool-size: 1 # The minimum size of the pool.
mq.queues.jms.connectionfactory.reap-timeout: 0 # The reap timeout, in seconds, for
 borrowed connections. 0 denotes no limit.


###############################
##### 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

######################################################
#####  BLOCKSIZE DEFAULT VALUE  ####
######################################################
#blockSizeDefault: 32760

#####################################
#####  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

########################################
#####  JCL Batch Restart Mechanism  ####
########################################
jcl:
checkpoint:
enabled: false
#expireTimeout: -1
#expireTimeoutUnit: SECONDS # Supported values: java.util.concurrent.TimeUnit
#provider: redis
```

```
###############################
#####  System parameters  ####
###############################
system.date.format:MDY

################################################################
#####  Database metadata schema inclusion configuration  ####
################################################################
gapwalk:
  database:
    metadata:
       schema-inclusion:
          enabled: false
          schemas:
            global:
              - "SCHEMA1"
              - "SCHEMA2"

###############################
#####  URL configuration  ####
###############################
gapwalk.post.script.mediatype.json: false

#########################################
#####  CBLQDA parameters  ####
#########################################
gapwalk.cobol.cblqda.enabled: false
----
```

- 在 LISTCAT 命令中使用长度可变的文件

```
[**/*.*]
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)
```

- 在 LOAD/UNLOAD 实用程序中提供空字节指示器值

```
# 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
```

主应用程序的可用属性

此表提供了 key/values 参数的详尽视图。

> **Note**
>
> 如果您要迁移到 5.1.0 或更高版本并使用默认行为的 bluesam： openWarmUp
>
> - 的默认值`bluesam.openWarmUp`已从`true`更改为 `false`
> - 要保持现有行为，必须明确设置 `bluesam.openWarmUp=true`

Key	Type	默认 值	说明	发行版
`system.date.format`	字符串	MDY	系统日期格式 DATFMT。	---

Key	Type	默认 值	说明	发行版
`logging.config`	路径	classpath :logback-main.xml	用于引用 logback 配置文件的标准密钥。其他标准日志记录密钥也可用。	
`spring.jta.enabled`	布尔值	false	标准密钥。如果数据源支持模式不是 static-xa ，则必须禁用 Spring JTA 事务自动配置。	
`datasource.jicsDs` + `-driver-class-name` + `-url` + `-username` + `-password` + `-type`	带子键的标准 Spring 数据源		包含 Jics 数据库的连接信息。或者，强烈建议使用 AWS 密钥，如the section called “JICS 数据库”中所述。	
`jildb.datasource` + `-driver-class-name` + `-url` + `-username` + `-password`	带子键的标准 Spring 数据源		包含 Jics 数据库的连接信息。	
`jildb.backend`	字符串		指定 Jildb 后端中使用的数据库类型，例如 Oracle。	

Key	Type	默认 值	说明	发行版
`jildb.record-hold-timeout`	int	10000	指定 Jildb 中记录保存机制的等待超时时间（以毫秒为单位）。	
`datasource.bluesamDs` + `-driver-class-name` + `-url` + `-username` + `-password` + `-type`	带子键的标准 Spring 数据源		包含Blusam数据库的连接信息。或者，强烈建议使用 AWS 密钥，如the section called “Blusam 数据库”中所述。	
`blu4iv.userspace.pageSize`	int	4096	指定用于为用户空间分配存储空间的页面大小。在任何给定时间，用户空间的大小始终是该指定数字的倍数。	4.10.0
`blu4iv.userspace.store-type`	字符串	memory	指定用于用户空间注册表的存储类型。支持的值是内存和 redis。如果未提供存储类型，则默认使用内存。	4.10.0

Key	Type	默认 值	说明	发行版
`blu4iv.userspace.redis.*`	支持的 Redis 属性		为用户空间 redis 存储指定配置属性，请参阅the section called “可用的 Redis 缓存属性”。	4.10.0
`blu4iv.dataaccess.useOldDaoDesign`	布尔值	false	一个全局标志，用于确定应用程序是使用旧的 DAO 架构还是新的 DAO 架构。这取决于 DAO 是使用旧设计还是使用新设计生成的。	
`bluesam.disabled`	布尔值	false	是否Blusam完全禁用。	
`bluesam.cache`	字符串		如果未设置，则不会使用Blusam缓存。可能的值（缓存实现）是 ehcache 和 redis ()。the section called “Redis 缓存属性”	

Key	Type	默认 值	说明	发行版
`bluesam.openWarmUp`	布尔值	false	指定在打开期间是否缓存任何 bluesam 数据集。 警告：5.1.0 版本中的重大更改 -默认值从更改**`true`**为**`false`**。	5.1.0
`bluesam.fileList`	String[]		指定打开期间要缓存的 bluesam 数据集列表。 注意： -此参数仅在设置**`bluesam.openWarmUp`** 为时才有效**`false`**。	5.1.0
`bluesam.maxBluesamDisablingThreadpoolSize`	数字	10	指定用于禁用 bluesam 数据集进行批处理的最大线程池大小。	4.5.0
`bluesam.bluesamStatusPollingInterval`	数字	1000	指定轮询 bluesam 状态以检查在线活动时，每次迭代之间等待的时间（以毫秒为单位）。	4.5.0

Key	Type	默认 值	说明	发行版
bluesam.maxBluesamStatusPollingRetry	数字	3	指定轮询 bluesam 状态失败时的最大重试次数。	4.5.0
bluesam.checkBluesamStatus	布尔值	false	指定是否在访问 bluesam 数据集之前检查其状态。	4.5.0
bluesam.prefetchWindowSize	int	1000	指定要预取并在内存中缓存以优化 Redis 缓存的记录数。此属性仅在使用 Redis 缓存时有效。	
bluesam.prefetchWindowMultiples	int	5	指定预取窗口大小的乘数，以确定最大预取窗口容量。最大容量的计算公式为`prefetchWindowSize` × `prefetchWindowMultiples` 。此属性仅在使用 Redis 缓存时有效。	

Key	Type	默认 值	说明	发行版
spring.aws.client.bluesam.redis.secret	字符串	null	指定 Bluesam Redis 缓存的凭证密钥 ARN，请参阅 the section called “AWS 蓝光时代运行时的秘密”。	
spring.aws.client.bluesam.locks.redis.secret	字符串	null	指定 Bluesam 锁 Redis 缓存的凭证密钥 ARN，请参阅 the section called “AWS 蓝光时代运行时的秘密”。	
forcedDate	字符串		强制将日期（如果有）更改为提供的日期。	
clcommand.rtvmbrd.hasHeader	字符串	true	指定文件是否包含在计算行数时不应包含的标题。	
frozenDate	布尔值	true	指定是否冻结日期。仅在设置了 forcedDate 时适用。	

Key	Type	默认 值	说明	发行版
externalSort.threshold	数据大小（例如：12 MB）		排序阈值：确定系统何时应使用文件系统上的临时文件从内存中的合并排序切换到外部合并排序。如果未指定 (default=null)，则阈值将设置为每个新排序任务中可用内存的一半。为了估计可用内存，需要调用 system.gc ()。	
blockSizeDefault	数字	32760	用于 BDW 字节的默认区块大小。	
jics.parameters.datform	字符串	MMDDYY	日期形式。	

Key	Type	默认 值	说明	发行版
jics.initList	字符串		初始化 JICS 列表，用逗号分隔。指定时，该键定义列表中以逗号分隔的名称，以便在 Apache Tomcat 启动时在 CICS 列表中激活。示例值：UUU,DFHIVPL,PEZ1 。这将级联到这些列表中包含的组及其基础资源定义，之后会对运行时可见。默认为空。	
jics.parameters.applid	字符串	VELOCITY	用于标识 JICS 中的应用程序的 appid（至少 4 个字符，无最大长度限制）。	
jics.parameters.sysid	字符串	CICS	系统标识 (SYSID)。	
jics.parameters.eibtrmid	字符串	TERM	终端标识符（最多 4 个字符，最少 1 个字符）。	

Key	Type	默认 值	说明	发行版
jics.parameters.userid	字符串		用户 ID（最多 8 个字符，无最小字符数限制）。如果未提供任何值（默认为空白），则使用 HTTP 会话 ID 作为用户 ID。	
jics.parameters.username	字符串	MYUSERNAME	用户 ID（最多 10 个字符，最少 1 个字符）。	
jics.parameters.netname	字符串	MYNETNAME	网络名称（最多 8 个字符，最少 1 个字符）。	
jics.parameters.opid	字符串	XXX	3 个字符的操作员标识。	
jics.parameters.jobname	字符串	MJOBNAME	作业名称。	
jics.parameters.sysname	字符串	SYSNAME	AS400 系统名称 (sysname)。	
jics.parameters.cwa.length	数字	0	公共工作区（CWA）长度。	
jics.parameters.charset	字符串	CP037	JICS 全球使用的字符集。	

Key	Type	默认 值	说明	发行版
jics.parameters.tsqimpl	字符串	bluesam	JICS 临时存储队列 (TSQ) 实现（允许的值为 bluesam/memory/	
jics.queues.ts.redis.*	支持的 Redis 属性		指定 JICS TS 队列 Redis 服务器的配置属性，请参阅 the section called “支持的 Redis 属性”。	
spring.aws.client.jics.queues.ts.redis.secret	字符串	null	指定 JICS TS Queues Redis 服务器的凭证密钥 ARN，请参阅 the section called “AWS 蓝光时代运行时的秘密”。	
lockTimeout	数字	500	锁定超时，以毫秒为单位。	
sqlCodePointShift	数字		可选。sql 代码点转换。对在将遗留 RDBMS 数据迁移到现代 RDBMS 时可能遇到的控制字符的代码点进行转换。例如，您可以指定 384 来匹配 Unicode 字符 \u0180。	

Key	Type	默认 值	说明	发行版
sqlIntegerOverflowAllowed	布尔值	false	指定是否允许 SQL 整数溢出，即是否允许在主机变量中放置更大的值。	
database.cursor.overflow.allowed	布尔值	true	指定是否允许游标溢出。设置为 `true` 时，无论游标位置如何，均可在游标处执行下一次调用。设置为 `false` 时，会在游标处执行下一次调用之前检查游标是否位于最后一个位置。仅在游标可滚动（敏感或不敏感）的情况下将其启用。	
reportOutputPath	字符串	/reports	报告输出路径。	

Key	Type	默认 值	说明	发行版
server.servlet.session.tracking-modes	字符串	Cookie	指定要使用的会话跟踪模式。可能的值为 url、cookie、ss 可以指定多个值，用逗号分隔。注意：在 Tomcat 上部署时，需要cookies=false 在 context.xml 中进行设置，因为 Tomcat 优先配置跟踪模式。	
server.servlet.session.timeout	字符串	30m	指定会话超时持续时间。示例：60s（60 秒）、4m（4 分钟）、4h（4 小时）。	
spring.session.store-type	字符串	none	高可用性环境的会话缓存。可能的值为 none 或 redis。默认值为 none。	
stopExecutionWhenProgNotFound	布尔值	true	指定在找不到程序时是否停止运行。如果设置为 true，则在找不到程序时中断运行。	

Key	Type	默认 值	说明	发行版
forceHR	布尔值	false	指定是否在控制台或文件输出上使用人类可读的 SYSPRINT。	
rollbackOnRTE	布尔值	false	指定是否在运行时异常时回滚隐式运行单元事务。	
sctThreadLimit	长整数	5	用于触发脚本的线程限值。	
dataSimplifier.onInvalidNumericData	字符串	reject	解码无效的数字数据时如何应对。允许的值包括：reject、tol paces 、tolera paceslowv alues 、tolera ost 。默认值为 reject。	
dataSimplifier.doubleFloatingType	字符串	IEEE_754	双浮点类型是用于对浮点数进行编码和解码的格式。允许的值为IEEE_754/HFP（于十六进制浮点格式）。默认值为 IEEE_754。	4.7.0
filesDirectory	字符串		批处理 input/output 文件的目录。	

Key	Type	默认 值	说明	发行版
ims.messages.extendedSize	布尔值	false	指定是否在 IMS 消息上设置扩展的大小。	
defaultKeepExistingFiles	布尔值	false	指定是否设置数据集的默认先前值。	
jics.db.ddlScriptLocation	字符串		Jics DDL 脚本位置。支持您使用 .sql 脚本启动 Jics 数据库架构。默认情况下为空。例如 ./jics/sql/jics.sql 。	
jics.db.schemaTestQueryLocation	字符串		sql 文件的位置，该文件应包含唯一查询，且该查询返回 jics 架构中对象的数量（如果有）。	
jics.db.dataScriptLocation	字符串		定义用于初始化 JICS 数据库的 SQL 脚本的路径。接受以逗号分隔的文件和目录列表，允许指定多个脚本和文件夹。	

Key	Type	默认 值	说明	发行版
jics.db.dataTestQueryLocation	字符串		包含单个 sql 查询的 sql 脚本的位置，该查询应返回对象计数（例如：计算 jics 程序表中的记录数）。如果计数等于 0，则将使用 jics.db.dataScriptLocation 脚本加载数据库，否则将跳过数据库加载。	
jics.data.dataJsonInitLocation	字符串			
jics.xa.agent.timeout	数字			
query.useConcatCondition	布尔值	false	指定键条件是否通过串联键来构建。	
system.qdecfmt	字符串			

Key	Type	默认 值	说明	发行版
disposition.checkexistence	布尔值	false	指定是否针对配置 DISP SHR 或 OLD 的数据集检查文件是否存在。	
useControlMVariable	布尔值	false	指定是否使用 control-M 规范进行变量替换。	
card.encoding	字符串	CP1145	卡片编码：与 useControlMVariable 一起使用。	
mapTransfo.prefixes	字符串	&,@,%%	转换 controlM 变量时要使用的前缀列表。每一个前缀均用逗号隔开。	
checkinputfilesize	布尔值	false	指定在文件大小为记录大小倍数的情况下是否要进行检查。	
stepFailWhenAbend	布尔值	true	指定在步骤失败或完成执行时是否引发异常中止。	
returnHttp5xxResponseForFailedSyncJob	布尔值	false	指定是否针对同步作业执行失败返回 500 系列的 HTTP 响应代码。	4.10.0

Key	Type	默认 值	说明	发行版
bluesam.fileLoading.commitInterval	数字	100000	bluesam 提交间隔。	
uppercaseUserInput	布尔值	true	指定用户输入是否必须采用大写形式。	
jhdb.lterm	字符串		允许您在进行 IMS 仿真的情况下强制使用通用的逻辑终端 ID。如果未设置，则使用会话 ID。	
jhdb.identificationCardData	字符串		用于将某些“操作员识别卡数据”硬编码到 CARD 参数指定的 MID 字段。默认为空白，没有输入限制。	
encoding	字符串	ASCII	项目中使用的编码（不在 groovy 文件中）。预期的有效编码包含 CP1047、IBM930.	
cl.configuration.context.encoding	字符串	CP297	CL 文件的编码。预期的有效编码包含 CP1047、IBM930. 默认值为 CP297。	

Key	Type	默认 值	说明	发行版
cl.zonedMode	字符串	EBCDIC_STRICT	对控制语言 (CL) 命令进行编码或解码的模式。允许的值包括：EBCDIC_STRICT 、EBCDIC_I DIFIED 、AS400.	
ims.programs	字符串		要使用的 IMS 程序列表。用分号 (;) 分隔每个参数，用逗号 (,) 分隔每个事务。例如：PCP008,PCT008;PCP054,PCT054;PCP066,PCT066;PCP068,PCT068;	
jhdb.configuration.context.encoding	字符串	CP297	Java 分层数据库 (JHDB) 编码。预期的有效编码字符串包含 CP1047、IBM930.	
jhdb.metadata.extrapath	字符串	file:./setup/	用于为 psbs 和 dbds 文件夹指定一个特定于运行时的额外根文件夹的配置参数。	

Key	Type	默认 值	说明	发行版
jhdb.checkpointPersistence	字符串	none	检查点持久性模式。允许的值包括：none、add、使用 add 可在创建新检查点并将其添加到注册表后保留该检查点。使用 end 可在服务器关闭时保留检查点。任何其他值都会禁用持久性。请注意，每次向注册表中添加新的检查点时，所有现有的检查点都将被序列化并且文件会被擦除，而不是添加到文件中现有数据中。这样可能会对性能产生一些影响，具体取决于检查点的数量。	

Key	Type	默认 值	说明	发行版
jhdb.checkpointPath	字符串	file:./setup/	如果 jhdb.checkpointPersistence 不是 none，则此参数允许您设置检查点持久性路径（checkpoint.dat 文件存储位置），注册表中包含的所有检查点数据都将序列化并备份到所提供文件夹的文件 (checkpoint.dat) 中。请注意，此备份仅涉及检查点数据（scriptId、stepId、数据库位置和检查点区域）。	
jhdb.navigation.cachenexts	数字	5000	RDBMS 分层导航中使用的缓存持续时间（以毫秒为单位）。	
jhdb.use-db-prefix	布尔值	true	指定是否在 RDBMS 的分层导航中启用数据库前缀。	

Key	Type	默认 值	说明	发行版
`jhdb.query.limitJoinUsage`	布尔值	true	指定是否在 RDBMS 图形上使用限制联接使用参数。	
`taskExecutor.corePoolSize`	数字	5	当通过 groovy 脚本启动终端中的事务时，会创建一个新线程。可以使用此参数设置核心池大小。	
`taskExecutor.maxPoolSize`	数字	10	当通过 groovy 脚本启动终端中的事务时，会创建一个新线程。使用此参数设置池的最大大小（并行线程的最大数量）。	
`taskExecutor.queueCapacity`	数字	50	当通过 groovy 脚本启动终端中的事务时，会创建一个新线程。使用此参数设置队列大小。（= 达到 `taskExecutor.maxPoolSize` 时待处理事务的最大数量）	

Key	Type	默认 值	说明	发行版
taskExecutor.allowCoreThreadTimeOut	布尔值	false	指定是否允许核心线程在 JCIS 中超时。即使与非零队列结合使用，也可以实现动态增长和缩小（因为只有在队列已满后，池最大大小才会增长）。	
jics.runUnitLauncherPool.enable	布尔值	false	指定是否在 JICS 中激活运行单元启动器池。	
jics.runUnitLauncherPool.size	数字	20	JICS 中的运行单元启动器池大小。	
jics.runUnitLauncherPool.validationInterval	数字	1000	每次运行调整池大小的任务之间的间隔。	
jics.runUnitLauncherPool.parallelism	数字	2	当调整任务运行时，用于在队列中生成缺失实例的线程数。	
context.preconstruct.enable	布尔值	false	指定是否激活程序上下文的预构造。	

Key	Type	默认 值	说明	发行版
context.preconstruct.frequencyInMillis	数字	100	每次运行调整池大小的任务之间的间隔。	
context.preconstruct.parallelism	数字	5	当调整任务运行时，用于在队列中生成缺失实例的线程数。	
context.preconstruct.minInstances	数字	2	首次需要上下文时将创建的实例数量。	
spring.aws.application.credentials	字符串	null	从 JICS 的 AWS 凭证配置文件中加载凭证。	
jics.queues.sqs.region	字符串	eu-west-1	JICS 中使用的 Amazon 简单队列服务的 AWS 区域。	
jics.jcl.rt.encoding	字符串	CP037	在专用 JICS 队列中编写的 JCL 脚本的编码。	
jics.jcl.rt.queue	字符串	JICS	运行时可以逐行写入 JCL 脚本的队列的名称。	

Key	Type	默认 值	说明	发行版
mq.queues.sqs.region	字符串	eu-west-3	AWS SQS MQ 服务的 AWS 区域。	
quartz.scheduler.stand-by-if-error	布尔值	false	指定当作业调度程序处于待机模式时是否触发作业执行。如果设置为 true，则启用时不会触发作业执行。	
databaseStatistics	布尔值	false	指定是否允许 SQL 生成器收集和显示统计信息。	
dbDateFormat	字符串	yyyy-MM-dd	数据库目标日期格式。	
dbTimeFormat	字符串	HH:mm:ss	数据库目标时间格式。	
dbTimestampFormat	字符串	yyyy-MM-dd HH: mm: ss.sssssss	数据库目标时间戳格式。	
dateTimeFormat	字符串	ISO	dateTimeFormat 描述了如何将数据库日期时间戳类型泄漏到数据简化器实体中。允许的值包括：ISO、EUR、EI	

Key	Type	默认 值	说明	发行版
localDateFormat	字符串		本地日期格式列表。使用 \ 分隔每种格式。	
localTimeFormat	字符串		本地时间格式列表。使用 \ 分隔每种格式。	
localTimestampFormat	字符串		本地时间戳格式列表。使用 \ 分隔每种格式。	
pgmDateFormat	字符串	yyyy-MM-dd	日期时间格式	
pgmTimeFormat	字符串	HH.mm.ss	程序 (pgm) 执行的时间格式。	
pgmTimestampFormat	字符串	yyyy-MM-dd-HH.mm.ss.SSSSSS	时间戳格式。	
cacheMetadata	布尔值	true	指定是否缓存数据库元数据。	
forceDisableSQLTrimStringType	布尔值	false	指定是否禁用所有 sql 字符串参数的 TRIM。	
fetchSize	数字		游标的 fetchSize 值。在通过加载/卸载 utils 使用区块获取数据时使用。	

Key	Type	默认 值	说明	发行版
check-groovy-file	布尔值	true	指定是否在注册前检查 groovy 文件内容。	
qtemp.uuid.length	数字	9	QTEMP 的唯一 ID 长度。	
qtemp.dblog	布尔值	false	是否启用 QTEMP 数据库日志记录。	
qtemp.cleanup.threshold.hours	数字	0	指定何时启用 qtemp.dblog 。数据库分区的生命周期（以小时为单位）。	
invalidDataTolerence	布尔值	true	指定打包类型是否支持无效数据。	
program.timeout	数字	–1	指定任何 program/transaction 执行的超时时间（以秒为单位）。在这段时间之后，系统将尝试中断程序。	

Key	Type	默认 值	说明	发行版
gapwalk.line.separator	字符串	null	在 gapwalk 中指定行分隔符类型。支持的值为 WIN (CRLF)/UNIX (LF)/LINUX (LF)。其他值将被忽略，并使用 line.sepatator 系统属性。	
enableActivePgmIdCache	布尔值	false	指定是否启用活动程序 ID 本地缓存。请谨慎使用此特征，因为 JICS 资源可以在程序和用户之间共享。任何管理员都可以在外部更改这些资源，并且所放置的本地缓存可能会失效。	

Key	Type	默认 值	说明	发行版
mq.queues.default.syncpoint	布尔值	false	指定未设置 MQPMO_SYNCPOINT 和 MQPMO_NO_SYNCPOINT 时 MQ PUT 命令的默认行为。当设置为 true 时，它的作用就像 MQPMO_SYNCPOINT ，在 PUT 命令期间不会直接提交消息。当设置为 false 时，它的作用就像 MQPMO_NO_SYNCPOINT ，在 PUT 命令期间会直接提交消息。	
dataSimplifier.byteRangeBoundsCheck	布尔值	false	设置为 true 时，它可确保使用不正确的值创建 no ByteRange 。默认值为 false。	
file.stdoutIntoLogger	布尔值	false	指定是否启用写入记录器，而不是 SYSPRINT 和 SYSPUNCH 文件中的默认系统输出流。	

Key	Type	默认 值	说明	发行版
`tempFilesDirectory`	字符串	null	指定生成的临时文件所在文件夹位置的名称。	
`cleanTempFilesDirectoryAtStartup`	布尔值	true	指定是否在应用程序启动时清除临时文件文件夹的内容。	

Key	Type	默认 值	说明	发行版
tempFolderPattern	字符串	null	指定一种模式，该模式将用于根据以下预定义的和可自定义的信息动态构建临时文件夹的名称。 HOST：主机名。 JOBID：作业 ID。 HASHCODE：作业上下文的哈希码。 TIMESTAMP：获取时间戳时使用的模式。临时文件夹的目标名称是 TMP_DIR_{}。tempFolderPattern例如，在以下模式中，名称将以作业 ID 开头并以 “时间戳” 结尾： tempFolderPattern: JOBID、host=xxxx、HASHCODE、timestamp=yyymmddhhmmss。如果该属	

Key	Type	默认 值	说明	发行版
			性未添加到 YAML 文件中或该属性为空，tempFolderPattern 则临时文件夹的名称将为 “TMP_DIR_” + this.hashCode () ()。DefaultJobContext	
database.cursor.raise.already.opened.error	布尔值	false	指定是否启用在打开已打开的游标时引发 SQLCODE 错误 502。	
jics.spool.smtp.hostname	字符串	null	指定 SMTP 服务器主机。示例：smtp.xxx.com	
jics.spool.smtp.port	字符串	null	指定 SMTP 服务器端口。示例：25	
jics.spool.smtp.password	字符串	null	指定 SMTP 服务器的登录密码。	
jics.spool.smtp.username	字符串	null	指定 SMTP 服务器的用户名。	

Key	Type	默认 值	说明	发行版
jics.spool.smtp.debug	布尔值	false	指定 SMTP 服务器的调试模式。	
gapwalk-application.security	字符串	disabled	切换全局安全配置（XSS、CORS、CSRF、OAUTH 身份验证...）。支持的值包括 disabled 和 enabled。	
gapwalk-application.security.identity	字符串	null	全局身份验证方法。建议值为 oauth。支持的值包括 json 和 oauth。当 gapwalk-application.security 为 enabled 时，该属性为必需属性。	
gapwalk-application.security.issuerUri	字符串	null	身份提供者（IdP）的颁发机构 URI。当 gapwalk-application.identity 为 oauth 时，该属性为可选属性。	

Key	Type	默认 值	说明	发行版
gapwalk-application.security.allowedOrigins	string[]	null	支持的源列表。此选项要求将 gapwalk-application.identity 设置为 oauth。	
gdgDirectoryPath	字符串	output/gdg	GDG 目录路径是存储 gdg 文件的目录。	4.6.0
gapwalk-application.security.customAllowedHeaders	String[]	null	允许的自定义标题列表。此选项要求将 gapwalk-application.identity 设置为 oauth。	4.8.0
listOfMDCUrlParams	字符串	null	执行 Groovy 批处理脚本时要添加到 MDC 的 URL 参数列表。	
gapwalk-application.security.claimGroupName	字符串	cognito:groups	声明属性，包含用户所属的所有组的列表。对于 Amazon cognito，请使用 cognito:groups ；对于外部 IdP，请使用任何其他字符串。	

Key	Type	默认 值	说明	发行版
gapwalk-application.security.userAttributeName	字符串	username	用于标识用户请求的声明属性名称。对于 Amazon cognito，请使用 username；对于 Keycloak，请使用 preferred_username ；对于外部 IdP，请使用任何其他字符串。	
gapwalk-application.defaultSuperAdminUserName	字符串	sadmin	禁用 gapwalk-application.security 时，请指定默认的本地超级用户名。	
gapwalk-application.defaultSuperAdminUserPwd	字符串	sadmin	禁用 gapwalk-application.security 时，指定默认的本地超级用户密码。	
gapwalk-application.security.filterURIs	字符串	disabled	切换过滤 URIs 配置。支持的值包括 disabled 和 enabled。	

Key	Type	默认 值	说明	发行版
gapwalk-application.security.blockedURIs	String[]	null	URIs 要屏蔽的列表。当 gapwalk-application.security.filterURIs 为 enabled 时，该属性为必需属性。	
jics.redis.*	支持的 Redis 属性		指定 JICS Redis 服务器连接工厂的配置属性，请参阅the section called “支持的 Redis 属性”。	
spring.aws.client.jics.redis.secret	字符串	null	指定 JICS Redis 服务器连接工厂的凭证密钥 ARN，请参阅 the section called “AWS 蓝光时代运行时的秘密”。	

Key	Type	默认 值	说明	发行版
`jcl.checkpoint.enabled`	布尔值	false	指定是否启用 JCL 检查点机制以支持作业重启。在每个步骤或主程序调用开始时，都会创建 JCL 检查点并将其保存到内存中的注册表中。如果定义了持久性提供程序，则在作业结束时，所有步骤级别的检查点都将保留。	
`jcl.checkpoint.expireTimeout`	数字	–1	指定在持久性提供程序或内存中的注册表中保留 JCL 检查点的持续时间。	
`jcl.checkpoint.expireTimeoutUnit`	字符串	SECONDS	指定 `jcl.checkpoint.expireTimeout` 属性的持续时间单位。支持的枚举常量值：java.util.concurrent。 TimeUnit。	
`jcl.checkpoint.provider`	字符串	null	指定 JCL 检查点机制持久性提供程序。支持的值为 `redis`。	

Key	Type	默认 值	说明	发行版
`jcl.checkpoint.redis.*`	支持的 Redis 属性		指定 JCL 检查点机制的 REDIS 持久性提供程序的配置属性，请参阅the section called "支持的 Redis 属性"。	
`job.default.encoding`	字符串	CP1047	指定用于初始化作业级存储的默认编码。例如，BLU4IV 应用程序中的开关使用此编码初始化为零。	4.9.0
`spring.aws.client.jcl.checkpoint.redis.secret`	字符串	null	指定 JCL 检查点机制的 Redis 持久性提供程序的凭证密钥 ARN，请参阅 the section called "AWS 蓝光时代运行时的秘密"。	
`gapwalk.ssl.enabled`	布尔值	false	指示将以下 `gapwalk.ssl.*` 属性设置为当前 JVM 系统属性（如果应用程序启动时尚未设置）。	

Key	Type	默认 值	说明	发行版
gapwalk.ssl.trustStore	字符串	null	请将值设置为系统属性 javax.net.ssl.trustStore （如果应用程序启动时尚未设置）。	
gapwalk.ssl.trustStorePassword	字符串	null	请将值设置为系统属性 javax.net.ssl.trustStorePassword （如果应用程序启动时尚未设置）。或者，强烈建议使用 AWS 机密，如中所述。the section called “用于 SSL 密码设置的密钥管理器”	
gapwalk.ssl.trustStoreType	字符串	null	请将值设置为系统属性 javax.net.ssl.trustStoreType （如果应用程序启动时尚未设置）。	

Key	Type	默认 值	说明	发行版
gapwalk.ssl.keyStore	字符串	null	请将值设置为系统属性 javax.net.ssl.keyStore （如果应用程序启动时尚未设置）。	
gapwalk.ssl.keyStorePassword	字符串	null	请将值设置为系统属性 javax.net.ssl.keyStorePassword （如果应用程序启动时尚未设置）。或者，强烈建议使用 AWS 机密，如中所述。the section called “用于 SSL 密码设置的密钥管理器”	
mq.queues	字符串	sqs	指定使用哪个受支持的队列代理：sqs（使用 Amazon SQS）、rabbitm 用本地 Rabbit MQ）或 jms（使用本地 IBMMQ）。	

Key	Type	默认 值	说明	发行版
mq.queues.jmsMQQueueManagers[N]			如果 mq.queues 为 jms，则支持指定 IBM MQ 连接列表。对于第一个连接为 mq.queues.jmsMQQueueManagers[0] ，对于第二个连接为 mq.queues.jmsMQQueueManagers[1] ，依此类推。	
mq.queues.jmsMQQueueManagers[N].jmsMQQueueManager	字符串	null	IBMMQ 队列管理器名称。	
mq.queues.jmsMQQueueManagers[N].jmsMQAppName	字符串	null	IBMMQ 应用程序名称。	
mq.queues.jmsMQQueueManagers[N].jmsMQChannel	字符串	null	IBMMQ 频道名称。	

Key	Type	默认 值	说明	发行版
mq.queues.jmsMQQueueManagers[N].jmsMQHost	字符串	null	IBMMQ 主机名。	
mq.queues.jmsMQQueueManagers[N].jmsMQPort	数字	null	IBMMQ 端口。	
mq.queues.jmsMQQueueManagers[N].jmsMQUserid	字符串	null	IBMMQ 用户名。	
mq.queues.jmsMQQueueManagers[N].jmsMQPassword	字符串	null	IBMMQ 用户密码。或者，强烈建议使用 AWS 机密，如中所述。the section called “用于 IBM MQ 密码设置的密钥管理器”	
mq.queues.jmsMQQueueManagers[N].jmsMQMaxPoolSize	数字	0	IBMMQ 的最大池大小。如果为 0，则启用无限数量的物理连接。	

Key	Type	默认 值	说明	发行版
mq.queues.jmsMQQueueManagers[N].jmsMQSSLCipher	字符串	null	IBMMQ SSL 密码套件。例如 "*TLS12ORHIGHER" 。有关更多详细信息，请参阅 JMS CipherSuites 的官方文档 TLS CipherSpecs 和 IBM MQ 类中的文档。	
mq.queues.non.jms.client	布尔值	false	指明发送消息的目标客户端是否为非 JMS。非 JMS 客户端将使用原生 MQ 格式，而 JMS 将使用 RFH2 格式。	4.5.0
mq.connection.pool.share	布尔值	false	指明是使用同一个队列管理器共享还是重新创建 JMS 连接池。	4.9.0
mq.queues.rabbitMQHost	字符串	null	Rabbit MQ 主机名。	
mq.queues.rabbitMQVirtualHost	字符串	null	Rabbit MQ 虚拟主机名。	

Key	Type	默认 值	说明	发行版
mq.queues.rabbitMQPort	数字	null	Rabbit MQ 端口。	
mq.queues.rabbitMQUsername	字符串	null	Rabbit MQ 用户。	
mq.queues.rabbitMQPassword	字符串	null	Rabbit MQ 密码。	
mf.runtime.switch.N	布尔值	true	为 MF 自然行序文件启用空插入。	4.4.0
mf.runtime.switch.T	布尔值	false	允许在 MF 自然行序文件中插入制表符。	4.4.0
gapwalk.database.support.useSavePointToRestoreFail	布尔值	false	通过在插入查询中使用 savePoints，在出现故障时启用事务恢复。启用此属性可能会影响数据库性能。您可以使用 query-to-database映射配置为特定查询覆盖此设置。	4.6.0

Key	Type	默认 值	说明	发行版
`xmlGeneratorPrettyFormat`	布尔值	true	确定 XmlGenerator 类中生成的 XML 输出的格式样式。设置为时`true`，XML 输出将格式化为人类可读的缩进结构（打印精美）。设置为时 `false`，输出采用紧凑的单行格式，不带额外的空格。	4.8.0
`gapwalk.database.metadata.schema-inclusion.enabled`	布尔值	false	启用内存优化功能，该功能可控制应用程序缓存哪些数据库架构。此配置有助于通过智能管理架构元数据来减少内存占用，同时保持系统的稳定性。它自动包含一些常见的数据库系统架构，这些架构可能是正常操作所必需的。	4.9.0

Key	Type	默认 值	说明	发行版
gapwalk.database.metadata.schema-inclusion.schemas.global	清单 <String>	null	定义应缓存的架构列表。	4.9.0
sqlConnectionAutoCommitEnabled	布尔值	true	允许在连接重置操作期间自动提交当前连接。启用后，在非 XA 环境中切换到新的数据源连接之前，会自动提交当前连接，以防止连接不稳定。禁用后，会记录一条有关重置操作期间潜在连接问题的警告消息。	4.10.0

Key	Type	默认 值	说明	发行版
gapwalk.post.script.mediatype.json	布尔值	false	指定是否将来自端点的响应Content-Type 标头设置/post/script/{scriptId:.+}为application/json ，具体取决于布尔变量的值。如果设置为 false，则默认值为text/plain; charset=utf-8 。	
startDefaultJob	字符串	false	指定是否在第一次事务处理时初始化默认作业。	
gapwalk.cobol.cblqda.enabled	布尔值	false	启用 CBLQDA 功能，该功能可在 OPEN 语句期间控制 QSAM 文件的动态分配。此配置有助于动态分配程序中声明的临时文件，即使这些文件未在 JCL 脚本中定义。	

Key	Type	默认 值	说明	发行版
mq.queues.jms.connectionfactory.*	支持的 Atomikos 连接工厂属性		指定 JMS MQ 连接池的配置属性。	

可选 Web 应用程序的可用属性

根据您的现代化应用程序，您可能需要配置一个或多个可选 Web 应用程序，这些应用程序表示对依赖关系的支持，例如用于配置每个可选 Web 应用程序的z/OS, AS/400, or IMS/MFS. The following tables contain lists of the available key/value参数。

gapwalk-utility-pgm.war

这个可选的 Web 应用程序包含对 Z/OS 实用程序的支持。

此表提供了此应用程序 key/values 参数的详尽视图。

Key	Type	默认 值	说明	发行版
logging.config	路径	classpath:logback-utility.xml	用于引用 logback 配置文件的标准密钥。其他标准日志记录密钥也可用。	
spring.jta.enabled	布尔值	false	标准密钥。如果数据源支持模式不是 static-xa，则必须禁用 Spring JTA 事务自动配置。	
spring.datasource.primary.jndi-name	字符串	jdbc/primary	主要数据源的 Java 命名和目录接口（JNDI）	

Key	Type	默认 值	说明	发行版
			名称（如果使用 JNDI）。	
primary.datasource -driver-class-name -url -username -password	带子键的标准 Spring 数据源		如果不使用 JNDI，则包含应用程序数据库的连接信息。必须具有与现代化应用程序 YAML 文件相同的配置。 或者，强烈建议使用 AWS 机密，如中所述。[the section called “客户端数据库”]	
encoding	字符串	ASCII	实用程序中使用的编码。需要有效的编码 CP1047，IBM930	
sysPunchEncoding	字符串	ASCII	syspunch 编码字符集。需要有效的编码CP1047，IBM93 。	
systin.encoding	字符串	ASCII	SYSTIN 文件数据集的编码字符集。需要有效的编码CP1047，IBM93 。	4.5.0

Key	Type	默认 值	说明	发行版
zonedMode	字符串	EBCDIC_STRICT	对分区数据类型进行编码或解码的模式。允许的值包括：EBCDIC_STRICT 、EBCDIC_MODIFIED 、AS400.	
idcams.encoding.forced	字符串		IDCAMS 实用程序中使用的编码。需要有效的编码 CP1047，IBM930	4.4.0
unload.chunkSize	数字	0	卸载实用程序使用的区块大小。	
unload.computeRecordSizeIfNull	布尔值	false	如果未指定，则确定是否计算记录大小。如果指定，则该值保持不变。	
unload.sqlCodePointShift	数字	0	卸载实用程序的 SQL 代码点转换。运行字符转换进程。如果您的目标数据库来自 DB2 于 Postgresql，则为必填项。	
unload.columnFiller	字符串	space	卸载实用程序列填充内容。	

Key	Type	默认 值	说明	发行版
unload.varCharIsNull	布尔值	false	在 INFTILB 程序中使用此参数，如果设置为 true，则所有具有空（空格）值的不可为空的字段都将返回一个空字符串。	
unload.useDatabaseConfiguration	布尔值	false	指定是否在卸载实用程序中使用 application-main.yml 中的日期或时间配置。	
unload.format.date	字符串	MM/dd/yyyy	卸载实用程序中使用的日期格式（如果启用 unload.useDatabaseConfiguration ）。	
unload.format.time	字符串	HH.mm.ss	卸载实用程序中使用的时间格式（如果启用 unload.useDatabaseConfiguration ）。	

Key	Type	默认 值	说明	发行版
unload.format.timestamp	字符串	yyyy-MM-dd-HH.mm.ss.ssssss	卸载实用程序中使用的时间戳格式（如果启用 unload.useDatabaseConfiguration ）。	
unload.nbi.whenNull	十六进制	6F	数据库中的值为空时，要添加的空字节指示符（NBI）值。	
unload.nbi.whenNotNull	十六进制	00	数据库中的值不为空时，要添加的空字节指示符（NBI）值。	
unload.nbi.writeNullIndicator	布尔值	false	指定是否在卸载输出文件中写出空指示符。	
unload.bmc.useInto	布尔值	false	指定是否处理卸载实用程序的 INTO bmc 控制关键字。	
unload.fetchSize	数字	0	允许您在卸载实用程序中处理游标时调整提取大小。	

Key	Type	默认 值	说明	发行版
unload.noPad	布尔值	true	表示要卸载可变长度字符 (VARCHAR) 字段，而不会对最大长度进行任何填充。	4.5.0
treatLargeNumberAsInteger	布尔值	false	指定是否将大数字作为 Integer 处理。默认情况下，大数字作为 BigDecimal 处理。	
load.batchSize	数字	0	加载实用程序批次大小。	
load.format.localDate	字符串	dd.mm.yyyy\\ yyyy-mm-dd dd/MM/yyyy	要使用的加载实用程序本地日期格式。	
load.format.localTime	字符串	HH:mm:ss\ HH.mm.ss	要使用的加载实用程序本地时间格式。	
load.format.dbDate	字符串	yyyy-MM-dd	要使用的加载实用程序数据库格式。	
load.format.dbTime	字符串	HH:mm:ss	要使用的加载实用程序数据库时间。	

Key	Type	默认 值	说明	发行版
`load.sqlCodePointShift`	数字	0s	加载实用程序的 SQL 代码点转换。运行字符转换进程。如果您的目标数据库来自 DB2 于 Postgresql，则为必填项。	
`load.applyRollback`	布尔值	false	将此参数设置为 `true`，表示您希望服务在将数据加载到数据库遇到错误时回滚表更改。	
`forcedDate`	字符串		强制将日期（如果有）更改为提供的日期。	
`frozenDate`	布尔值	true	指定是否冻结日期。仅在设置了 `forcedDate` 时适用。	
`jcl.type`	字符串	MV	.jcl 文件类型。允许的值包括：`jcl`、`vse`。对于非 vse jcl，如果文件为空，IDCAMS 实用程序 PRINT/REPRO 命令将返回 4。	

Key	Type	默认 值	说明	发行版
hasGraphic	布尔值	false	INFUTILB 实用程序是否需要处理图形列。 DB2	
convertGraphicDataToFullWidth	布尔值	true	指定是否将图形数据转换为全角格式。	

gapwalk-cl-command.war

此可选的 Web 应用程序包含对 AS/400 实用程序的支持。

此表提供了此应用程序 key/values 参数的详尽视图。

Key	Type	默认 值	说明
logging.config	路径	classpath:logback-utility.xml	用于引用 logback 配置文件的标准密钥。其他标准日志记录密钥也可用。
spring.jta.enabled	布尔值	false	标准密钥。如果数据源支持模式不是 static-xa，则必须禁用 Spring JTA 事务自动配置。
spring.datasource.primary.jndi-name	字符串	jdbc/primary	主要数据源的 Java 命名和目录接口（JNDI）名称（如果使用 JNDI）。
primary.datasource+-driver-cl	带子键的标准 Spring 数据源		如果不使用 JNDI，则包含应用程序数据库的连接信息。必须具

Key	Type	默认 值	说明
ass-name + -url + -username + -password			有与现代化应用程序 YAML 文件相同的配置。 或者，强烈建议使用 AWS 机密，如中所述。the section called “客户端数据库”
encoding	字符串	ASCII	实用程序中使用的编码。预期的有效编码包含 CP1047、IBM930、ASCII、U
zonedMode	字符串	EBCDIC_STRICT	对分区数据类型进行编码或解码的模式。允许的值包括：EBCDIC_STRICT 、EBCDIC_MODIFIED 、AS400。
commands-off	字符串		要关闭的命令列表，以逗号分隔。允许的值包括：PGM_BASIC 、RCVMSG、SNDRCVF、CHGV 当你想禁用或覆盖现有程序时很有用。PGM_BASIC 是专为调试目的而设计的特定的 AWS Blu Age 运行时程序。
forcedDate	字符串		强制将日期（如果有）更改为提供的日期。

gapwalk-hierarchical-support.war

这个可选的 Web 应用程序包含 IMS/MFS 事务支持。

此表提供了此应用程序 key/values 参数的详尽视图。

Key	Type	默认 值	说明
`logging.config`	路径	classpath:logback-utility.xml	用于引用 logback 配置文件的标准密钥。其他标准日志记录密钥也可用。
`spring.jta.enabled`	布尔值	false	标准密钥。如果数据源支持模式不是 static-xa，则必须禁用 Spring JTA 事务自动配置。
`jhdb.configuration.context.encoding`	字符串		Java 分层数据库 (JHDB) 编码。预期的有效编码字符串包含 `CP1047`、`IBM930`、`ASCII`、`U`
`jhdb.checkpointPersistence`	字符串	none	检查点持久性模式。允许的值包括：`none`、`add`、`end`。使用 `add` 可在创建新检查点并将其添加到注册表后保留该检查点。使用 `end` 可在服务器关闭时保留检查点。任何其他值都会禁用持久性。请注意，每次向注册表中添加新的检查点时，所有现有的检查点都将被序列化并且文件

Key	Type	默认 值	说明
			会被擦除，而不是添加到文件中现有数据中。这样可能会对性能产生一些影响，具体取决于检查点的数量。
`jhdb.keepParent`	布尔值	false	在 IMS 插入期间是否保留 _parent 和 _logicalparent。设置为 true 时，IMS ISRT 调用将在数据库列中插入 _parent 和 _logicalparent。
`jhdb.query.timeout`	int	–1	指定任何 jhdb 事务执行的超时时间（以秒为单位）。在这段时间之后，系统将尝试中断程序。如果值为 -1（默认），则不设置超时。
`jhdb.backend`	字符串		Jhdb 后端类型。预期值为 `rdbms`
`jhdb.transaction.scope.programs`	字符串		要声明为整个交易的程序列表。用逗号 (`,`) 分隔每个程序。例如：`PCP008,PCT008`

Key	Type	默认 值	说明
metadata.datasource + -driver-class-name + -url + -username + -password + -type	字符串	带子键的标准 Spring 数据源	包含 jhdb 数据库的元数据信息，包括 psbs、dbds、连接。或者，强烈建议使用 AWS 机密。

客户端应用程序的可用属性

您的现代化应用程序可能需要对客户端 Spring 应用程序进行特定的属性配置。这些属性从运行时 JAR 文件中打包的类初始化 Bean。在该`application-profile.yaml`文件中，配置文件值是在应用程序生成期间设置的，允许您配置这些属性。下表列出了可用于配置客户端 Web 应用程序的 key/value 参数，该应用程序使用 Gapwalk 运行时中打包的类中的 Bean

Key	Type	默认 值	说明	发行版
blu4iv.dtaara.library.disable	布尔值	false	控制数据区操作上下文中库的使用。如果设置为 true，则数据区操作将禁用库的使用，但这不会影响的使用 QTemp。如果设置为 false，则在对数据区域执行 CRUD 操作时会考虑库。	4.5.0

客户端应用程序的可用属性

您的现代化应用程序可能需要对客户端 Spring 应用程序进行特定的属性配置。这些属性从运行时 JAR 文件中打包的类初始化 Bean。在该`application-profile.yaml`文件中，配置文件值是在应用程

序生成期间设置的，允许您配置这些属性。下表列出了可用于配置客户端 Web 应用程序的 key/value 参数，该应用程序使用 Gapwalk 运行时中打包的类中的 Bean

Key	Type	默认 值	说明	发行版
blu4iv.dtaara.library.disable	布尔值	false	控制数据区操作上下文中库的使用。如果设置为 true，则数据区操作将禁用库的使用，但这不会影响的使用 QTemp。如果设置为 false，则在对数据区域执行 CRUD 操作时会考虑库。	4.5.0
blu4iv.dao.data.max	数字	10	输入 DAO 操作的内部缓存的大小。此缓存存在于程序级别（单实例）。	4.9.0
blu4iv.dao.sort.function	字符串		blu4iv 数据库的排序函数名称。	4.9.0
blu4iv.dao.support	字符串	JPA	指定要使用的数据访问实现（JPA 或 JDBC）。	4.9.0
blu4iv.dao.cache.enabled	布尔值	false	启用内存缓存。	4.8.0

Key	Type	默认 值	说明	发行版
blu4iv.dao.cache.enabledHits	布尔值	false	将请求数量的跟踪添加到缓存中。	4.8.0
blu4iv.dao.cache.initMaxResults	数字	10000	缓存的大小。	4.8.0
blu4iv.dao.cache.entries.fileIds	字符串		要缓存的数据库表的列表。	4.8.0
blu4iv.dao.cache.entries.programIds	字符串		程序的标识符，应在其中为指定表激活缓存。缓存可用于执行堆栈中的所有子程序。	4.8.0
blu4iv.librarylist.enabled	布尔值	false	允许使用库列表来解析文件库。	4.10.0
blu4iv.librarylist.libraries	字符串		图书馆的初始列表从左到右排序。	4.10.0

为 AS4 00 个应用程序配置内存缓存

内存缓存功能使用户能够在内存中缓存只读数据，从而显著提高数据密集型程序的性能。缓存系统通过三种方式优化性能：仅在首次访问时加载数据、按需创建数据访问地图，以及高效地将缓存的数据存储在程序存储器附近。这种设计最大限度地减少了内存开销，同时最大限度地提高了速度。

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

```
# Cache configuration for read operations called from DAO classes #
####################################################################
blu4iv:
 dao:
   cache:
     enabled: true
     enabledHits: true
     initMaxResults: 100000
     entries:
     -
       fileIds: [${TABLEA}]
       programIds: [${PROGRAM1}]
     -
       fileIds: [${TABLEB},${TABLEC}]
       programIds: [${PROGRAM2}]
  librarylist:
         enabled: true
         libraries: ["LIBA","LIBC","LIBE"]
```

配置 API 访问日志

API 访问记录功能可记录有关 BAC、JAC 和 Gapwalk 应用程序的 API 请求的详细信息，通过跟踪谁访问了什么内容和何时访问了来帮助满足安全要求。

Key	Type	默认 值	说明	发行版
`api.logging.enabled`	布尔值	false	为 BAC、JAC 和 Gapwalk 应用程序启用 API 访问日志记录。启用后，将记录所有 API 请求并附上详细信息。	5.0.0

启用后，用户可以提供日志文件配置，以使用记录器名称在选定位置记录 API 访问日志。`api.access`日志可以包括以下 MDC 字段：

MDC Field	说明
`api-date`	发出 API 请求的日期和时间
`api-duration`	处理 API 请求花费了多长时间（毫秒）
`api-host`	API 请求的来源地址
`api-context`	API 的托管位置
`api-user`	发出 API 请求的人员的用户名或标识符
`api-sessionid`	用户当前登录会话的唯一标识符
`api-scheme`	使用的传输协议，例如 HTTP 或 HTTPS。
`api-name`	endpoint/URI 被访问的
`api-retcode`	API 请求完成后返回的状态码
`api-secure`	用于保护 API 调用的安全方法（例如，令牌 OAuth2、匿名）
`api-method`	发出的 HTTP 请求的类型（例如，GET、POST、DELETE 等）

要为 API 访问日志配置专用日志文件，请在您的日志回溯配置中添加以下内容：

```
<!-- File appender -->
<appender name="ApiAccessFile" class="ch.qos.logback.core.FileAppender">
    <file>logs/main/main-api-access.log</file>
    <immediateFlush>true</immediateFlush>
    <encoder>
        <pattern>%date %X{api-scheme} %X{api-host} %X{api-method} %X{api-name} - Return
 code: %X{api-retcode} - Start date: %X{api-date} - Duration: %X{api-duration} ms -
 User: %X{api-user} - SessionId: %X{api-sessionid} - SecureMethod: %X{api-secure} %n</
pattern>
    </encoder>
</appender>

<logger name="api.access" level="info" additivity="false">
    <appender-ref ref="ApiAccessFile" />
    <appender-ref ref="CONSOLE" />
```

```
</logger>
```

示例日志输出：

```
2025-06-13 22:24:36,174 http 127.0.0.1 GET /bac/api/services/rest/bluesamservice/
listDataSet - Return code: 200 - Start date: 06-13-2025 22:24:35.863+0200 - Duration:
 308 ms - User: velocity_admin - SessionId:
1X1X1X1X1X1X1X1X1X1X1X1X1X1X1X1X- SecureMethod: Oauth2
```

AWS 蓝光时代运行时中可用的 Redis 缓存属性

你可以使用本文档来了解 AWS Blu Age Runtime 中的 Redis 缓存、Gapwalk 配置、支持的 Redis 属性以及`application-main.yml`文件如何引用 Redis 缓存的秘密 ARN。

AWS 蓝光时代运行时中的 Redis 缓存

Redis 服务器可用作 AWS Blu Age Gapwalk 应用程序中各种功能的缓存，例如：

AWS 使用 Redis 缓存的 Blu Age 运行时功能	说明
Blusam缓存	Redis Blusam 缓存，用于使用写入策略高效读取记录，以优化批处理负载上遇到的写入密集型工作负载。
Blusam锁	用于数据集和记录的分布式锁的缓存。
数据集目录	目录数据集缓存。
会话缓存	用于的 Redis 缓存。 HttpSession缓存存储用户名、Angular 前端对话的状态以及特定的 “方言”（BMS、MFS、00）信息。 AS4
会话跟踪器	包含相关用户名和 session-creation-time信息的活动会话缓存。
JICS 缓存	JICS 资源定义的缓存。
TS 队列	TS 队列的存储。
JCL 检查点	JCL 检查点缓存。

AWS 使用 Redis 缓存的 Blu Age 运行时功能	说明
Gapwalk 文件锁	按作业分配的分布式文件锁缓存。
Blu4iv 锁	Blu4iv 记录锁的存储空间。

Redis Gapwalk 配置

如果将 `redis` 指定为缓存机制，并且没有为特定特征提供 Redis 配置，则使用全局 Redis 配置。此配置使您可以同时对多个 Redis 缓存使用相同的配置。

在以下示例中，Blusam数据集缓存和 JICS 缓存使用 `gapwalk.redis` (`redis.server1`) 配置，因为它们的缓存类型设置为`redis`，且未在和下the section called “JICS 资源定义”指定隐式 Redis 属性。the section called “JICS 资源定义”但是，Blusam锁缓存将使用不同的 Redis 配置 (`redis.server2`)，因为其 Redis 属性是明确定义的。

```
...

 gapwalk:
   redis:
     hostName: redis.server1
  port: 6379
 ...

 bluesam:
   # Redis bluesam cache
   cache: redis
   # Redis locks cache
   locks:
     cache: redis
  hostName: redis.server2
  port: 6379
 ...
 # Redis jics cache
 jics:
   resource-definitions:
     store-type: redis
 ...
```

要启用全局 Redis 配置，请在 `main-application.yml` 中添加以下配置。

```
gapwalk:
  redis:
    hostName: localhost
    port: 6379
    mode: standalone                        # Optional
    username:                               # Optional
    password: ""                            # Optional
    useSsl: false                           # Optional
    database: 0                             # Optional
    maxTotal: 128                           # Optional
    maxIdle: 128                            # Optional
    minIdle: 16                             # Optional
    testOnBorrow: true                      # Optional
    testOnReturn: true                      # Optional
    testWhileIdle: true                     # Optional
    testOnCreate: true                      # Optional
    minEvictableIdleTimeMillis: 60000       # Optional
    timeBetweenEvictionRunsMillis: 30000    # Optional
    numTestsPerEvictionRun: -1              # Optional
    blockWhenExhausted: true                # Optional
    nettyThreads: 32                        # Optional
    subscriptionsPerConnection: 10          # Optional
    subscriptionConnectionPoolSize: 100     # Optional
    pageSizeInBytes: 8192                   # Optional
    readTimeout: 2000                       # Optional
```

支持的 Redis 属性

下表显示了 AWS Blu Age Runtime 上全局缓存和特定 Redis 缓存支持的 Redis 属性。

属性名称	必填？	说明	值	默认
mode	否	Redis 的运行模式。	standalone | cluster	standalone
hostname	是	Redis 服务器的主机名或 IP 地址。	字符串	null

属性名称	必填？	说明	值	默认
`port`	是	Redis 服务器侦听连接的端口号。	int	null
`username`	否	用于身份验证的用户名。	字符串	null
`password`	否	用于身份验证的密码。	字符串	空字符串
`useSsl`	否	指定是否为 Redis 连接启用 SSL/TLS 加密。	布尔值	false
`database`	否	要使用的 Redis 数据库编号。Redis 支持多个逻辑数据库，而此属性指定要使用哪个逻辑数据库。	整数	0
`maxTotal`	否	Redis 连接池中支持的最大连接数。	int	128
`maxIdle`	否	Redis 连接池中支持的最大空闲连接数。	int	128
`minIdle`	否	要在 Redis 连接池中保持的最小空闲连接数。	int	16

属性名称	必填？	说明	值	默认
testOnBorrow	否	布尔值，指示在从池中借用连接之前是否对连接进行验证。	布尔值	true
testOnReturn	否	布尔值，指示在将连接返回到池之前是否对连接进行验证。	布尔值	true
testWhileIdle	否	布尔值，指示是否定期验证池中的空闲连接。	布尔值	true
testOnCreate	否	布尔值，指示在创建连接时是否对连接进行验证。	布尔值	true
minEvictableIdleTimeMillis	否	空闲连接在被退出之前必须在池中保持的最少时间（以毫秒为单位）。	长整数	60000L
timeBetweenEvictionRunsMillis	否	连续运行空闲连接退出器线程之间的时间（以毫秒为单位）。	长整数	30000L
numTestsPerEvictionRun	否	每次运行空闲连接退出程序线程期间要测试的最大连接数。	int	–1

属性名称	必填？	说明	值	默认
blockWhenExhausted	否	布尔值，指示在连接池耗尽时是否阻塞并等待可用的连接。	布尔值	true
nettyThreads	否	用于处理 Redis 连接的 Netty 线程数。	int	32
subscriptionsPerConnection	否	每个 Redis 连接支持的最大订阅数。	int	10
subscriptionConnectionPoolSize	否	Redis 订阅连接池中支持的最大连接数。	int	100
pageSizeInBytes	否	Redis 操作的默认页面大小（以字节为单位）。	长整数	262144000
readTimeout	否	Redis 操作的读取超时时间（以毫秒为单位）。	长整数	2000
timeToLiveMillis	否	缓存条目在被视为过期并被删除之前在缓存中停留的持续时间（以毫秒为单位）。如果未指定此属性，则默认情况下，缓存条目不会自动过期。	长整数	–1

属性名称	必填？	说明	值	默认
useAsyncBatch	否	启用 Redis 批量写入操作的异步执行以提高性能。设置为 false 时，将回退到同步执行模式。	布尔值	true
useBatchInMemoryAtomic	否	启用 Redis 批量读取操作的 In-memory-atomic 模式。设置为 false 时，会回退到默认的内存中批处理模式。	布尔值	false
connectionPoolSize	否	用于 Redisson 客户端连接到 Redis 服务器的连接池的大小。	int	64
connectionMinimumIdleSize	否	Redisson 将在其连接池中保持的最小空闲连接数。	int	24
idleConnectionTimeout	否	超时时间（以毫秒为单位），在此之后池中的空闲连接将关闭。	int	10000
connectTimeout	否	建立与 Redis 服务器的连接的超时时间（以毫秒为单位）。	int	10000

Redis 缓存属性

Redis 缓存 Blusam

```
bluesam:
   cache: redis
 # If the following redis properties are not specified gapwalk.redis configuration will
 be used for this cache
   redis:
     hostName: localhost
     port: 6379
     mode: standalone                          # Optional
     username:                                 # Optional
     password: ""                              # Optional
     useSsl: false                             # Optional
     database: 0                               # Optional
     maxTotal: 128                             # Optional
     maxIdle: 128                              # Optional
     minIdle: 16                               # Optional
     testOnBorrow: true                        # Optional
     testOnReturn: true                        # Optional
     testWhileIdle: true                       # Optional
     testOnCreate: true                        # Optional
     minEvictableIdleTimeMillis: 60000         # Optional
     timeBetweenEvictionRunsMillis: 30000      # Optional
     numTestsPerEvictionRun: -1                # Optional
     blockWhenExhausted: true                  # Optional
     nettyThreads: 32                          # Optional
     subscriptionsPerConnection: 10            # Optional
     subscriptionConnectionPoolSize: 100       # Optional
     pageSizeInBytes: 8192                     # Optional
     readTimeout: 2000                         # Optional
     timeToLiveMillis: 60000                   # Optional
     connectionPoolSize: 64                    # Optional
     connectionMinimumIdleSize: 24             # Optional
     idleConnectionTimeout: 10000              # Optional
     connectTimeout: 10000                     # Optional
```

Redis 缓存 Blusam

```
bluesam:
   locks:
     cache: redis
```

```
# If the following redis properties are not specified gapwalk.redis configuration will
be used for this cache
      hostName: localhost
      port: 6379
      mode: standalone                        # Optional
      username:                               # Optional
      password: ""                            # Optional
      useSsl: false                           # Optional
      database: 0                             # Optional
      maxTotal: 128                           # Optional
      maxIdle: 128                            # Optional
      minIdle: 16                             # Optional
      testOnBorrow: true                      # Optional
      testOnReturn: true                      # Optional
      testWhileIdle: true                     # Optional
      testOnCreate: true                      # Optional
      minEvictableIdleTimeMillis: 60000       # Optional
      timeBetweenEvictionRunsMillis: 30000    # Optional
      numTestsPerEvictionRun: -1              # Optional
      blockWhenExhausted: true                # Optional
      nettyThreads: 32                        # Optional
      subscriptionsPerConnection: 10          # Optional
      subscriptionConnectionPoolSize: 100     # Optional
      pageSizeInBytes: 8192                   # Optional
      readTimeout: 2000                       # Optional
```

会话缓存

```
spring:
  session:
    store-type: redis
# If the following redis properties are not specified gapwalk.redis configuration will
be used for this cache
jics:
  redis:
    hostName: localhost
    port: 6379
    mode: standalone                        # Optional
    username:                               # Optional
    password: ""                            # Optional
    useSsl: false                           # Optional
    database: 0                             # Optional
```

```
      maxTotal: 128                         # Optional
      maxIdle: 128                          # Optional
      minIdle: 16                           # Optional
      testOnBorrow: true                    # Optional
      testOnReturn: true                    # Optional
      testWhileIdle: true                   # Optional
      testOnCreate: true                    # Optional
      minEvictableIdleTimeMillis: 60000     # Optional
      timeBetweenEvictionRunsMillis: 30000  # Optional
      numTestsPerEvictionRun: -1            # Optional
      blockWhenExhausted: true              # Optional
      nettyThreads: 32                      # Optional
      subscriptionsPerConnection: 10        # Optional
      subscriptionConnectionPoolSize: 100   # Optional
      pageSizeInBytes: 8192                 # Optional
      readTimeout: 2000                     # Optional
```

JICS 资源定义

```
jics:
   resource-definitions:
     store-type: redis
 # If the following redis properties are not specified gapwalk.redis configuration will
 be used for this cache
   redis:
     hostName: localhost
     port: 6379
     mode: standalone                       # Optional
     username:                              # Optional
     password: ""                           # Optional
     useSsl: false                          # Optional
     database: 0                            # Optional
     maxTotal: 128                          # Optional
     maxIdle: 128                           # Optional
     minIdle: 16                            # Optional
     testOnBorrow: true                     # Optional
     testOnReturn: true                     # Optional
     testWhileIdle: true                    # Optional
     testOnCreate: true                     # Optional
     minEvictableIdleTimeMillis: 60000      # Optional
     timeBetweenEvictionRunsMillis: 30000   # Optional
     numTestsPerEvictionRun: -1             # Optional
     blockWhenExhausted: true               # Optional
```

```
    nettyThreads: 32                        # Optional
    subscriptionsPerConnection: 10          # Optional
    subscriptionConnectionPoolSize: 100     # Optional
    pageSizeInBytes: 8192                   # Optional
    readTimeout: 2000                       # Optional
```

JICS TS 队列

```
jics:
  parameters:
    tsqimpl: redis
 # If the following redis properties are not specified gapwalk.redis configuration will
 be used for this cache
  queues:
    ts:
      redis:
        hostName: localhost
        port: 6379
        mode: standalone                        # Optional
        username:                               # Optional
        password: ""                            # Optional
        useSsl: false                           # Optional
        database: 0                             # Optional
        maxTotal: 128                           # Optional
        maxIdle: 128                            # Optional
        minIdle: 16                             # Optional
        testOnBorrow: true                      # Optional
        testOnReturn: true                      # Optional
        testWhileIdle: true                     # Optional
        testOnCreate: true                      # Optional
        minEvictableIdleTimeMillis: 60000       # Optional
        timeBetweenEvictionRunsMillis: 30000    # Optional
        numTestsPerEvictionRun: -1              # Optional
        blockWhenExhausted: true                # Optional
        nettyThreads: 32                        # Optional
        subscriptionsPerConnection: 10          # Optional
        subscriptionConnectionPoolSize: 100     # Optional
        pageSizeInBytes: 8192                   # Optional
        readTimeout: 2000                       # Optional
```

会话跟踪器

```
session-tracker:
```

```
  store-type: redis
# If the following redis properties are not specified gapwalk.redis configuration will
be used for this cache
  redis:
    hostName: localhost
    port: 6379
    mode: standalone                       # Optional
    username:                              # Optional
    password: ""                           # Optional
    useSsl: false                          # Optional
    database: 0                            # Optional
    maxTotal: 128                          # Optional
    maxIdle: 128                           # Optional
    minIdle: 16                            # Optional
    testOnBorrow: true                     # Optional
    testOnReturn: true                     # Optional
    testWhileIdle: true                    # Optional
    testOnCreate: true                     # Optional
    minEvictableIdleTimeMillis: 60000      # Optional
    timeBetweenEvictionRunsMillis: 30000   # Optional
    numTestsPerEvictionRun: -1             # Optional
    blockWhenExhausted: true               # Optional
    nettyThreads: 32                       # Optional
    subscriptionsPerConnection: 10         # Optional
    subscriptionConnectionPoolSize: 100    # Optional
    pageSizeInBytes: 8192                  # Optional
    readTimeout: 2000                      # Optional
```

JCL 检查点

```
jcl:
  checkpoint:
    provider: redis
# If the following redis properties are not specified gapwalk.redis configuration will
be used for this cache
  redis:
    hostname: localhost
    port: 6379
    mode: standalone                       # Optional
    username:                              # Optional
    password: ""                           # Optional
    useSsl: false                          # Optional
    database: 0                            # Optional
```

```
      maxTotal: 128                         # Optional
      maxIdle: 128                          # Optional
      minIdle: 16                           # Optional
      testOnBorrow: true                    # Optional
      testOnReturn: true                    # Optional
      testWhileIdle: true                   # Optional
      testOnCreate: true                    # Optional
      minEvictableIdleTimeMillis: 60000     # Optional
      timeBetweenEvictionRunsMillis: 30000  # Optional
      numTestsPerEvictionRun: -1            # Optional
      blockWhenExhausted: true              # Optional
      nettyThreads: 32                      # Optional
      subscriptionsPerConnection: 10        # Optional
      subscriptionConnectionPoolSize: 100   # Optional
      pageSizeInBytes: 8192                 # Optional
      readTimeout: 2000                     # Optional
```

Gapwalk 文件锁

```
filesLocks:
   enabled: true
   retryTime: 1000
   MaxRetry: 5
   provider: redis
 # If the following redis properties are not specified gapwalk.redis configuration will
 be used for this cache
   redis:
     hostName: localhost
     port: 6379
     mode: standalone                       # Optional
     username:                              # Optional
     password: ""                           # Optional
     useSsl: false                          # Optional
     database: 0                            # Optional
     pool:
       maxTotal: 128                        # Optional
       maxIdle: 128                         # Optional
       minIdle: 16                          # Optional
       testOnBorrow: true                   # Optional
       testOnReturn: true                   # Optional
       testWhileIdle: true                  # Optional
       testOnCreate: true                   # Optional
       minEvictableIdleTimeMillis: 60000    # Optional
```

```
      timeBetweenEvictionRunsMillis: 30000    # Optional
      numTestsPerEvictionRun: -1              # Optional
      blockWhenExhausted: true                # Optional
      nettyThreads: 32                        # Optional
      subscriptionsPerConnection: 10          # Optional
      subscriptionConnectionPoolSize: 100     # Optional
      pageSizeInBytes: 8192                   # Optional
      readTimeout: 2000                       # Optional
```

Blu4iv 锁

```
blu4iv.lock: redis
blu4iv.lock.timeout: 10 #(in millisecondes)
 # If the following redis properties are not specified gapwalk.redis configuration
will be used for this cache
blu4iv.lock.redis:
      hostName: localhost
      port: 6379
      mode: standalone                        # Optional
      username:                               # Optional
      password: ""                            # Optional
      useSsl: false                           # Optional
      database: 0                             # Optional
      maxTotal: 128                           # Optional
      maxIdle: 128                            # Optional
      minIdle: 16                             # Optional
      testOnBorrow: true                      # Optional
      testOnReturn: true                      # Optional
      testWhileIdle: true                     # Optional
      testOnCreate: true                      # Optional
      minEvictableIdleTimeMillis: 60000       # Optional
      timeBetweenEvictionRunsMillis: 30000    # Optional
      numTestsPerEvictionRun: -1              # Optional
      blockWhenExhausted: true                # Optional
      nettyThreads: 32                        # Optional
      subscriptionsPerConnection: 10          # Optional
      subscriptionConnectionPoolSize: 100     # Optional
      pageSizeInBytes: 8192                   # Optional
      readTimeout: 2000                       # Optional
```

数据集目录

```
datasimplifier:
   catalogImplementation: redis
   # If the following redis properties are not specified gapwalk.redis configuration
 will be used for this cache
   redis:
     hostName: localhost
     port: 6379
     mode: standalone                          # Optional
     username:                                 # Optional
     password: ""                              # Optional
     useSsl: false                             # Optional
     database: 0                               # Optional
     maxTotal: 128                             # Optional
     maxIdle: 128                              # Optional
     minIdle: 16                               # Optional
     testOnBorrow: true                        # Optional
     testOnReturn: true                        # Optional
     testWhileIdle: true                       # Optional
     testOnCreate: true                        # Optional
     minEvictableIdleTimeMillis: 60000         # Optional
     timeBetweenEvictionRunsMillis: 30000      # Optional
     numTestsPerEvictionRun: -1                # Optional
     blockWhenExhausted: true                  # Optional
     nettyThreads: 32                          # Optional
     subscriptionsPerConnection: 10            # Optional
     subscriptionConnectionPoolSize: 100       # Optional
     pageSizeInBytes: 8192                     # Optional
     readTimeout: 2000                         # Optional
```

用于 Redis 缓存的密钥管理器

`application-main.yaml` 文件可以引用 Redis 缓存的密钥 ARN。有关如何进行集成 AWS Secrets Manager 以在运行时安全检索 Redis 连接详细信息的信息，请参阅the section called “AWS 蓝光时代运行时的秘密”。

为 Gapwalk 应用程序配置安全性

以下主题介绍如何保护 Gapwalk 应用程序。

您有责任提供正确的配置，以确保安全地使用 AWS Blu Age 框架。

默认情况下，会禁用所有与安全有关的特征。要启用身份验证（以及 CSRF、XSS、CSP 等），请将 `gapwalk-application.security` 设置为 `enabled`，将 `gapwalk-application.security.identity` 设置为 `oauth`。

主题

- 为 Gapwalk 应用程序配置 URI 可访问性
- 为 Gapwalk 应用程序配置身份验证
- 为 AWS Blu Age 运行时配置速率限制

为 Gapwalk 应用程序配置 URI 可访问性

本主题介绍如何为 Gapwalk 应用程序配置过滤。 URIs 此特征不需要身份提供者（IdP）。

要屏蔽列表 URIs，请在现代化应用程序中`application-main.yml`添加以下两行，将*URI-1URI-2*、等替换为要屏蔽 URIs 的。

```
gapwalk-application.security.filterURIs: enabled
gapwalk-application.security.blockedURIs: URI-1, URI-2, URI-3
```

为 Gapwalk 应用程序配置身份验证

要为您的 Gapwalk 应用程序配置 OAuth2 身份验证，您需要设置身份提供商 (IdP) 并将其与您的应用程序集成。本指南介绍了使用 Amazon Cognito 或 Keycloak 作为 IdP 的步骤。使用 Amazon Cognito，您可以使用 Cognito 用户池详细信息更新应用程序的配置文件。使用 Keycloak，您可以根据用户分配的 APIs 角色控制对应用程序和资源的访问权限。

主题

- 使用 Amazon Cognito 配置 Gapwalk OAuth2 身份验证
- 使用 Keycloak 配置 Gap OAuth2 walk 身份验证

使用 Amazon Cognito 配置 Gapwalk OAuth2 身份验证

本主题介绍如何使用 Amazon Cognito 作为 OAuth2 身份提供者 (IdP)，为 Gapwalk 应用程序配置身份验证。

先决条件

在本教程中，我们将使用 Amazon Cognito 作为 IdP 和 PlanetDemo 现代化项目。

您可以使用任何其他外部身份提供者。这些 ClientRegistration 信息必须从您的 IdP 处获取，并且是 Gapwalk 身份验证所必需的。有关更多信息，请参阅《Amazon Cognito 开发人员指南》。

ClientRegistration 信息：

client-id

ClientRegistration 的 ID。在我们的示例中，它将是 PlanetsDemo。

client-secret

您的客户端密钥。

授权端点

授权服务器的授权端点 URI。

令牌端点

授权服务器的令牌端点 URI。

jwks 端点

用于获取 JSON Web 密钥（JWK）的 URI，其中包含用于验证授权服务器颁发的 JSON Web 签名的密钥。

重定向 URI

授权服务器将最终用户（如果授予了访问权限）重定向到的 URI。

Amazon Cognito 设置

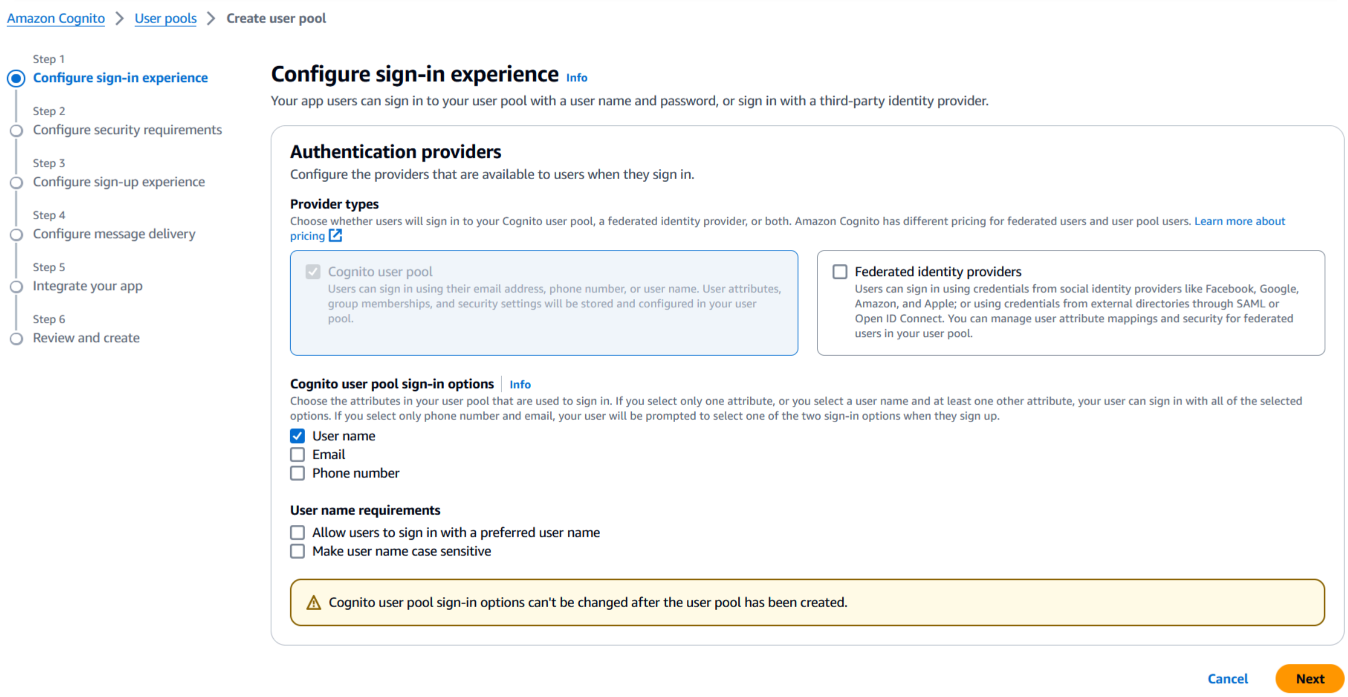
首先，我们将创建和配置一个 Amazon Cognito 用户池和用户，并将其与已部署的 Gapwalk 应用程序用于进行测试。

> **Note**
> 如果您使用其他 IdP，则可以跳过此步骤。

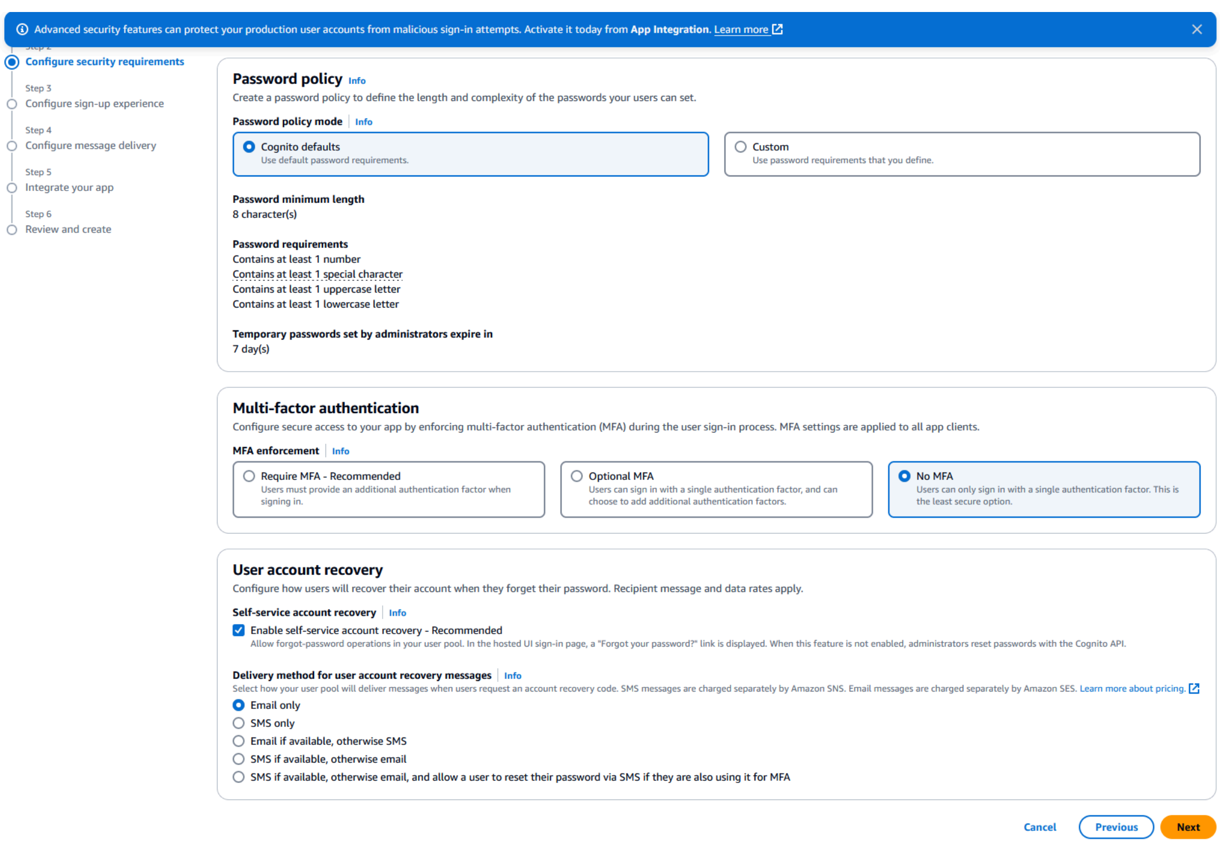
创建用户池

1. 前往中的 Amazon Cognito AWS 管理控制台 并使用您的 AWS 凭证进行身份验证。
2. 选择用户池。
3. 选择创建用户池。

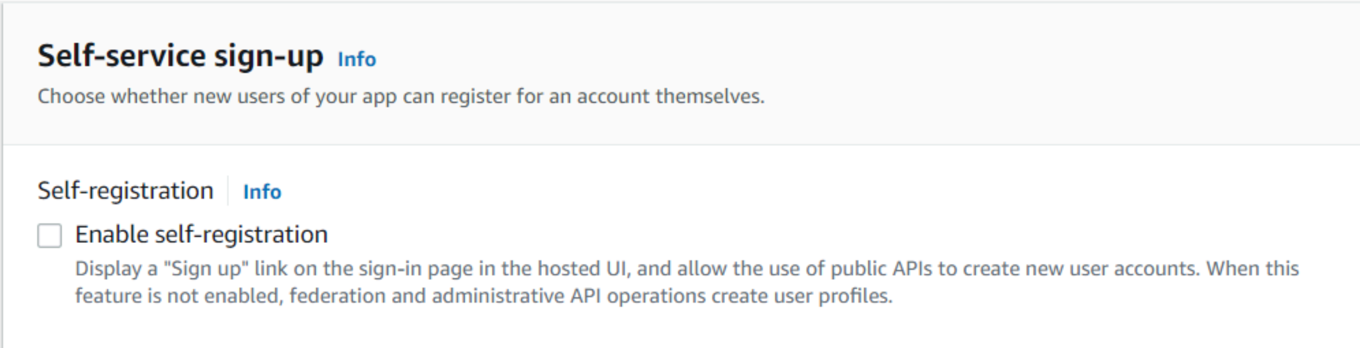
4. 在配置登录体验中，保持 Cognito 用户池的默认提供者类型。您可以选择一个或多个 Cognito 用户池登录选项；现在，选择用户名，然后选择下一步。



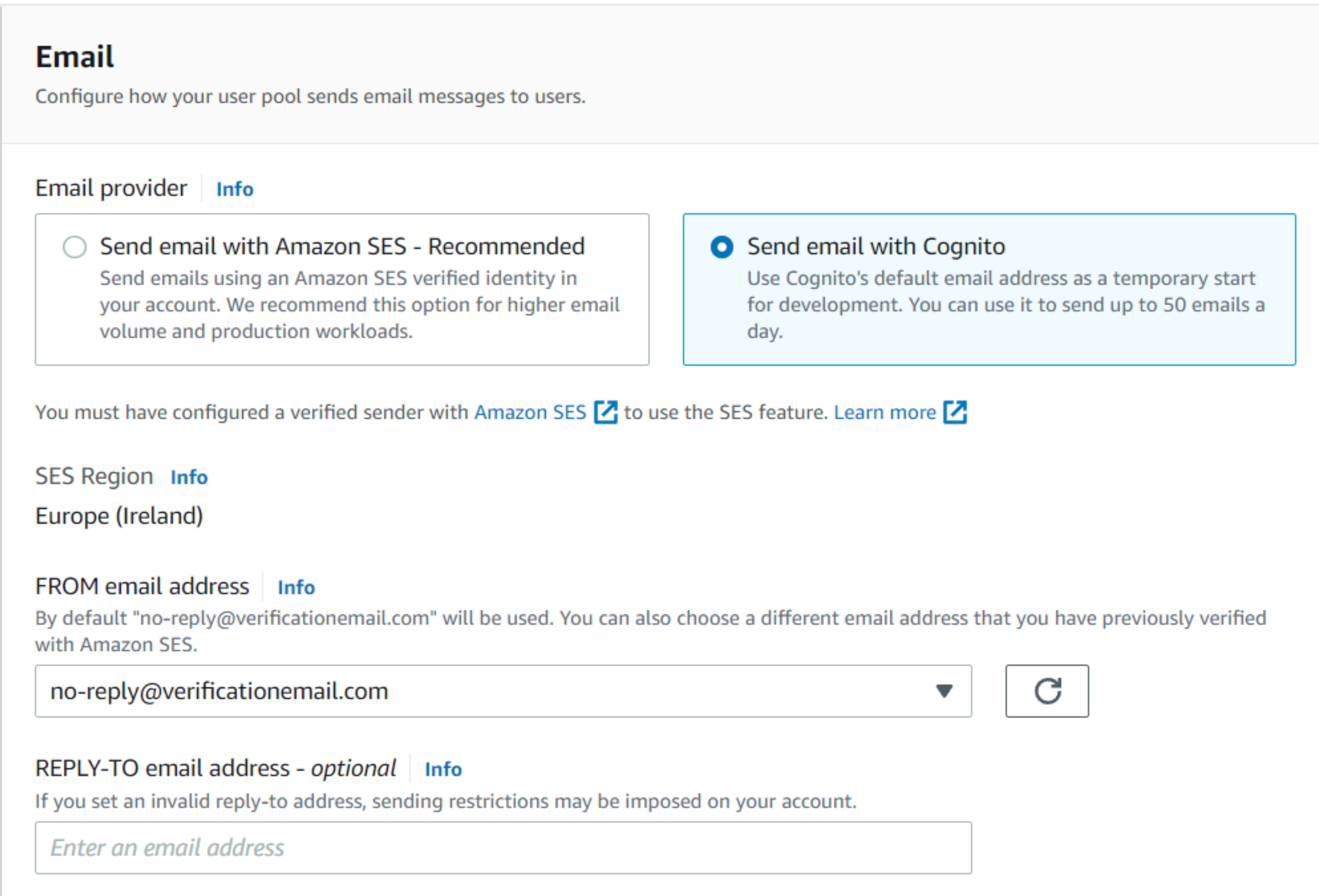
5. 在配置安全要求中，保留默认设置并通过选择无 MFA 来禁用多重身份验证，然后选择下一步。



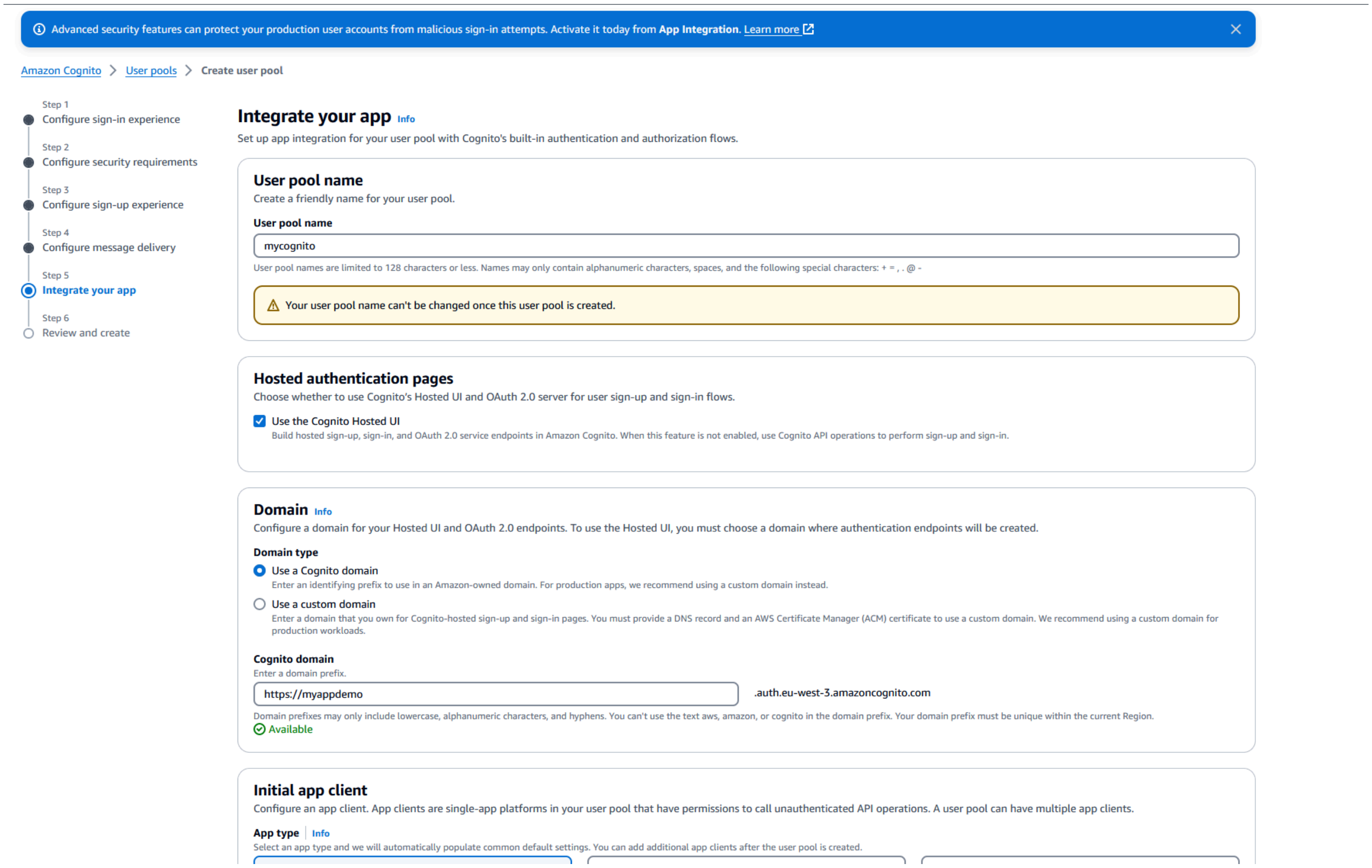
6. 作为安全措施，请禁用启用自动注册，然后选择下一步。



7. 选择使用 Cognito 发送电子邮件，然后选择下一步。



8. 在集成应用程序中，为您的用户池指定一个名称。在托管身份验证页面中，选择使用 Cognito 托管 UI。



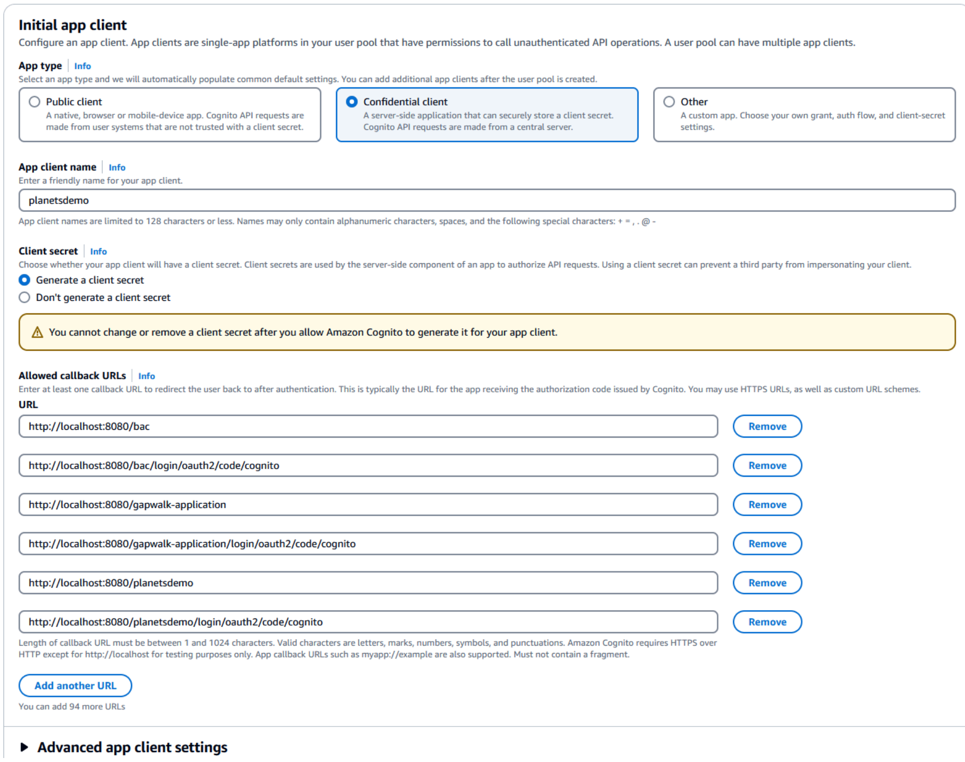
9. 为简单起见，在域中，选择使用 Cognito 域并输入域前缀；例如，`https://planetsdemo`。必须将演示应用程序添加为客户端。

   a. 在初始应用程序客户端中，选择机密客户端。输入应用程序客户端名称（例如 **`planetsdemo`**），然后选择生成客户端密钥。

   b. 在支持的回调 URL 中，输入用户在通过身份验证后要重定向到的 URL。该 URL 必须以 `/login/oauth2/code/cognito` 结尾。例如，对于我们的应用程序和后端 Gapwalk 和 BAC 应用程序：

   ```
   http://localhost:8080/bac
          http://localhost:8080/bac/login/oauth2/code/cognito
          http://localhost:8080/gapwalk-application
          http://localhost:8080/gapwalk-application/login/oauth2/code/cognito
          http://localhost:8080/planetsdemo
          http://localhost:8080/planetsdemo/login/oauth2/code/cognito
   ```

   您可以稍后编辑该 URL。



c. 在 “允许注销” 中， URLs输入您希望 Amazon Cognito 在应用程序注销用户时重定向到的注销页面的 URL。例如，对于后端 Gapwalk 和 BAC 应用程序：

```
http://localhost:8080/bac/logout
http://localhost:8080/gapwalk-application/logout
http://localhost:8080/planetsdemo/logout
```

您可以稍后编辑该 URL。

d. 在高级应用程序客户端设置和属性读写权限部分中保留默认值。

e. 选择下一步。

10. 在检查并创建中，验证您的选择，然后选择创建用户池。

有关更多信息，请参阅创建用户池。

创建用户

鉴于自助注册已禁用，请创建一个 Amazon Cognito 用户。导航到 AWS 管理控制台中的 Amazon Cognito。选择您创建的用户池，然后在用户中，选择创建用户。

在用户信息中，选择发送电子邮件邀请，输入用户名和电子邮件地址，然后选择生成密码。选择创建用户。

创建角色

在群组选项卡中，创建 3 个群组（SUPER_ADMIN、ADMIN 和 USER），并将您的用户关联到其中一个或多个群组。Gapwalk 应用程序稍后会将这些角色映射到 ROLE_SUPER_ADMIN、ROLE_ADMIN 和 ROLE_USER，从而可以访问一些受限制的 API REST 调用。

该应用程序实现了与多个 OAuth2 身份提供商配合使用的分层 scope-to-role映射。当使用 Cognito 发行的 JWT 令牌进行资源服务器授权时，令牌中定义的作用域会自动映射到相应的角色。

将 Amazon Cognito 集成到 Gapwalk 应用程序中

在您的 Amazon Cognito 用户池和用户准备就绪后，请访问您的经过现代化改造的应用程序的 `application-main.yml` 文件并添加以下代码：

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

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: "<redirect-uri>"
        provider:
          cognito:
            issuer-uri: ${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: username
      resourceserver:
```

```
        jwt:
          jwk-set-uri: ${gapwalk-application.security.issuerUri}/.well-known/jwks.json
```

按如下所述替换以下占位符：

1. 前往中的 Amazon Cognito AWS 管理控制台 并使用您的 AWS 凭证进行身份验证。
2. 选择用户池，然后选择您创建的用户池。你可以在用户池 ID ***pool-id*** 中找到。
3. 选择应用程序集成，在那里你可以找到你的*your-cognito-domain*，然后前往应用程序客户端和分析并选择你的应用程序。
4. 在 App 客户端：YouRapp 中，你可以找到*client-nameclient-id*、和*client-secret*（显示客户端密钥）。
5. *region-id*对应于您在其中创建 Amazon Cognito 用户和用户池的 AWS 区域 ID。示例：`eu-west-3`。
6. *redirect-uri*输入您为 “允许的回传 URL” 指定的 UR I。在我们的示例中，该 URI 为 `http://localhost:8080/planetsdemo/login/oauth2/code/cognito`。

您可以立即部署 Gapwalk 应用程序，并使用之前创建的用户登录您的应用程序。

使用 Keycloak 配置 Gap OAuth2 walk 身份验证

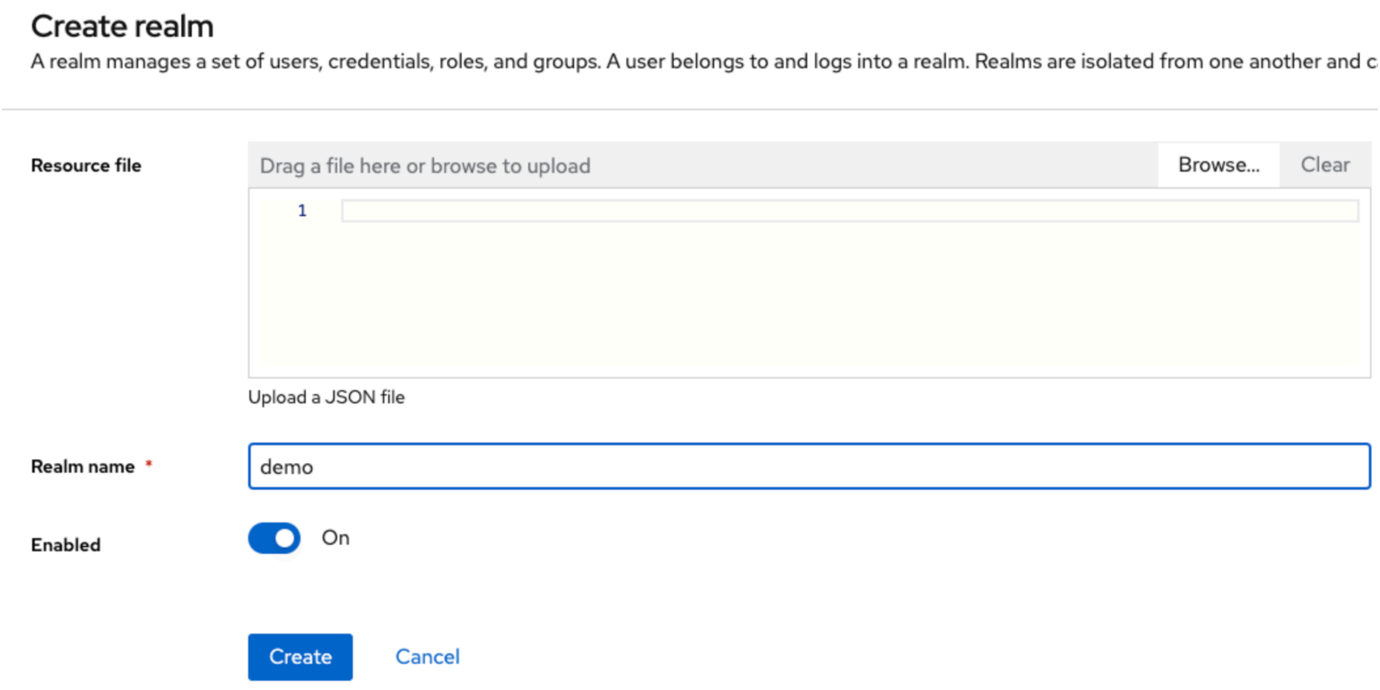
本主题介绍如何使用 Keycloak 作为 OAuth2 身份提供者 (IdP) 为 Gapwalk 应用程序配置身份验证。在本教程中，我们使用 Keycloak 24.0.0。

先决条件

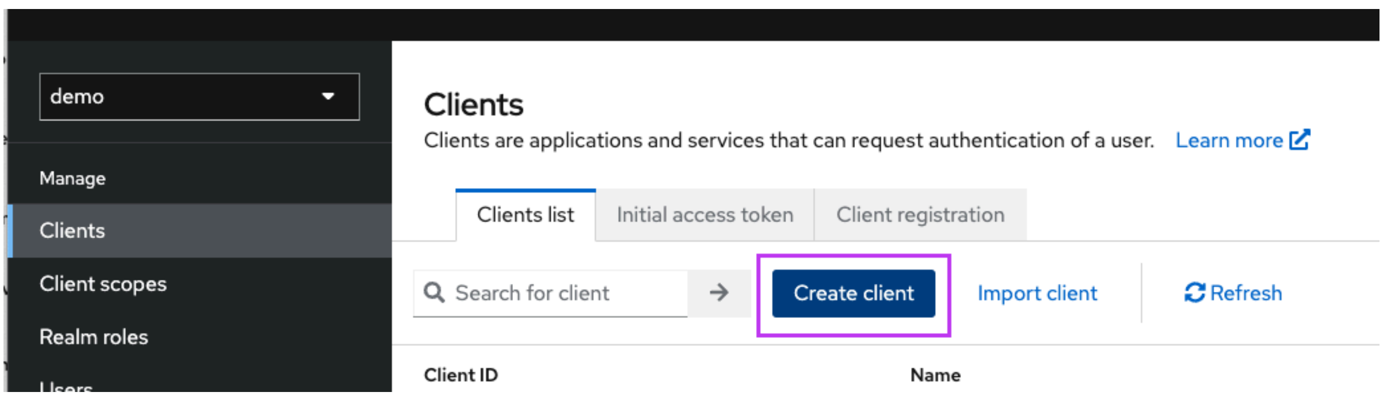
- Keycloak
- Gapwalk 应用程序

Keycloak 设置

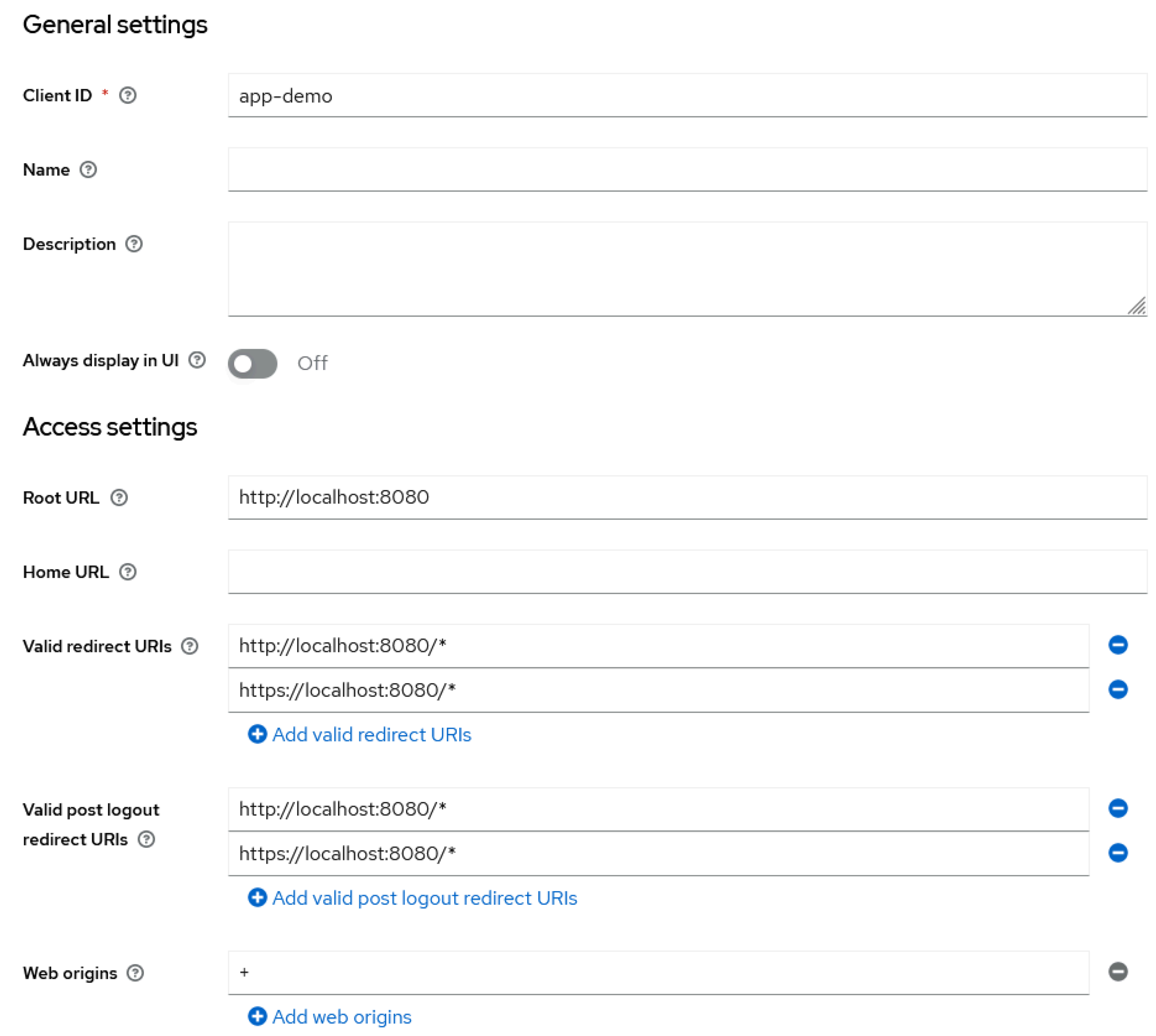
1. 在 Web 浏览器中前往 Keycloak 控制面板。默认凭证为 admin/admin。转到左上角导航栏，创建名称为 **demo** 的领域，如下图所示。

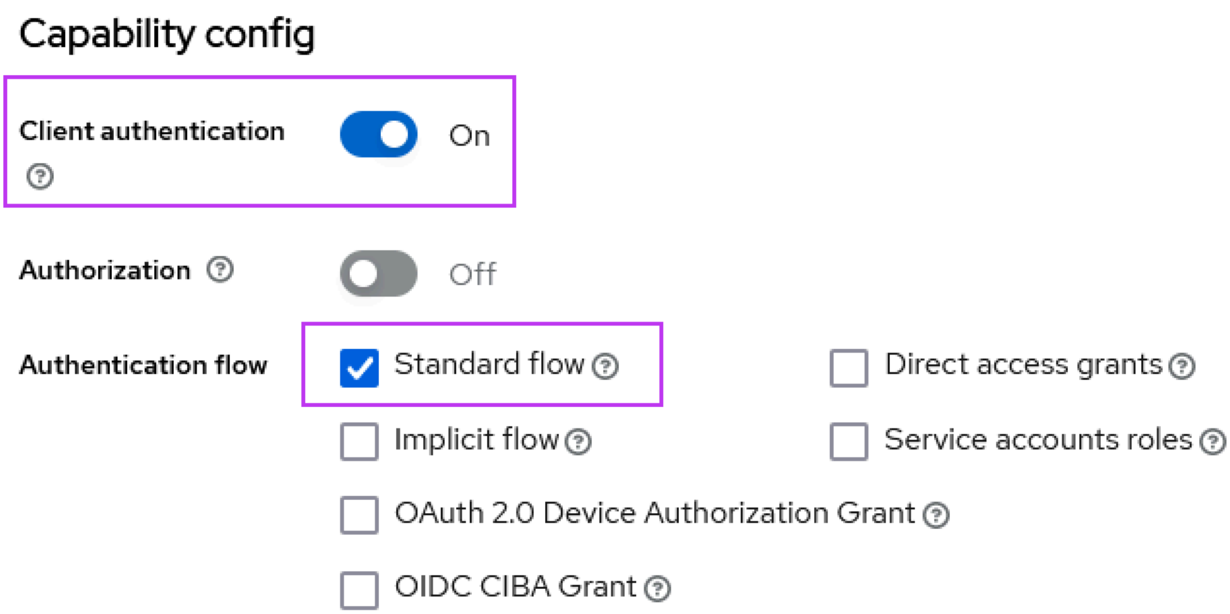


2. 创建名为 **app-demo** 的客户端。

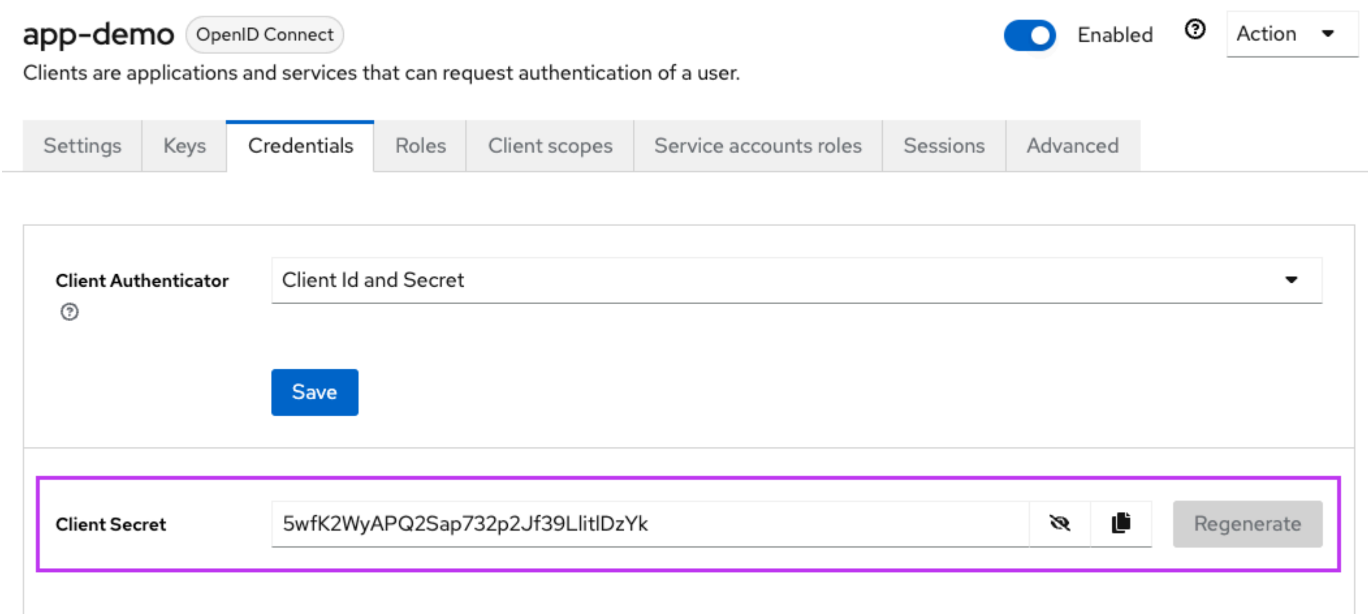


将 `localhost:8080` 替换为您的 Gapwalk 应用程序的地址

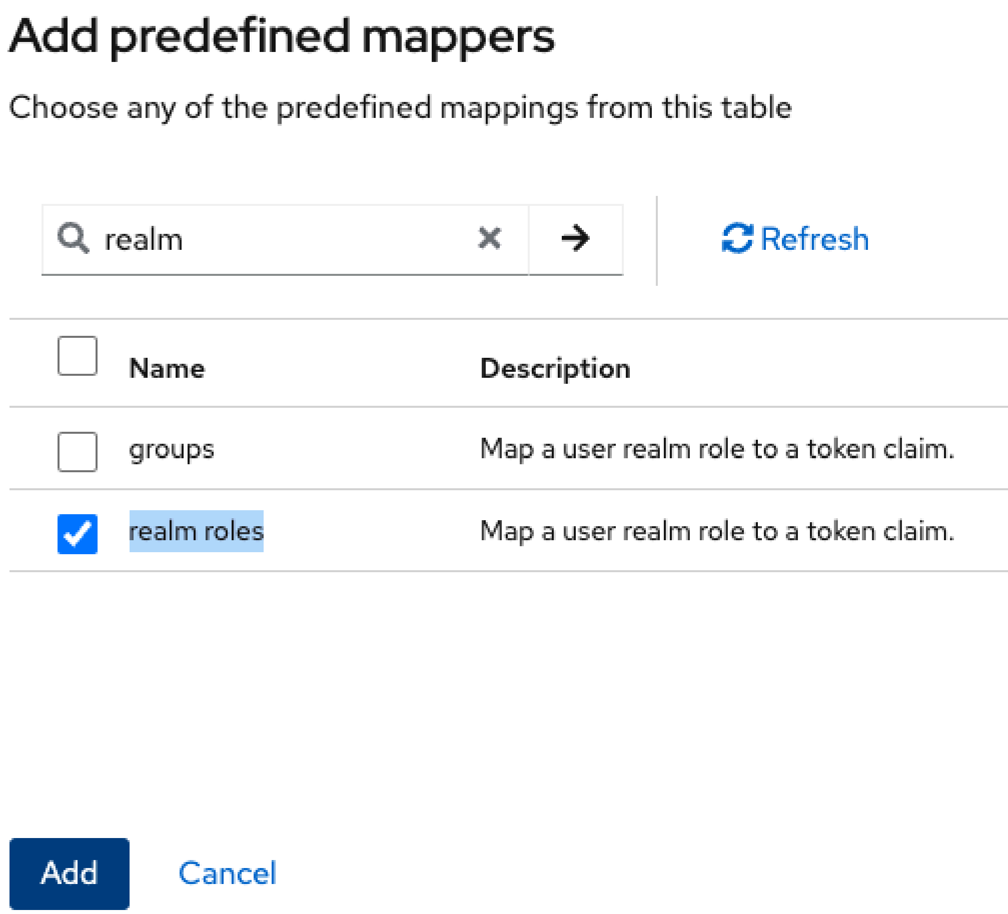

General settings
Client ID *
app-demo
Name
Description
Always display in UI
Off
Access settings
Root URL
http://localhost:8080
Home URL
Valid redirect URIs
http://localhost:8080/*
https://localhost:8080/*
Add valid redirect URIs
Valid post logout redirect URIs
http://localhost:8080/*
https://localhost:8080/*
Add valid post logout redirect URIs
Web origins
+
Add web origins



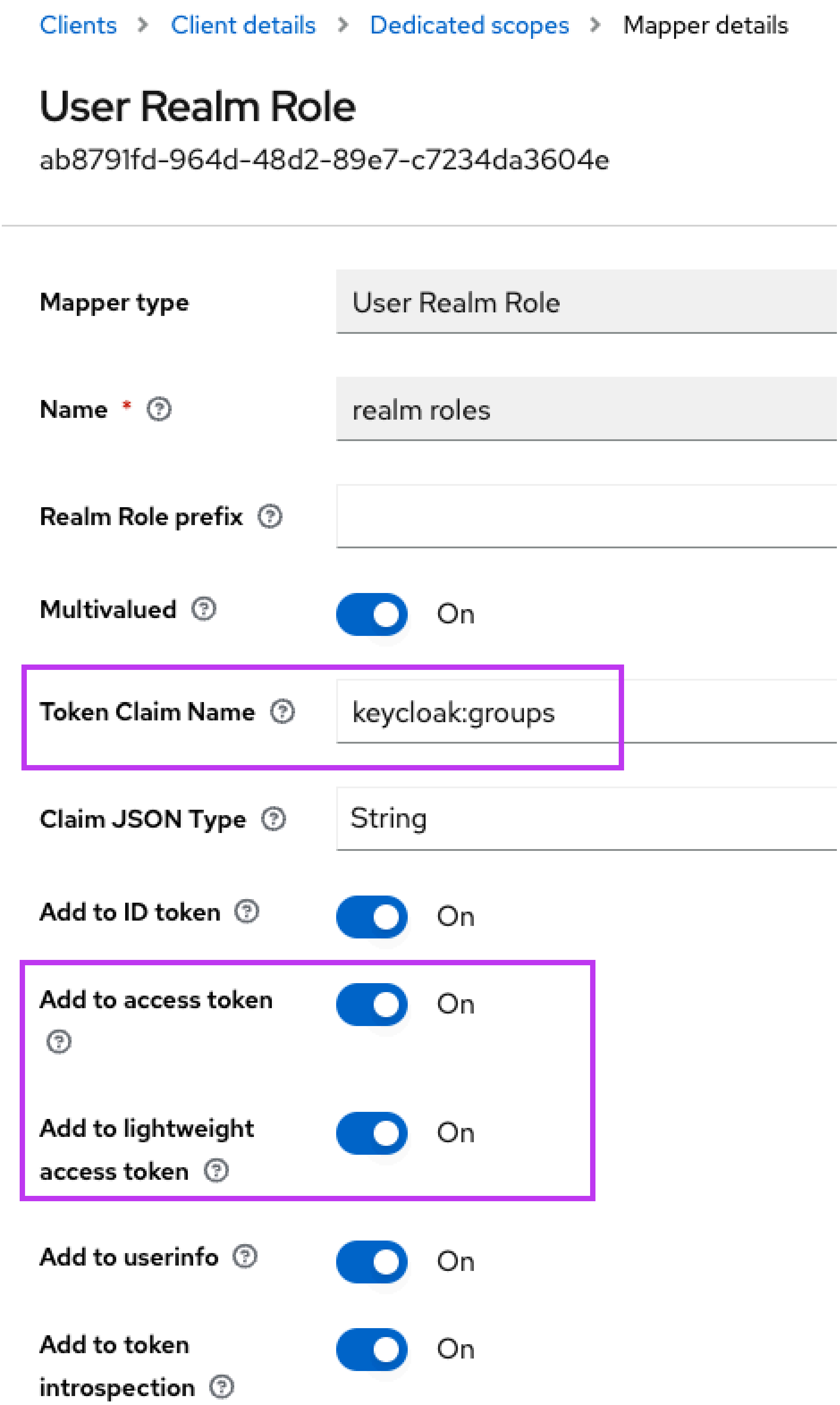
3. 要获取您的客户端密钥，请依次选择客户端、app-demo 和凭证。



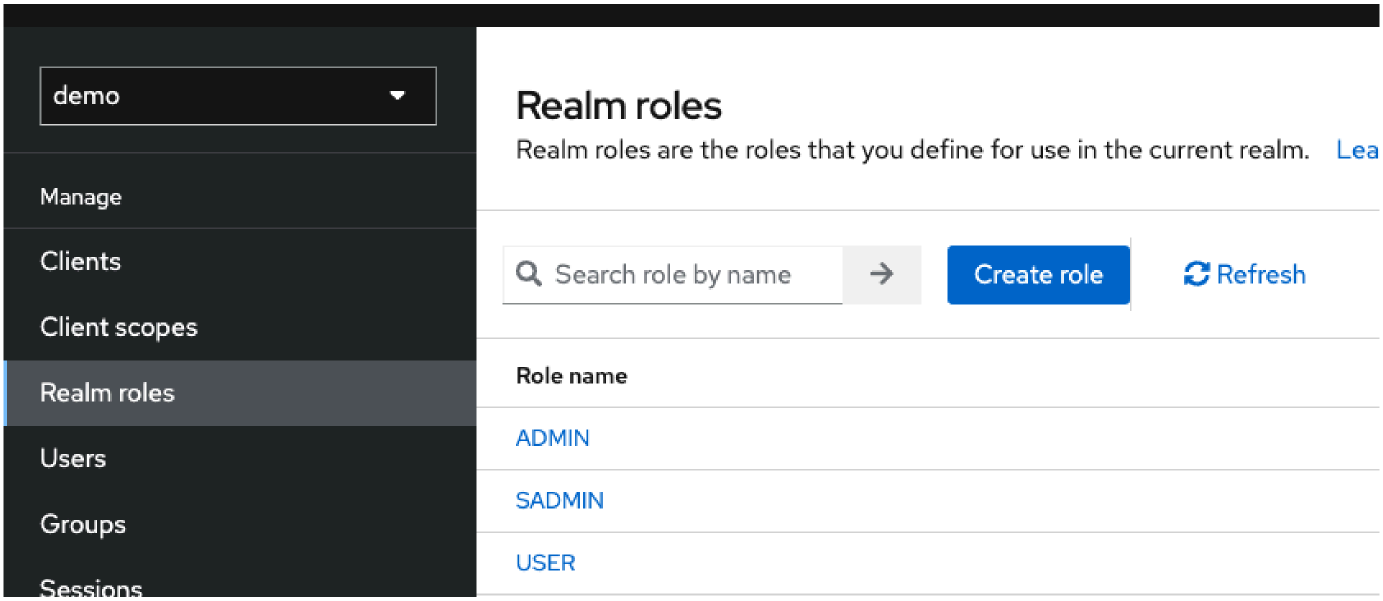
4. 依次选择客户端、客户端作用域和添加预定义映射器。选择领域角色。



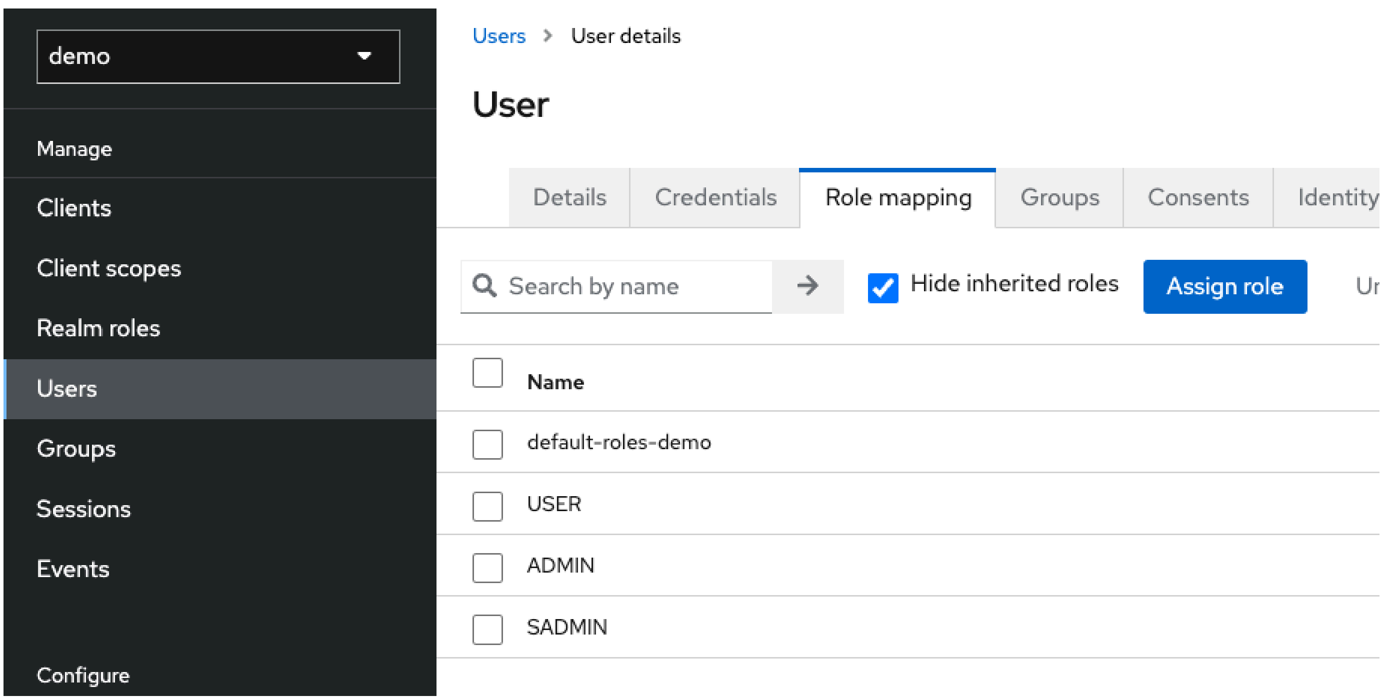
5. 使用如下图所示的配置编辑您的领域角色。



6. 记住已定义的令牌申领名称。您需要在 `gapwalk-application.security.claimGroupName` 属性的 Gapwalk 设置定义中使用此值。



7. 选择领域角色，然后创建 3 个角色：**SUPER_ADMIN**、**ADMIN** 和 **USER**。Gapwalk 应用程序稍后会将这些角色映射到 ROLE_SUPER_ADMIN、ROLE_ADMIN 和 ROLE_USER，以便能够访问一些受限制的 API REST 调用。



将 Keycloak 集成到 Gapwalk 应用程序中

按如下方式编辑您的 application-main.yml：

```
gapwalk-application.security: enabled
```

```
gapwalk-application.security.identity: oauth
gapwalk-application.security.issuerUri: http://<KEYCLOAK_SERVER_HOSTNAME>/realms/
<YOUR_REALM_NAME>
gapwalk-application.security.claimGroupName: "keycloak:groups"

gapwalk-application.security.userAttributeName: "preferred_username"
# Use "username" for cognito,
#     "preferred_username" for keycloak
#      or any other string

spring:
  security:
    oauth2:
      client:
        registration:
          demo:
            client-id: <YOUR_CLIENT_ID>
            client-name: Demo App
            client-secret: <YOUR_CLIENT_SECRET>
            provider: keycloak
            authorization-grant-type: authorization_code
            scope: openid
            redirect-uri: "{baseUrl}/login/oauth2/code/{registrationId}"
        provider:
          keycloak:
            issuer-uri: ${gapwalk-application.security.issuerUri}
            authorization-uri: ${gapwalk-application.security.issuerUri}/protocol/
openid-connect/auth
            jwk-set-uri: ${gapwalk-application.security.issuerUri}/protocol/openid-
connect/certs
            token-uri: ${gapwalk-application.security.issuerUri}/protocol/openid-
connect/token
            user-name-attribute: ${gapwalk-application.security.userAttributeName}
      resourceserver:
        jwt:
          jwk-set-uri: ${gapwalk-application.security.issuerUri}/protocol/openid-
connect/certs
```

用 Key *<YOUR_CLIENT_ID>* cloak 服务器主机名、领域名称、客户端 ID 和*<YOUR_CLIENT_SECRET>*客户端密钥替换*<KEYCLOAK_SERVER_HOSTNAME>*、、和。*<YOUR_REALM_NAME>*

为 AWS Blu Age 运行时配置速率限制

AWS Blu Age Runtime 包括内置的速率限制功能，可保护 gapwalk 应用程序免受过多请求和潜在滥用的侵害。速率限制系统使用 Token Bucket 算法来提供突发容量和持续速率限制。

主题

速率限制概述

速率限制系统提供以下功能：

代币桶算法

- 允许高达配置的突发容量的突发流量
- 根据每分钟的请求量以稳定的速度充值代币
- 在不阻塞合法流量峰值的情况下提供平滑的速率限制

客户识别

- 通过 IP 地址识别客户端，支持代理服务器
- 支架 X-Forwarded-For和 X-Real-IP标题
- 处理负载均衡器和反向代理方案

自动内存管理

- 自动清理过期的速率限制存储桶
- 可配置的清理间隔和过期时间
- 防止长时间运行的应用程序出现内存泄漏

HTTP 集成

- 超过限制时返回 HTTP 429（请求过多）
- 在响应中包含标准速率限制标题

- 为客户提供重试信息

配置属性

在application-main.yaml文件中配置速率限制：

```
gapwalk:
  ratelimiting:
    enabled: true                                    # Enable/disable rate limiting
    requestsPerMinute: 1000                          # Sustained rate limit per
 minute
    burstCapacity: 1500                              # Maximum burst requests
 allowed
    includeHeaders: true                             # Include X-RateLimit-*
 headers
    cleanupIntervalMinutes: 5                        # Cleanup interval for expired
 buckets
    bucketExpiryHours: 1                             # Hours after which unused
 buckets expire
    errorMessage: "Too many requests. Try again later."  # Custom error message
    whitelistIps: ""                                 # Comma-separated IPs to
 bypass limiting
    perEndpointLimiting: false                       # Apply limits per endpoint
 (not implemented)
```

属性描述

已启用

用于启用或禁用速率限制功能的主开关。默认值：`false`

requestsPerMinute

持续速率限制每分钟允许的请求数。这代表代币充值率。默认值：`1000`

爆发容量

在应用速率限制之前，突发中允许的最大请求数。应高于`requestsPerMinute`以允许流量激增。默认值：`1500`

包含标题

是否在 HTTP 响应中包含标准速率限制标头 (`X-RateLimit-LimitX-RateLimit-Remaining`,,`X-RateLimit-Reset`)。默认值：`true` cleanupIntervalMinutes

自动清理过期速率限制存储桶之间的间隔（以分钟为单位）。有助于防止内存泄漏。默认值：5

bucketExpiryHours

以小时为单位的时间，在此之后未使用的速率限制存储桶被视为已过期且有资格进行清理。默认值：1

errorMessage

超过速率限制时，JSON 响应中返回的自定义错误消息。默认值：`"Too many requests. Try again later."`

Whitelis提示

以逗号分隔的 IP 地址列表，这些地址完全绕过速率限制。对运行状况检查或可信系统很有用。默认值：`empty`

perEndpointLimiting

是否对每个端点应用单独的速率限制，而不是仅对每个客户端应用速率限制。目前尚未实施。默认值：`false`

启用速率限制

要使用默认设置启用速率限制，请执行以下操作：

```
gapwalk:
  ratelimiting:
    enabled: true
```

客户识别

速率限制系统使用以下优先级顺序识别客户端：

1. X-Forwarded-For 标头（如果用逗号分隔，则第一个 IP）
2. X-Real-IP 标题
3. 来自 HTTP 请求的@@ 远程地址

这样可以确保在应用程序落后时正确识别客户端：

- 负载均衡器

- 反向代理
- CDNs
- API 网关

客户识别示例

```
# Direct connection
Client IP: 192.168.1.100

# Behind load balancer with X-Forwarded-For
X-Forwarded-For: 203.0.113.45, 192.168.1.100
Client IP: 203.0.113.45 (first IP used)

# Behind reverse proxy with X-Real-IP
X-Real-IP: 203.0.113.45
Client IP: 203.0.113.45
```

速率限制标题

启用后`includeHeaders`，以下标头将添加到 HTTP 响应中：

X-RateLimit-极限

: 客户端的速率限制上限（每分钟请求数）

X-RateLimit-剩余

: 当前速率限制窗口中剩余的请求数

X-RateLimit-重置

: 速率限制窗口重置的时间（Unix 时间戳）

响应标头示例

```
X-RateLimit-Limit: 1000
X-RateLimit-Remaining: 847
X-RateLimit-Reset: 1640995200
```

超出响应速率限制

当超过速率限制时，系统会返回：

HTTP 状态

429 请求过多

Content-Type

application/json

Retry-After

重试前等待的秒数

```
{
  "error": "Rate limit exceeded",
  "message": "Too many requests. Try again later.",
  "retryAfter": 60,
  "timestamp": 1640995140000
}
```

内存管理

速率限制系统会自动管理内存，以防止长时间运行的应用程序出现泄漏：

自动清理

- 每cleanupIntervalMinutes分钟运行一次
- 移除长达数小时未使用的bucketExpiryHours存储桶
- 记录清理活动以供监视

内存效率

- 使用并发数据结构确保线程安全
- 延迟创建存储桶（仅在需要时）
- 高效的代币桶实现

监视清理活动

检查日志中是否有清理消息：

```
INFO  RateLimitingService - Cleaned up 15 expired rate limiting buckets
```

AWS Blu Age 运行时错误代码

在使用 AWS Blu Age Runtime 的应用程序的生命周期中，可能会由于各种原因而出现错误，例如配置错误或网络问题。此处记录了由 AWS Blu Age Runtime 管理的错误代码，以及错误原因和解决错误的措施。 AWS Blu Age Runtime 处理的错误代码遵循以下约定：BA-X1234，其中 X 是字母，1234 是四位数字。在本文档中，错误按第一个字母，然后按数字分类。

主题

AWS Blu Age 运行时通用错误代码

通用错误代码，前缀为。BA-A

安全配置错误

全局安全设置

键	严重性	文本	其他详细信息
BA-A0001	严重	的值无效gapwalk-application.security 。仅支持enabled或disabled的值。如果未设置，则默认情况下安全性处于禁用状态。设置gapwalk-application.security: enabled 为激	

键	严重性	文本	其他详细信息
		活全局安全功能（XSS、CORS、CSRF、OAUTH 身份验证），或者将其停用。gapwalk-application.security: disabled 移除该属性以使用默认值 (disabled)。	

身份配置

键	严重性	文本	其他详细信息
BA-A0011	警告	禁用安全功能时，gapwalk-application.security.identity 不应设置属性。当前配置不一致。要么通过设置启用安全性gapwalk-application.security: enabled ，要么移除该gapwalk-application.security.identity 属性。身份配置仅在启用安全保护时适用。	
BA-A0012	严重	缺少必需的属性gapwalk-application.security.identity 。启用安全保护后，此属性是必需的。设置gapwalk-application.security.identity: oauth 为使用 OAuth 身份验证（推荐），或者gapwalk-application.security.identity: json 在启用安全保护时使用 JSON 身份验证。	
BA-A0013	严重	的值无效gapwalk-application.security.identity 。仅支持json或oauth的值。 OAuth 是推荐的身份验证方法。设置gapwalk-	

键	严重性	文本	其他详细信息
		application.securit y.identity: oauth 为使用 OAuth 身份验证（推荐），或者gapwalk-application.securit y.identity: json 在启用安全保护时使用 JSON 身份验证。	

OAuth 配置

键	严重性	文本	其他详细信息
BA-A0031	警告	当 gapwalk-application.securit y.identity 未设置属性时，gapwalk-application.securit y.issuerUri 不应设置属性。oauth当前配置不一致。要么设置gapwalk-applicatio n.security.identity: oauth并配置发行者 URI，要么删除 Gapwalk-Application.security.issueruri 属性。发行者 URI 配置仅在使用 OAuth 身份验证时适用。	
BA-A0032	警告	缺少必需的属性gapwalk-a pplication.securit y.issuerUri 。使用 OAuth 身份验证时，此属性是必需的。使用 OAuth 身份验证 (gapwalk-applicatio n.security.identity: oauth) 时，必须设置gapwalk-applicatio n.security.issuerUri 为身份提供商 (IdP) 的 URL。	
BA-A0033	错误	的 URI 格式无效gapwalk-application.securit	

键	严重性	文本	其他详细信息
		y.issuerUri 。该值必须是有效的 URI。设置gapwalk-application.security.issuerUri 为身份提供商 (IdP) 的有效 URI。	
BA-A0041	警告	当 gapwalk-application.security.identity 未设置属性时，gapwalk-application.security.allowedOrigins 不应设置属性。oauth当前配置不一致。要么设置gapwalk-application.security.identity: oauth和配置允许的来源，要么删除 Gapwalk-Application.security.allowedOrigins 属性。允许的起源配置仅在使用 OAuth 身份验证时适用。	

URI 过滤

键	严重性	文本	其他详细信息
BA-A0101	严重	的值无效gapwalk-application.security.filterURIs 。仅支持enabled或disabled的值。如果未设置，则默认情况下将禁用 URI 过滤。设置gapwalk-application.security.filterURIs: enabled 为激活 URI 过滤，或gapwalk-application.security.filterURIs: disabled将其停用。移除该属性以使用默认值 (disabled)。	
BA-A0111	警告	禁用 URI 筛选时，gapwalk-application.securit	

键	严重性	文本	其他详细信息
		y.blockedURIs 不应设置属性。当前配置不一致。要么通过设置启用 URI 过滤gapwalk-application.filterURIs: enabled ，要么移除该gapwalk-application.security.blockedURIs 属性。URI 屏蔽配置仅在启用 URI 筛选时适用。	
BA-A0112	警告	必填属性缺失或为空gapwalk-application.security.blockedURIs 。此属性是必需的，启用 URI 筛选时不得为空。gapwalk-application.security.blockedURIs 使用逗号分隔的列表进行设置， URIs 以便在启用 URI 过滤时阻止。例如：gapwalk-application.security.blockedURIs: URI-1, URI-2, URI-3	

OAuth 安全配置

键	严重性	文本	其他详细信息
BA-A1010	严重	配置 OAuth 安全筛选器链时出错。检查您的 OAuth 配置，包括客户端注册、CORS 设置和 CSRF 保护。确保在 application-main.yml 文件中正确设置了所有必需的属性。验证您的 OAuth 提供商是否可访问且配置正确。	

AWS Secrets Manager 配置错误

键	严重性	文本	其他详细信息
BA-A2000	严重	无法解析 aws 密钥。	
BA-A2001	严重	解析密钥时出现 AWS 权限问题。	该错误显示了潜在的错误。这通常与所使用的 IAM 角色中缺少权限有关。
BA-A2002	严重	应用程序需要通过网络访问 AWS Secrets Manager 服务才能启动。这可以通过使用可公开访问的环境来解决，也可以通过创建或 NAT 网关来解决。PrivateLink	请参阅AWS Mainframe Modernization 使用 AWS PrivateLink 接口端点进行访问。
BA-A2003	严重	数据库密钥的数据库名称既没有在密钥中传递，也没有在 yaml 密钥中传递。将数据库名称添加到配置中，可以在密钥的dbname字段中或专用 yaml 密钥中。	请参阅AWS 蓝光时代运行时的秘密。

AWS 与 Blu Age 相关的运行时错误代码 Blusam

Blusam错误代码，前缀为。BA-B

键	严重性	文本	其他详细信息
BA-B0001	警告	的值无效bluesam.disabled 。true仅支持禁用Blusam。默认行为Bluesam是启用的。如果要禁用，请在配置bluesam.disabled: true中进行设置Bluesam。移除该属性以保持Blusam启用状态。	
BA-B0011	警告	的值无效bluesam.cache 。redis仅支持ehcache或缓存实现。设	

键	严重性	文本	其他详细信息
		置`bluesam.cache: ehcache` 为使用 EhCache 实现，或者设置`bluesam.cache: redis` 为使用 Redis 实现。如果您不想使用Blusam缓存，请移除该属性。	
BA-B0021	严重	Blusam处于活动状态，但配置中未定义 blusam 数据源。设置`datasource.blusamDs` 配置密钥（子密钥 url、用户名、密码...）或使用 AWS 密钥。	请参阅 Blusam 配置

AWS 与 Datasimplifier 相关的蓝光时代运行时错误代码

Datasimplifier 错误代码，前缀为 `BA-D`

Key	严重性	文本	其他详细信息
BA-D0001	严重	参数化的 GDG 生成参考无效。使用有符号整数格式`(-nnn to +nnn)`或`0`。	
BA-D1001	严重	无效的空并集：Union 必须至少包含一个子字段。确保 Union 字段的格式正确并遵循预期的结构。	

AWS 与文件相关的 Blu Age 运行时错误代码

文件错误代码，前缀为。`BA-F`这些错误与文件操作有关，包括 ESDS 和 GDG（生成数据组）。

GDG（生成数据组）

Key	严重性	文本	其他详细信息
BA-F2000	错误	无法处理 GDG 删除事件。验证 GDG 文件路径是否有效且事件队列配置正确。	

Key	严重性	文本	其他详细信息
BA-F2001	警告	无法从路径中提取文件名。验证 GDG 文件路径格式是否正确。	

AWS 与 Blu Age 相关的运行时错误代码 CL Command Programs

CL Command pgm错误代码，前缀为。BA-H

Key	严重性	文本	其他详细信息
BA-H2010	严重	命令 RMVM 中必须使用文件参数。为 RMVM 命令提供有效的文件参数。FILE 参数指定应从中删除成员的文件。	
BA-H2011	严重	命令 RMVM 中出现意外的 FILE 参数类型。确保 FILE 参数以字符串或 RecordAdaptable 类型提供。检查传递给 RMVM 的命令语法和参数类型。	
BA-H2020	警告	文件不是数据库文件。命令中忽略了锁定操作。只能锁定数据库文件。该文件是平面文件或不存在。验证文件名和库。	

AWS 与 JICS 相关的 Blu Age 运行时错误代码

JICS 错误代码，前缀为。BA-J

键	严重性	文本	其他详细信息
BA-J0001	严重	JICS XA DataSource 的配置错误。要么正确配置datasource.jicsDs.xa.data-source-class-name和其他 xa 属性，spring.jta.enabled 要么设置为false。	

AWS 与队列相关的 Blu Age 运行时错误代码

队列错误代码，前缀为。BA-Q

键	严重性	文本	其他详细信息
BA-Q0001	警告	的值无效mq.connection.pool .share 。true仅支持启用 JMS 连接池的共享。默认行为是禁用 JMS 连接池共享。如果要共享 JMS 连接池，请在配置中进行设置mq.connection.pool .share: true 。移除该属性可禁用 JMS 连接池共享。	
BA-Q1001	错误	创建连接池失败。检查池连接配置和资源可用性，并确保提供正确的凭据。	
BA-Q1002	错误	关闭工厂时出错。确认工厂未在使用中，并且所有相关资源均已正确释放。	

AWS Blu Age 运行时 Redis 错误代码

Redis 错误代码，前缀为。BA-R

Redis 配置错误

Key	严重性	文本	其他详细信息
BA-R0001	严重	缺少通过指定路径的功能的 Redis 配置。要么在给定路径上指定配置，要么定义全局 Redis 设置。	AWS 蓝光时代运行时中可用的 Redis 缓存属性
BA-R0002	严重	缺少 Redis 配置参数。请添加所需的参数。	

Key	严重性	文本	其他详细信息
BA-R0003	警告	未找到路径的显式 Redis 配置。Gapwalk Redis 配置将用于连接。如果有意思，你可以忽略。	
BA-R0004	严重	从指定路径检索 Redis 配置变量时出错。检查配置文件和环境设置。	
BA-R0005	严重	指定参数缺少 Redis 值。检查配置文件和环境设置。	

Redis 属性验证错误

Key	严重性	文本	其他详细信息
BA-R0020	错误	Redis 配置错误：参数的 Redis 配置值无效。	
BA-R0021	严重	Redis 配置错误：Redis 模式无效。必须是两种指定模式之一。	
BA-R0022	错误	设置属性时出错。检查属性值的格式和约束。	AWS 蓝光时代运行时中可用的 Redis 缓存属性
BA-R0023	错误	在属性映射中输入的属性键或值为空。确保所有属性键和值均为非空。	
BA-R0024	错误	未知或不支持的属性密钥。	AWS 蓝光时代运行时中可用的 Redis 缓存属性
BA-R0025	错误	属性的数值无效。确保该值可以转换为所需的数值类型。	

Redis 连接错误

Key	严重性	文本	其他详细信息
BA-R0081	严重	连接到 Redis 服务器失败-连接被拒绝。验证 Redis 服务器是否正在运行并且是否 host/port 正确。	
BA-R0082	严重	连接到 Redis 服务器时连接超时。检查网络连接或延长连接超时时间。	
BA-R0083	严重	连接到 Redis 服务器时读取超时。检查 Redis 服务器性能或延长读取超时时间。	
BA-R0084	严重	连接 Redis 时出错 SSL。验证 SSL 设置和证书。	
BA-R0085	严重	建立 Redis 连接失败。检查 Redis 服务器状态和配置。	
BA-R0086	严重	Redis 连接的身份验证失败。验证 Redis 身份验证凭证。	
BA-R0087	严重	Redis 数据异常。有关更多详细信息，请查看 Redis 服务器日志。	
BA-R0088 - BA-R0089	严重	连接到 Redis 时出现意外错误。查看错误消息和 Redis 配置。	
BA-R0090	严重	创建 Redis 池配置失败。查看错误消息和 Redis 配置。	

特定功能的配置错误

Key	严重性	文本	其他详细信息
BA-R0100	错误	Blusam缓存配置错误：Blusam缓存配置为使用 redis (`bluesam.cache : redis`) 缺少所需的主机名和端口配置。要么在路径上指定 Redis 配置，要`bluesam.redis` 么使用定义全局 Redis 设置。`gapwalk.redis` 如果您不想使用 redis，请移除默认`blusam.cache` 属性或将其设置为其他值。	[AWS 蓝光时代运行时中可用的 Redis 缓存属性]
BA-R0200	错误	Blusam锁配置错误：Blusam锁配置为使用 redis (`bluesam.locks : redis`)，缺少所需的主机名和端口配置。要么在路径上指定 Redis 配置，要`bluesam.locks` 么使用定义全局 Redis 设置。`gapwalk.redis` 如果您不想使用 redis，请移除默认`blusam.locks` 属性或将其设置为其他值。	[AWS 蓝光时代运行时中可用的 Redis 缓存属性]
BA-R0300	错误	Gapwalk 文件锁配置错误：Gapwalk 文件锁配置为使用 redis (`filesLocks.provider : redis`) 缺少所需的主机名和端口配置。要么在路径上指定 Redis 配置，要`filesLocks.redis` 么使用定义全局 Redis 设置。`gapwalk.redis` 如果您不想使用 redis，请移除默认`filesLocks.provider` 属性或将其设置为其他值。	[AWS 蓝光时代运行时中可用的 Redis 缓存属性]
BA-R0400	错误	数据集目录配置错误：数据集目录配置为使用 redis (`datasimplifier.catalogImplementation : redis`) 缺少所需的主机名和端口配置。	[AWS 蓝光时代运行时中可用的 Redis 缓存属性]

Key	严重性	文本	其他详细信息
		要么在路径上指定 Redis 配置，要datasimplifier.redis 么使用定义全局 Redis 设置。gapwalk.redis 如果您不想使用 redis，请移除默认datasimplifier.catalogImplementation 属性或将其设置为其他值。	
BA-R0500	错误	会话缓存配置错误：会话缓存配置为使用 redis (spring.session.store-type : redis) 缺少所需的主机名和端口配置。要么在路径上指定 Redis 配置，要jics.redis 么使用定义全局 Redis 设置。gapwalk.redis 如果您不想使用 redis，请移除默认spring.session.store-type 属性或将其设置为其他值。	AWS 蓝光时代运行时中可用的 Redis 缓存属性
BA-R0550	错误	会话跟踪器配置错误：会话跟踪器配置为使用 redis (session-tracker.store-type : redis) 缺少所需的主机名和端口配置。要么在路径上指定 Redis 配置，要session-tracker.redis 么使用定义全局 Redis 设置。gapwalk.redis 如果您不想使用 redis，请移除默认session-tracker.store-type 属性或将其设置为其他值。	AWS 蓝光时代运行时中可用的 Redis 缓存属性

Key	严重性	文本	其他详细信息
BA-R0600	错误	JICS 资源定义配置错误：JICS 资源定义配置为使用 redis (jics.resource-definitions.store-type : redis) 缺少所需的主机名和端口配置。要么在路径上指定 Redis 配置，要jics.redis 么使用定义全局 Redis 设置。gapwalk.redis 如果您不想使用 redis，请移除默认jics.resource-definitions.store-type 属性或将其设置为其他值。	AWS 蓝光时代运行时中可用的 Redis 缓存属性
BA-R0700	错误	JICS TS 队列配置错误：JICS TS 队列配置为使用 redis (jics.parameters.tsqimpl : redis) 缺少所需的主机名和端口配置。要么在路径上指定 Redis 配置，要jics.queues.ts.redis 么使用定义全局 Redis 设置。gapwalk.redis 如果您不想使用 redis，请移除默认jics.parameters.tsqimpl 属性或将其设置为其他值。	AWS 蓝光时代运行时中可用的 Redis 缓存属性
BA-R0800	错误	JCL 检查点配置错误：JCL 检查点配置为使用 redis (jcl.checkpoint.provider : redis) 缺少所需的主机名和端口配置。要么在路径上指定 Redis 配置，要jcl.redis 么使用定义全局 Redis 设置。gapwalk.redis 如果您不想使用 redis，请移除默认jcl.checkpoint.provider 属性或将其设置为其他值。	AWS 蓝光时代运行时中可用的 Redis 缓存属性

Key	严重性	文本	其他详细信息
BA-R0900	错误	blu4iv 锁配置错误：blu4iv 锁配置为使用 redis () `blu4iv.lock : redis` 缺少所需的主机名和端口配置。要么在路径上指定 Redis 配置，要`blu4iv.lock.redis` 么使用定义全局 Redis 设置。`gapwalk.redis` 如果您不想使用 redis，请移除默认`blu4iv.lock` 属性或将其设置为其他值。	AWS 蓝光时代运行时中可用的 Redis 缓存属性

Blusam Redis 缓存错误

Blusam Redis 缓存配置和连接错误

Key	严重性	文本	其他详细信息
BA-R1200	错误	无法初始化 ******* Redis 缓存。验证 Redis 连接和配置参数。	
BA-R1201	错误	无法关闭 Redis 连接。检查 Redis 连接状态并验证之前是否已建立连接。如果尝试关闭不存在的连接，则可能会出现此错误。	

缓存操作

Key	严重性	文本	其他详细信息
BA-R1300	错误	无法获取名称的缓存。验证缓存名称和 Redis 连接。	
BA-R1301	错误	无法获取名称的缓存索引。验证缓存名称和 Redis 连接。	

元数据管理

Key	严重性	文本	其他详细信息
BA-R1400	错误	无法检索缓存的元数据。检查元数据密钥和 Redis 连接。	
BA-R1401	错误	无法存储用于缓存的元数据。验证元数据对象和 Redis 连接。	
BA-R1402	错误	无法删除缓存的元数据。检查缓存名称和 Redis 连接。	

缓存预热

Key	严重性	文本	其他详细信息
BA-R1500	错误	无法检查缓存的预热状态。验证缓存名称和 Redis 连接。	
BA-R1501	错误	未能将缓存标记为已预热。检查缓存名称和 Redis 连接。	

出版商/订阅者运营

Key	严重性	文本	其他详细信息
BA-R1600	错误	无法为缓存创建元数据更新侦听器。检查 Redis pub/sub 属性： subscriptionsPerConnection （默认值：10）、 subscriptionConnectionPool大小（默认值：100）、ConnectTimeout（默认值：10000）。	
BA-R1601	错误	未能发布缓存的元数据更新。检查 Redis pub/sub 属性：ConnectTimeout（默认值：10000）、readTimeout（默认值：2000）。	

Key	严重性	文本	其他详细信息
BA-R1602	错误	未能为缓存创建缓存失效监听器。检查 Redis pub/sub 属性并根据失效负载进行调整。	
BA-R1603	错误	未能发布缓存的缓存失效。考虑增加大型失效集的超时值。	

缓存记录操作

Key	严重性	文本	其他详细信息
BA-R1710	错误	无法从缓存中检索记录。验证记录是否存在且 Redis 连接处于活动状态。	
BA-R1711	错误	无法将记录存储在缓存中。检查 Redis 连接并记录数据。	
BA-R1712	错误	未能在缓存中存储多条记录。验证 Redis 连接和批量操作设置。	
BA-R1720	错误	在分页写入期间，Batch 操作失败。检查 pageSizeIn 字节和连接超时属性。	
BA-R1721	错误	无法执行批量操作。验证 Redis 集群状态和连接设置。	
BA-R1730	错误	无法从缓存中删除记录。检查记录是否存在和 Redis 连接。	
BA-R1731	错误	Batch 删除操作部分失败。验证 Redis 集群状态并记录存在情况。	
BA-R1732	错误	清除缓存失败。检查 Redis 连接和缓存名称。	

缓存索引操作

Key	严重性	文本	其他详细信息
BA-R1801	错误	无法从缓存中检索索引。验证索引是否存在且 Redis 连接处于活动状态。	
BA-R1802	错误	未能将索引存储在缓存中。检查 Redis 连接和索引数据的完整性。	
BA-R1803	错误	处理缓存中的索引键时遇到无效的密钥格式。检查数据是否损坏或密钥模式无效。	
BA-R1804	错误	无法从缓存中检索索引键。验证 Redis 连接和缓存名称。	
BA-R1805	错误	无法从缓存中删除索引。检查索引是否存在和 Redis 连接。	
BA-R1811	警告	已跳过对无效索引键的处理。验证密钥格式和数据完整性。	

缓存失效操作

Key	严重性	文本	其他详细信息
BA-R1900	错误	无法初始化数据集的缓存失效监听器。验证配置中的订阅参数。	
BA-R1901	错误	处理数据集的缓存失效消息时出错。检查消费者实现和消息格式。	
BA-R1902	错误	收到了数据集的无效失效消息。验证消息格式和内容。	
BA-R1903	错误	无法维护数据集的订阅者连接。检查 Redis 订阅参数和连接设置。	

Blusam Redis 锁定错误代码

锁定操作

Key	严重性	文本	其他详细信息
BA-R2300	错误	Bluesam 锁定操作的参数无效。验证数据集名称和锁定参数是否为空。	
BA-R2301	错误	保存数据集锁定失败。检查 Redis 连接和锁定状态。	
BA-R2302	错误	无法保存数据集的记录锁。验证记录是否存在且 Redis 连接处于活动状态。	
BA-R2303	错误	移除数据集锁定失败。检查锁是否存在以及 Redis 连接是否处于活动状态。	
BA-R2304	错误	无法移除数据集的记录锁。验证记录和锁是否存在。	

锁定状态操作

Key	严重性	文本	其他详细信息
BA-R2310	错误	锁定状态清理参数无效。为 maxAge 参数提供正值。	
BA-R2311	警告	锁定状态完整性警告。检查锁定状态下的数据是否损坏。	
BA-R2312	错误	无法清除锁定状态。验证 Redis 连接和状态完整性。	
BA-R2313	错误	锁定状态清理操作失败。检查 Redis 连接和可用内存。	
BA-R2314	错误	无法获取数据集状态。检查 Redis 连接和数据集是否存在。	
BA-R2315	错误	无法获取数据集的记录状态。验证记录是否存在且 Redis 连接处于活动状态。	

Key	严重性	文本	其他详细信息
BA-R2316	错误	无法检查数据集写入权限。检查 Redis 连接和锁定状态。	
BA-R2317	错误	无法检查数据集的记录写入权限。验证记录和锁定状态。	
BA-R2318	错误	无法检查数据集读取权限。检查 Redis 连接和锁定状态。	
BA-R2319	错误	无法检查数据集的记录读取权限。验证记录和锁定状态。	
BA-R2320	错误	保存状态失败。检查 Redis 连接和状态数据完整性。	

锁定超时错误

Key	严重性	文本	其他详细信息
BA-R2330	错误	等待数据集写入权限时超时。考虑延长锁定超时时间或检查是否存在死锁。	
BA-R2331	错误	获取数据集锁时超时。验证是否有长时间运行的操作锁定。	
BA-R2332	错误	等待锁定数据集时中断。检查线程中断或系统问题。	
BA-R2333	错误	获取数据集记录的锁定时超时。考虑延长锁定超时时间或优化记录访问模式。	
BA-R2334	错误	等待锁定数据集中的记录时中断。调查线程中断或系统级问题。	

GWFile锁错误代码

配置和初始化错误

Key	严重性	文本	其他详细信息
BA-R3099	错误	无法关闭 L GWFile ocks Redisson 客户端。检查操作是否处于活动状态，然后重试关机。	
BA-R3101	严重	无法为 GapWalk 文件锁创建 Redission 客户端。如果使用，请通过 “FilesLocks.redis” 或全局 Redis 设置 “gapwalk.redis” 验证您的配置。	
BA-R3102	严重	对于 GapWalk 文件锁定，Redisson 客户端不能为空。如果使用，请通过 “FilesLocks.redis” 或全局 Redis 设置 “gapwalk.redis” 验证您的配置。	

GWFile锁定操作

Key	严重性	文本	其他详细信息
BA-R3200	警告	文件的锁定元数据为空。这可能会导致数据不一致并导致锁定失败或死锁。验证 Redis 数据的完整性并检查 outdated/corrupted 锁定元数据。	
BA-R3210	错误	无法检查文件的锁定状态。检查 Redis 连接和文件路径。	
BA-R3220	错误	文件锁获取中断。检查线程是否中断。	
BA-R3230	错误	无法存储文件的锁定元数据。验证 Redis 连接和内存可用性。	
BA-R3240	信息	文件已被任务锁定。等待当前操作完成。	
BA-R3250	错误	锁定文件失败。检查 Redis 连接和锁定配置。	

Key	严重性	文本	其他详细信息
BA-R3260	错误	无法解除文件锁。验证锁定是否存在且连接处于活动状态。	
BA-R3270	错误	无法解锁任务锁。检查作业上下文和 Redis 连接。	
BA-R3280	错误	无法检索作业上下文的锁。验证作业上下文是否存在和 Redis 连接。	
BA-R3290	错误	检测到文件的锁定元数据损坏。检查 Redis 数据的完整性。	

数据集目录错误代码

数据集操作

Key	严重性	文本	其他详细信息
BA-R4200	错误	注册数据集失败。检查 Redis 连接和数据集数据。	
BA-R4201	错误	无法检查数据集的存在。验证数据集标识符和 Redis 连接。	
BA-R4220	错误	检索数据集失败。检查数据集标识符和 Redis 连接。	
BA-R4221	错误	无法检索所有数据集。验证 Redis 连接并重试操作。	
BA-R4300	错误	移除数据集失败。检查 Redis 连接和数据集是否存在。	

会话和会话跟踪器错误代码

会话操作

Key	严重性	文本	其他详细信息
BA-R5050	错误	无法为会话跟踪器获取 Redis 连接。检查 Redis 连接设置和服务器状态。	
BA-R5051	错误	无法反序列化会话数据。验证会话数据格式和完整性。	
BA-R5052	错误	无法检索节点的会话。检查节点 ID 和 Redis 连接。	
BA-R5053	错误	无法保留会话。验证 Redis 连接和会话数据。	
BA-R5054	错误	无法清理会话。检查会话 ID 和 Redis 连接。	
BA-R5055	警告	未找到节点的会话。验证节点 ID 或会话可能已过期。	
BA-R5056	错误	无法将会话列表序列化为 JSON。检查会话数据的完整性。	
BA-R5057	错误	会话跟踪器操作参数无效。验证是否提供了所有必需的参数。	
BA-R5058	错误	会话的 Batch 操作失败。检查 Redis 连接并重试操作。	
BA-R5059	警告	会话过期警告。可能需要调整会话超时。	

JICS 资源定义错误代码

Redis 错误代码，前缀为 BA-R。

配置错误

Key	严重性	文本	其他详细信息
BA-R6101	错误	更新带有 ID 的 Redis 资源失败。检查 Redis 连接并验证写入权限。	
BA-R6102	错误	无法从 Redis 检索资源。检查 Redis 连接并验证资源是否存在。	
BA-R6110	错误	无法执行 Redis 模板操作。检查 Redis 连接和服务器状态。	

JICS 列表操作

Key	严重性	文本	其他详细信息
BA-R6200	错误	在 Redis 缓存中找不到 JICS 列表。验证列表名称并确保其存在于 Redis 缓存中。	
BA-R6201	错误	无法从 Redis 检索 JICS 列表。检查 Redis 连接并验证列表是否存在。	
BA-R6202	错误	检索所有 JICS 列表失败。检查 Redis 连接和服务器状态。	
BA-R6203	错误	无法保存 JICS 列表。检查 Redis 连接并验证写入权限。	
BA-R6204	错误	删除 JICS 列表失败。检查 Redis 连接并验证删除权限。	
BA-R6205	错误	未能按 ID 查找列表。验证列表 ID 并确保 Redis 连接正常。	
BA-R6206	错误	无法检索包含子女的名单。检查 Redis 连接并验证列表是否存在。	

JICS 集团运营

Key	严重性	文本	其他详细信息
BA-R6300	错误	在 Redis 缓存中找不到 JICS 组。验证组名并确保其存在于 Redis 缓存中。	
BA-R6301	错误	无法检索包含列表的 JICS 组。检查 Redis 连接并验证群组是否存在。	
BA-R6302	错误	未找到群组引用的列表。在将列表与群组关联之前，请先验证该列表是否存在。	
BA-R6303	错误	无法检索 JICS 列表的组列表。检查 Redis 连接并验证列表是否存在。	
BA-R6304	错误	无法为组检索。检查 Redis 连接并验证资源是否存在。	
BA-R6305	错误	无法检索包含所有子女的群组。检查 Redis 连接并验证群组是否存在。	
BA-R6306	错误	无法检索所有 JICS 组。检查 Redis 连接和服务器状态。	
BA-R6307	错误	未能在列表中找到群组。检查 Redis 连接并验证列表是否存在。	
BA-R6308	错误	无法检查群组是否存在。检查 Redis 连接和服务器状态。	
BA-R6309	错误	无法对群组进行计数。检查 Redis 连接和服务器状态。	
BA-R6310	错误	向列表添加群组失败。检查 Redis 连接并验证写入权限。	
BA-R6311	错误	无法保留 JICS 组。检查 Redis 连接并验证写入权限。	

Key	严重性	文本	其他详细信息
BA-R6312	错误	无法为群组删除。检查 Redis 连接并验证删除权限。	
BA-R6313	错误	删除 JICS 组失败。检查 Redis 连接并验证删除权限。	

JICS 文件操作

Key	严重性	文本	其他详细信息
BA-R6400	错误	在组中找不到文件。验证文件名并确保它存在于指定的组中。	
BA-R6401	错误	无法获取组中的文件 ID。检查 Redis 连接并验证文件是否存在。	
BA-R6402	错误	无法获取活动文件 IDs。检查 Redis 连接并验证文件是否存在。	
BA-R6403	错误	无法按名称查找文件。检查 Redis 连接并验证文件是否存在。	
BA-R6404	错误	无法按 ID 查找文件。检查 Redis 连接并验证文件是否存在。	
BA-R6405	错误	未能在组中找到文件。检查 Redis 连接并验证群组是否存在。	
BA-R6406	错误	无法从组中获取第一个文件。检查 Redis 连接并验证文件是否存在。	
BA-R6407	错误	无法保留文件。检查 Redis 连接并验证写入权限。	
BA-R6408	错误	删除带有 ID 的文件失败。检查 Redis 连接并验证删除权限。	

Key	严重性	文本	其他详细信息
BA-R6409	错误	未找到 ID 为的要删除的文件。在尝试删除之前，请验证文件 ID 是否存在。	
BA-R6410	错误	无法初始化组的文件列表。检查 Redis 连接并验证群组是否存在。	

JICS 计划运营

Key	严重性	文本	其他详细信息
BA-R6500	错误	在群组中找不到程序。验证程序名称并确保它存在于指定的组中。	
BA-R6501	错误	无法获取群组中的程序 ID。检查 Redis 连接并验证程序是否存在。	
BA-R6502	错误	无法获得激活的程序 IDs。检查 Redis 连接并验证程序是否存在。	
BA-R6503	错误	无法按名称查找程序。检查 Redis 连接并验证程序是否存在。	
BA-R6504	错误	无法通过 ID 找到程序。检查 Redis 连接并验证程序是否存在。	
BA-R6505	错误	未能在组中找到程序。检查 Redis 连接并验证群组是否存在。	
BA-R6506	错误	无法从群组中获取第一个程序。检查 Redis 连接并验证程序是否存在。	
BA-R6507	错误	无法保留程序。检查 Redis 连接并验证写入权限。	
BA-R6508	错误	删除带有 ID 的程序失败。检查 Redis 连接并验证删除权限。	

Key	严重性	文本	其他详细信息
BA-R6509	错误	找不到要删除的 ID 的程序。在尝试删除之前，请验证程序 ID 是否存在。	
BA-R6510	错误	无法初始化组的程序列表。检查 Redis 连接并验证群组是否存在。	

JICS 交易操作

Key	严重性	文本	其他详细信息
BA-R6600	错误	在群组中找不到交易。验证交易名称并确保其存在于指定的组中。	
BA-R6601	错误	未能获取群组中的交易 ID。检查 Redis 连接并验证交易是否存在。	
BA-R6602	错误	无法获得活跃的交易 IDs。检查 Redis 连接并验证交易是否存在。	
BA-R6603	错误	未能按名称查找交易。检查 Redis 连接并验证交易是否存在。	
BA-R6604	错误	无法通过 ID 查找交易。检查 Redis 连接并验证交易是否存在。	
BA-R6605	错误	未能在组中找到交易。检查 Redis 连接并验证群组是否存在。	
BA-R6606	错误	无法保留交易。检查 Redis 连接并验证写入权限。	
BA-R6607	错误	删除带有 ID 的交易失败。检查 Redis 连接并验证删除权限。	
BA-R6608	错误	无法初始化组的事务列表。检查 Redis 连接并验证群组是否存在。	

JICS 运营 TDQueue

Key	严重性	文本	其他详细信息
BA-R6700	错误	TDQueue 未在群组中找到。验证该 TDQueue 名称并确保其存在于指定的组中。	
BA-R6701	错误	未能获取组内的 TDQueue ID。检查 Redis 连接并验证是否 TDQueue 存在。	
BA-R6702	错误	无法激活 TDQueue IDs。检查 Redis 连接并验证是否 TDQueue 存在。	
BA-R6703	错误	未能 TDQueues 按名称查找。检查 Redis 连接并验证是否 TDQueues 存在。	
BA-R6704	错误	未能按 ID TDQueue 查找。检查 Redis 连接并验证是否 TDQueue 存在。	
BA-R6705	错误	未能 TDQueues 在群组中找到。检查 Redis 连接并验证群组是否存在。	
BA-R6706	错误	未能坚持下去 TDQueue。检查 Redis 连接并验证写入权限。	
BA-R6707	错误	TDQueue 使用 ID 删除失败。检查 Redis 连接并验证删除权限。	
BA-R6708	错误	无法初始化组的 TDQueue 列表。检查 Redis 连接并验证群组是否存在。	

JICS 运营 TSModel

Key	严重性	文本	其他详细信息
BA-R6800	错误	TSModel 未在群组中找到。验证该 TSModel 名称并确保其存在于指定的组中。	

Key	严重性	文本	其他详细信息
BA-R6801	错误	未能获取组内的 TSModel ID。检查 Redis 连接并验证是否 TSModel 存在。	
BA-R6802	错误	无法激活 TSModel IDs。检查 Redis 连接并验证是否 TSModel 存在。	
BA-R6803	错误	未能 TSModels 按名称查找。检查 Redis 连接并验证是否 TSModels 存在。	
BA-R6804	错误	未能按 ID TSModel 查找。检查 Redis 连接并验证是否 TSModel 存在。	
BA-R6805	错误	未能 TSModels 在群组中找到。检查 Redis 连接并验证群组是否存在。	
BA-R6806	错误	未能坚持下去 TSModel。检查 Redis 连接并验证写入权限。	
BA-R6807	错误	TSModel 使用 ID 删除失败。检查 Redis 连接并验证删除权限。	
BA-R6808	错误	无法初始化组的 TSModel 列表。检查 Redis 连接并验证群组是否存在。	

身份验证错误

Key	严重性	文本	其他详细信息
BA-R6900	错误	无法获取连接的用户名。检查安全上下文和身份验证状态。	
BA-R6901	错误	访问 Redis 资源时出现身份验证错误。验证用户权限和身份验证配置。	

TS Queue Redis 错误代码

TS 队列常规操作

Key	严重性	文本	其他详细信息
BA-R7000	警告	TS 队列操作警告：从任务范围的注册表中获取，而不是 Redis 存储。如果您想使用 Redis TS 队列存储，请对其进行配置。	

配置错误

Key	严重性	文本	其他详细信息
BA-R7110	错误	Jics TS Queue Redis 初始化错误：无法初始化列表操作。验证 Redis 模板配置。	

队列操作

Key	严重性	文本	其他详细信息
BA-R7711	错误	无法从 TS 队列读取。验证队列是否存在和数据完整性。	
BA-R7712	错误	写入 TS 队列失败。检查 Redis 连接和写入权限。	
BA-R7713	错误	删除 TS 队列失败。验证队列是否存在并删除权限。	
BA-R7714	错误	TS 队列列表操作失败。检查 Redis 列表操作的可用性。	
BA-R7715	错误	TS 队列的索引访问权限无效。验证索引是否在范围内。	
BA-R7716	警告	TS 队列大小操作警告。检查队列是否存在和连接。	

AWS Blu Age 运行时 JCL 检查点错误代码

配置错误

Key	严重性	文本	其他详细信息
BA-R8110	错误	JCL Checkpoint Redis 初始化错误：无法初始化列表操作。验证 Redis 模板配置。	

检查点检索操作

Key	严重性	文本	其他详细信息
BA-R8200	错误	无法检索作业的 JCL 检查点。检查 Redis 连接和任务名称。	
BA-R8210	错误	从持久性中检索 JCL 检查点时出错。检查 Redis 连接和任务名称。	
BA-R8600	警告	未找到工作检查点。验证作业名称和检查点是否存在。	

检查点持久化操作

Key	严重性	文本	其他详细信息
BA-R8300	错误	无法为作业保留 JCL 检查点。验证 Redis 连接和写入权限。	
BA-R8305	错误	无法为任务检查点设置过期时间。验证 Redis 连接和过期设置。	
BA-R8400	错误	检查点反序列化失败。验证检查点数据的完整性。	

检查点清除操作

Key	严重性	文本	其他详细信息
BA-R8500	错误	无法从 redis 持久化中清除 JCL 检查点。检查 Redis 连接和删除权限。	
BA-R8501	错误	无法从 redis 持久化中清除 JCL 检查点。检查 Redis 连接和删除权限。	

blu4iv 锁定错误代码

锁定操作

Key	严重性	文本	其他详细信息
BA-R9200	错误	无法保存 Blu4iv 锁以备记录。检查 Redis 连接并锁定数据。	
BA-R9201	错误	无法检查锁定存储库状态。验证 Redis 连接和存储库状态。	
BA-R9210	错误	无法检查锁定权限。验证 Redis 连接和超时设置。	
BA-R9220	错误	无法移除 blu4iv 锁。检查 Redis 连接并锁定是否存在。	
BA-R9225	错误	无法清除 blu4iv 锁。检查 Redis 连接和权限。	
BA-R9900	警告	锁已存在以供记录在案。验证锁定状态和记录标识符。	

AWS 与实用程序相关的 Blu Age 运行时错误代码

实用程序 pgm 错误代码，前缀为。BA-U

Key	严重性	文本	其他详细信息
BA-U1001	警告	输入 lilian 秒刻度超过最大刻度。将改用最大缩放比例。	
BA-U2001	错误	处理遇到的 TrailerItem 子类失败。确保支持拖车物品。	
BA-U2002	错误	无法解析 TrailerItem 参数。确保支持预告片物品参数。	
BA-U2003	错误	无法处理 DFSORT 日期格式参数。确保支持日期格式参数。	
BA-U2004	错误	从中提取 Header1 时输入无效。SortGrammar确保 Header1 字符串的格式正确且符合预期的结构。	
BA-U2005	错误	处理时出现意外错误 SortGrammar。查看异常详细信息并验证输入完整性和语法解析逻辑。	
BA-U2006	错误	反馈代码 CEE2 F3：无法检索当前系统的日期和时间。检查系统时钟配置和时区设置。	
BA-U2007	错误	反馈代码 CEE2 EC：无法解析传递给 CEEDAYS 或 CEESECS 的日期值。确保支持日期格式。	
BA-U2008	错误	反馈码 E CEE2 H： date/time 在 CEEISEC、CEEDAYS 或 CEESECS 呼叫中输入的内容不在有效范围内。确保输入日期在支持的范围内。	
BA-U2009	错误	反馈代码 CEE2 EM：无法解析调用 date/time 服务时输入的图片字符串。确保支持图片格式。	

Key	严重性	文本	其他详细信息
BA-U2010	错误	反馈码 CEE2例如:输入的莉莲日期值不在有效范围内. 确保输入 lilian 日期在支持的范围内。	
BA-U2011	严重	收到的参数数量与预期的参数数量不匹配。确保向被调用的程序传递了正确数量的参数。	
BA-U2012	错误	无法处理 DFSORT 时间自变量。确保参数的格式正确。	
BA-U2013	错误	无法处理 DFSORT TIMENS 参数。确保参数的格式正确。	
BA-U3001	错误	解析 INZUTILB 程序的 SYSIN 时出错。验证 SYSIN 语法，如果需要支持，请联系支持人员。	
BA-U3002	错误	无法检测数据库类型。默认为 PostgreSQL。验证数据库连接、驱动程序配置，并确保加载了正确的 JDBC 驱动程序。	
BA-U3003	错误	INZUTILB 卸载操作失败。查看 INZUTILB 实用程序的 SYSIN 命令语法，验证数据库连接。	
BA-U3004	错误	SYSIN 输入为空或不包含有效命令。向 SYSIN 添加至少一个有效命令（卸载、选择等）。	
BA-U3005	错误	缺少必需的参数 OUTDDN。在 SYSIN 中使用 OUTDDN (ddname) 命令指定输出数据集。	
BA-U3006	错误	不支持数据库类型。使用支持的数据库之一：PostgreSQL、Oracle 或。 DB2	

Key	严重性	文本	其他详细信息
BA-U3007	错误	运行 INZUTILB 程序时，未找到该数据库类型的格式化程序。分析是否应考虑该类型并联系支持人员进行分析。	
BA-U3008	错误	所选输出格式未处理格式化，而是使用 DSNTIAUL 格式。检查控制卡中的输出格式，如果需要实现，请联系支持人员。	
BA-U3009	错误	PIC 选项上的掩码参数无效。验证您的控制卡并使用有效的口罩。	
BA-U3010	错误	PIC 选项上的符号参数无效。可接受的值为 '-'、'+'、'P' 和 'V'。验证您的控制卡并使用有效的参数。	
BA-U3012	错误	无法解析 INZUTILB 程序的 SYSIN。检查上述异常并联系支持人员进行分析。	
BA-U3013	错误	INFUTILB 卸载操作失败。查看 INFUTILB 实用程序的 SYSIN 命令语法，验证数据库连接。	
BA-U3014	错误	运行 INFUTILB 程序时，SYSPUNCH 的输出文件类型无效。考虑使用以下类型之一： GDGSupport配置， FileSystemFileConfiguration 或 DummyFile Configuration。	
BA-U3015	错误	运行 INFUTILB 程序时，未找到该数据库类型的格式化程序。分析是否应考虑该类型并联系支持人员进行分析。	
BA-U3016	错误	运行 INFUTILB 程序时缺少 SYSPUNCH 的记录大小。在文件配置中设置记录大小或将其添加到目录中。	

Key	严重性	文本	其他详细信息
BA-U3017	错误	SYSREC 的输出文件类型无效。考虑使用以下类型之一： GDGSupport配置或 FileSystemFileConfiguration。	
BA-U3018	错误	不支持的 SQL 查询。检查 SQL 查询。	
BA-U3019	错误	不支持的列名。检查表的定义。	
BA-U3020	错误	DFSIGDCB 选项：无法更新 SYSREC 文件的 LRECL。考虑使用类型 FileSystemFileConfiguration，或者删除 DFSIGDCB 选项。	
BA-U3021	错误	当前的数据库类型不支持将卸载分成块。使用支持的数据库之一：Postgre SQL 或 Oracle。	
BA-U4001	错误	IDCAMS 解析错误。SYSIN 内容包含无效语法或不支持的命令。查看 SYSIN 内容是否存在语法错误或错别字。查看错误消息，了解有关无法解析的内容的详细信息。	
BA-U5001	错误	调用 DSNUTILB 加载实用程序时位置选项无效。使用有效的位置选项。	

AWS 蓝光时代运行时间 APIs

AWS Blu Age Runtime 使用多个 Web 应用程序来公开 REST 端点，从而提供使用 REST 客户端与现代化应用程序交互的方法（例如，使用调度器调用作业）。

本文档旨在列出可用的 REST 端点，并提供以下方面的详细信息：

- 作用
- 正确使用方法

端点列表的分类具体取决于所提供服务的性质和公开端点的 Web 应用程序。

我们假设您已经具备以下基础知识：使用 POSTMAN、Thunder Client、CURL、Web 浏览器等专用工具使用 REST 端点的基础知识，或者编写一段代码来进行 API 调用。

主题

用户在构建时可用的端点 URLs

本主题列出了终端节点 URLs 的根路径。以下每个 Web 应用程序均定义一个根路径，根路径可由所有端点共用。然后，每个端点会添加自己的专用路径。要使用的 URL 是通过串联这些路径得出的。例如，假设要得到 Gapwalk 应用程序的第一个端点，我们使用：

- `/gapwalk-application` 作为根 Web 应用程序路径。
- `/scripts` 作为专用端点路径。

得出 URL 为 `http://server:port/gapwalk-application/scripts`

server

表示服务器名称（托管给定 Web 应用程序的服务器）。

端口

服务器公开的端口。

AWS Blu Age 中 Gapwalk 应用程序的终端

在本主题中，了解 Gapwalk Web 应用程序的端点。这些端点使用根路径 `/gapwalk-application`。

主题

- 指标端点
- 其他端点
- 作业队列相关端点

Batch 作业（现代 JCLs 化和类似的）相关端点

批处理作业既可以同步运行，也可以异步运行（请参阅以下详细信息）。批处理作业是使用 groovy 脚本执行的，这些脚本是遗留脚本（JCL）现代化的结果。

主题

- 列出已部署的脚本
- 同步启动脚本
- 异步启动脚本
- 列出触发的脚本
- 检索作业执行详细信息
- 列出可以终止的异步启动脚本
- 列出可以终止的同步启动脚本
- 终止给定的作业执行
- 列出用于实现可重启性的现有检查点
- 重新启动作业（同步）
- 重新启动作业（异步）
- 为异步作业执行设置线程限值

列出已部署的脚本

- 支持的方法：GET

  需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/scripts`
- 参数：无
- 此端点以字符串形式返回服务器上已部署的 groovy 脚本列表。此端点主要从 Web 浏览器中使用，因为返回的字符串是一个 HTML 页面，其中包含活动链接（每个可启动脚本一个链接，请参阅以下示例）。

示例响应:

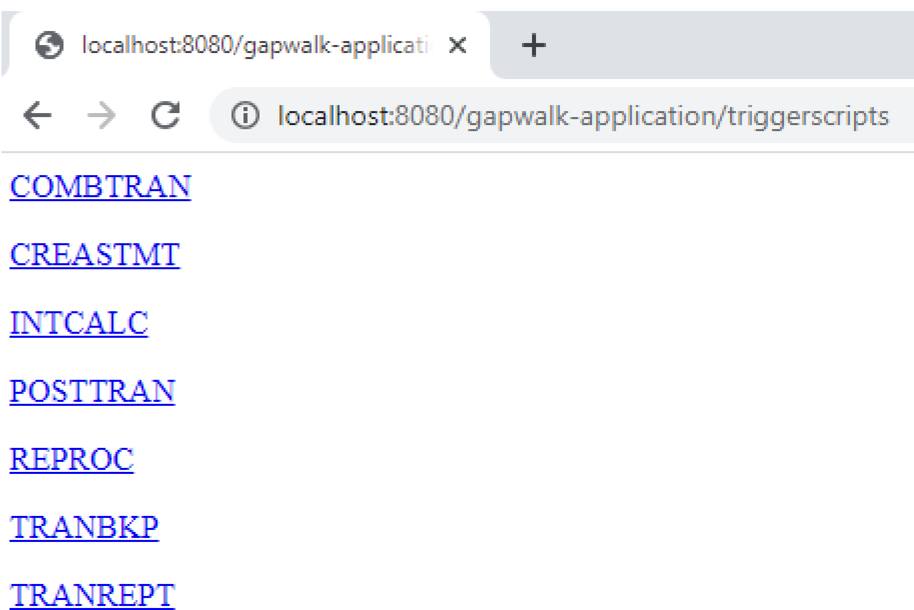
```
<p><a href=./script/COMBTRAN>COMBTRAN</a></p><p><a href=./script/CREASTMT>CREASTMT</
a></p><p><a href=./script/INTCALC>INTCALC</a></p><p><a href=./script/
POSTTRAN>POSTTRAN</a></p><p><a href=./script/REPROC>REPROC</a></p><p><a href=./script/
TRANBKP>TRANBKP</a></p><p><a href=./script/TRANREPT>TRANREPT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

> **Note**
>
> 这些链接是用于同步启动每个列出的脚本的 URL。

- 支持的方法：GET/POST

  需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/triggerscripts`
- 参数：无
- 此端点以字符串形式返回服务器上已部署的 groovy 脚本列表。此端点主要从 Web 浏览器中使用，因为返回的字符串是一个 HTML 页面，其中包含活动链接（每个可启动脚本一个链接，请参阅以下示例）。

  与之前的端点响应相反，这些链接是用于异步启动每个列出的脚本的 URL。



同步启动脚本

此端点有两个变体，分别针对 GET 和 POST 方法使用专用路径（请参阅下文）。

- 支持的方法：GET

  需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/script/{scriptId:.+}`
- 支持的方法：POST

  需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/post/script/{scriptId:.+}`
- 参数：
  - 要启动输入验证的脚本的标识符：脚本 ID 不能为空，不能超过 255 个字符，并且必须与以下模式匹配：`^[a-zA-Z0-9._-]+$`
  - 可选：使用请求参数（显示为 `Map<String,String>`）传递给脚本的参数。给定的参数将自动添加到调用用的 groovy 脚本的绑定中。输入验证：参数映射不能超过 50 个条目。
- 该调用使用可选参数执行脚本（由 ScriptID 标识），并等待完成后再返回带以下任一参数的：*`ResponseEntityString`*
  - HTTP 200："完成。" 或成功执行时的 JSON 成功消息
  - HTTP 200：包含执行失败详细信息的 JSON 错误消息。服务器日志中提供了其他信息。

  > **Note**
  >
  > 运行时现在支持为失败的任务执行返回 HTTP 500 状态码。要配置此响应代码，请参阅主应用程序的可用属性。

  - 输入验证：无效的脚本 ID 或参数将返回包含验证错误详细信息的 HTTP 400 错误请求。

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

通过查看服务器日志，可以发现此处出现了部署问题（预期的程序未正确部署，因此无法被找到，导致作业执行失败）：

```
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**
>
> 同步调用应用于短时间运行的作业。长时间运行的作业应采用异步启动（请参阅以下专用端点）。

异步启动脚本

- 支持的方法：GET/POST

  需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/triggerscript/{scriptId:.+}`
- 参数：
  - 要启动输入验证的脚本的标识符：脚本 ID 不能为空，不能超过 255 个字符，并且必须与以下模式匹配：`^[a-zA-Z0-9._-]+$`
  - 可选：使用请求参数（显示为 `Map<String,String>`）传递给脚本的参数。给定的参数将自动添加到调用的 groovy 脚本的绑定中。输入验证：参数映射不能超过 50 个条目。
- 与上面的同步模式相反，此端点不会等待作业执行完成后再发送响应。如果可以找到可用的线程，作业执行会立即启动，并立即向调用方发送响应。响应中包含作业执行 ID（表示作业执行的唯一标识符），该 ID 可用于查询作业执行状态或强制终止出现故障的作业执行。响应的格式为：

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

- 由于作业异步执行依赖于固定、有限数量的线程，因此如果找不到可用的线程，则可能无法启动作业执行。在这种情况下，返回的消息将如下所示：

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

请参阅下文的 `settriggerthreadlimit` 端点，了解如何提高线程限制。

示例响应:

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

借助唯一作业执行标识符，您可以根据需要快速检索服务器日志中的相关日志条目。其他几个端点也会使用该标识符，详细信息如下文所述。

列出触发的脚本

- 支持的方法：GET/POST

  需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/triggeredscripts/{status:.+}`、`/triggeredscripts/{status:.+}/{namefilter}`
- 参数：
  - Status（必需）：要检索的已触发脚本的状态。输入验证：状态不能为空且不能超过 50 个字符。可能的值为：
    - `all`：显示所有作业执行详细信息，无论作业是否仍在运行。
    - `running`：仅显示当前正在运行的作业的作业详细信息。
    - `done`：仅显示已结束执行的作业的作业详细信息。
    - `killed`：仅显示已使用专用端点强制终止执行的作业的作业详细信息（请参阅下文）。
    - `triggered`：仅显示已触发但尚未启动的作业的作业详细信息。
    - `failed`：仅显示已标记为执行失败的作业的作业详细信息。
    - _namefilter（可选）_：仅检索给定脚本标识符的执行。输入验证：不能超过 255 个字符
- 以 JSON 形式返回作业执行详细信息的集合。有关更多信息，请参阅 作业执行详细信息消息结构。

示例响应:

```
[
```

```
    {
      "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"
    }
  ]
```


检索作业执行详细信息

- 支持的方法：GET

  需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/getjobexecutioninfo/{jobexecutionid:.+}`
- 参数：
  - jobexecutionid（必需）：用于检索相应作业执行详细信息的唯一作业执行标识符。输入验证：Job 执行 ID 不能为空且不能超过 255 个字符
- 返回表示单个作业执行详细信息的 JSON 字符串（请参阅作业执行详细信息消息结构），如果根据给定标识符找不到相应的作业执行详细信息，则响应为空。

列出可以终止的异步启动脚本

- 支持的方法：GET

  需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/killablescripts`
- 返回一组异步启动作业的作业执行标识符，这些作业当前仍在运行并且可以被强制终止（请参阅下文的 `/kill` 端点）。

列出可以终止的同步启动脚本

- 支持的方法：GET

  需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER

- 路径：`/killablesyncscripts`
- 返回一组同步启动作业的作业执行标识符，这些作业当前仍在运行并且可以被强制终止（请参阅下文的 `/kill` 端点）。

终止给定的作业执行

- 支持的方法：POST

  需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN
- 路径：`/kill/{identifier:.+}`
- 参数：作业执行标识符（必需）：要强制终止的作业执行的唯一作业执行标识符。输入验证：标识符不能为空且不能超过 255 个字符
- 返回一条详细描述作业执行终止尝试结果的文本消息；该消息将包含脚本标识符、作业执行唯一标识符以及执行终止发生的日期和时间。如果找不到具有给定标识符且正在运行的作业执行，则将返回一条错误消息。

> **Warning**
>
> - 运行时会尽最大努力合理地终止目标作业执行。因此，来自 /kill 端点的响应可能需要一点时间才能到达调用方，因为 AWS Blu Age 运行时会尽量减少终止任务对业务的影响。
> - 强制终止作业执行应引起注意，因为它可能会产生直接的业务后果，包括可能的数据丢失或损坏。强制终止应该用于给定作业执行出现意外情况且已明确确定数据补救措施的情况。
> - 终止作业后，应进行进一步的调查（事后分析），以找出问题所在，并采取适当的补救措施。
> - 在任何情况下，终止正在运行的作业的尝试都会记录在服务器日志中，并提供警告级别的消息。

列出用于实现可重启性的现有检查点

作业可重启性依赖于脚本能否在 `CheckpointRegistry` 中注册检查点来跟踪作业执行进度。如果作业执行未能正常结束，并且已经注册了重新启动检查点，则只需从最后一个已知的已注册检查点重新启动作业执行即可（无需执行检查点之前的先行步骤）。

- 支持的方法：POST

需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN

- 路径：`/restarts/{scriptId}/{jobId}`
- 参数：
  - scriptId（可选，字符串）：正在重新启动的脚本。
  - jobId（可选，字符串）：作业执行的唯一标识符。
- 返回现有重启检查点的 JSON 格式的列表，这些重启检查点可用于重新启动未正确执行和结束的作业，或通过绕过先前已执行的步骤触发延迟的重启。如果任何脚本都未注册任何检查点，则页面内容会显示“未注册检查点。”

重新启动作业（同步）

- 支持的方法：POST

  需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN
- 路径：`/restart/{hashcode}/{scriptId}/{skipflag}`
- 参数：
  - hashcode（必需，整数）：使用提供的哈希码作为检查点值（请参阅上文的 `/restarts` 端点，以了解如何检索有效的检查点值），重新启动最近的作业执行。
  - scriptId（可选，字符串）：正在重新启动的脚本。
  - skipflag（可选，布尔值）：跳过所选（检查点）步骤的执行并从直接后续步骤（如果有）发起重新启动。
- 返回：请参阅上文的 `/script` 返回内容描述。

重新启动作业（异步）

- 支持的方法：GET

  需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/triggerrestart/{hashcode}/{scriptId}/{skipflag}`
- 参数：
  - hashcode（必需，整数）：使用提供的哈希码作为检查点值（请参阅上文的 `/restarts` 端点，以了解如何检索有效的检查点值），重新启动最近的作业执行。
  - scriptId（可选，字符串）：正在重新启动的脚本。

  - skipflag（可选，布尔值）：跳过所选（检查点）步骤的执行并从直接后续步骤（如果有）发起重新启动。
- 返回：请参阅上文的 `/triggerscript` 返回内容描述。

为异步作业执行设置线程限值

作业异步执行依赖于 JVM 中的专用线程池。该池对可用线程的数量有固定的限值。用户能够根据主机容量（数量 CPUs、可用内存等）调整限制。默认情况下，线程限值设置为 5 个线程。

- 支持的方法：POST

  需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN
- 路径：`/settriggerthreadlimit/{threadlimit:.+}`
- 参数（整数）：要应用的新线程限值。输入验证：必须介于 1 到 1000 之间（含）。
- 返回一条消息 (`String`)，其中包括新的线程限制值和前一个线程限值；如果提供的线程限值无效（不是严格的正整数），则返回错误消息。

示例响应:

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

计算当前正在运行的已触发任务执行个数

- 支持的方法：GET

  需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/countrunningtriggeredscripts`
- 返回一条消息，指示正在运行的异步启动作业数和线程限值（即可以同时运行的最大已触发作业数）。

示例响应:

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

> **Note**
>
> 此端点可用于在启动作业之前检查是否未达到线程限值（达到该值时，将阻止作业启动）。

清除作业执行信息

只要服务器启动，作业执行信息就会保留在服务器内存中。此端点可便于清除内存中较旧的信息，因为这些信息已经不适用。

- 支持的方法：POST

  需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN
- 路径：`/purgejobinformation/{age:.+}`
- 参数：一个严格的正整数值，表示要清除信息的留存时间（以小时为单位）。输入验证：必须介于 0 到 365 之间（含）。
- 返回一条包含以下信息的消息：
  - 清除文件的名称，该文件存储已清除的作业执行信息以供存档。
  - 已清除的作业执行信息的数量。
  - 内存中剩余的作业执行信息数量。

指标端点

输入验证：所有指标端点都会验证请求参数，如果值无效，则返回 HTTP 400 错误请求。

JVM

此端点返回与 JVM 相关的可用指标。

- 支持的方法：GET

  需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/metrics/jvm`
- 参数：无
- 返回一条包含以下信息的消息：
  - threadActiveCount：活动线程数。
  - jvmMemoryUsed：Java 虚拟机正在使用的内存。

  - jvmMemoryMax：Java 虚拟机允许的最大内存。
  - jvmMemoryFree：Java 虚拟机当前未使用的可用内存。

会话

此端点返回与当前打开的 HTTP 会话相关的指标。

- 支持的方法：GET

  需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/metrics/session`
- 参数：无
- 返回一条包含以下信息的消息：
  - sessionCount：服务器当前维护的活跃用户会话数。

批处理

- 支持的方法：GET

  需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/metrics/batch`
- 参数：
  - startTimestamp（可选，数字）：用于数据筛选的起始时间戳。输入验证：必须是有效的数值。
  - endTimestamp（可选，数字）：用于数据筛选的结束时间戳。输入验证：必须是有效的数值。
  - page（可选，数字）：用于分页的页码。输入验证：必须为正整数。
  - pageSize（可选，数字）：分页中每页的项目数。输入验证：必须是严格的正整数，最大值为 500。
  - 输入验证：参数映射不能超过 20 个条目
- 返回一条包含以下信息的消息：
  - content：批量执行指标的列表。
  - pageNumber：分页中的当前页码。
  - pageSize：每页显示的项目数。
  - totalPages：可用的总页数。
  - numberOfElements：当前页面上的项目数。

  - last：最后一页的布尔标志。
  - first：第一页的布尔标志。

事务

- 支持的方法：GET

  需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/metrics/transaction`
- 参数：
  - startTimestamp（可选，数字）：用于数据筛选的起始时间戳。输入验证：必须是有效的数值。
  - endTimestamp（可选，数字）：用于数据筛选的结束时间戳。输入验证：必须是有效的数值。
  - page（可选，数字）：用于分页的页码。输入验证：必须为正整数。
  - pageSize（可选，数字）：分页中每页的项目数。输入验证：必须是严格的正整数，最大值为 500。
  - 输入验证：参数映射不能超过 20 个条目
- 返回一条包含以下信息的消息：
  - content：批量执行指标的列表。
  - pageNumber：分页中的当前页码。
  - pageSize：每页显示的项目数。
  - totalPages：可用的总页数。
  - numberOfElements：当前页面上的项目数。
  - last：最后一页的布尔标志。
  - first：第一页的布尔标志。

其他端点

使用以下端点列出已注册的程序或服务、了解运行状况和管理 JICS 事务。

主题

- [列出已注册程序]
- [列出已注册服务]
- [运行状况]

- 列出可用的 JICS 事务
- 启动 JICS 事务
- 启动 JICS 事务（备用）
- 列出活动会话

列出已注册程序

- 支持的方法：GET

  需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/programs`
- 以 html 页面的形式返回已注册程序的列表。每个程序都由其主程序标识符指定。返回的列表中包含经过现代化改造的遗留程序和实用程序（IDCAMS、IEBGENER等...）。请注意，可用的实用程序将取决于您的 tomcat 服务器上部署的实用程序 Web 应用程序。例如， z/OS 公用事业支持计划可能不适用于现代化的 iSeries 资产，因为它们无关紧要。

列出已注册服务

- 支持的方法：GET

  需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/services`
- 以 html 页面的形式返回已注册运行时服务的列表。给定的服务由 AWS Blu Age 运行时作为实用程序提供，例如，可以在 groovy 脚本中使用。Blusam 加载服务（从遗留数据集创建 Blusam 数据集）属于此类服务。

示例响应:

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

运行状况

- 支持的方法：GET

  需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/`

- 返回一条简单消息，表明 gapwalk-application 已启动并正在运行 (`Jics application is running.`)

列出可用的 JICS 事务

- 支持的方法：GET

  需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/transactions`
- 返回一个 html 页面，其中列出了所有可用的 JICS 事务。这仅适用于具有 JICS 元素（对遗留 CICS 元素进行现代化）的环境。

示例响应:

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

启动 JICS 事务

- 支持的方法：POST

  需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/jicstransrunner/{jtrans:.+}`
- 参数：
  - JICS 事务标识符（字符串，必需）：要启动的 JICS 事务的标识符（最多 8 个字符）
  - 必需：以 Map<String,Object> 形式传递给事务的其他输入数据。该映射的内容用于提供由 JICS 事务使用的 COMMAREA。如果运行事务不需要任何数据，则该映射可以为空。
  - 可选：Http 标头条目，用于自定义给定事务的运行环境。支持以下标头键：
    - `jics-channel`：将在本事务启动时启动的程序要使用的 JICS CHANNEL 的名称。
    - `jics-container`：用于启动此 JICS 事务的 JICS 容器的名称。
    - `jics-startcode`：在 JICS 事务启动时使用的 STARTCODE（字符串，最多 2 个字符）。有关可能的值，请参阅 STARTCODE（向下浏览页面）。
    - `jicxa-xid`：由调用方发起的“全局事务”（XA）的 XID（X/Open 事务标识符 XID 结构），当前的 JICS 事务启动将参与该全局事务。输入验证：XID 不能为空且不能超过 255 个字符。

- 返回表示 JICS 事务启动结果的 `com.netfective.bluage.gapwalk.rt.shared.web.TransactionResultBean` JSON 序列化。
- 输入验证：无效的 XID 值（空白或超过 255 个字符）将返回包含验证错误详细信息的 HTTP 400 错误请求。

有关该结构的更多详细信息，请参阅事务启动结果结构。

启动 JICS 事务（备用）

- 支持的方法:POST

  需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/jicstransaction/{jtrans:.+}`
- 参数：

  JICS 事务标识符（字符串，必需）

    要启动的 JICS 事务的标识符（长度为最多 8 个字符）

  必需：以 Map<String,Object> 形式传递给事务的其他输入数据

    该映射的内容用于提供由 JICS 事务使用的 COMMAREA。如果运行事务不需要任何数据，则该映射可以为空。

  可选：Http 标头条目，用于自定义给定事务的运行环境。

    支持以下标头键：

    - `jics-channel`：将在本事务启动时启动的程序要使用的 JICS CHANNEL 的名称。
    - `jics-container`：用于启动此 JICS 事务的 JICS 容器的名称。
    - `jics-startcode`：在 JICS 事务启动时使用的 STARTCODE（字符串，最多 2 个字符）。有关可能的值，请参阅 STARTCODE（向下浏览页面）。
    - `jicxa-xid`：由调用方发起的"全局事务"(XA) 的 XID（X/Open 事务标识符 XID 结构），当前的 JICS 事务启动将参与该全局事务。输入验证：XID 不能为空且不能超过 255 个字符。
- 返回表示 JICS 事务启动结果的 `com.netfective.bluage.gapwalk.rt.shared.web.RecordHolderBean` JSON 序列化。有关该结构的详细信息，请参阅事务启动记录结果结构。
- 输入验证：无效的 XID 值（空白或超过 255 个字符）将返回包含验证错误详细信息的 HTTP 400 错误请求。

列出活动会话

- 支持的方法：GET

  需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN
- 路径：`/activesessionlist`
- 参数：无
- 输入验证：参数映射不能超过 20 个条目
- 返回 JSON 序列化形式的 `com.netfective.bluage.gapwalk.application.web.sessiontracker.SessionTrackerObje` 列表，表示活动用户会话的列表。如果禁用了会话跟踪，将会返回空列表。

作业队列相关端点

任务队列是 AWS Blu Age 对 AS4 00 个作业提交机制的支持。在 AS4 00 中，Job 队列用于在特定的线程池上运行作业。作业队列由名称和最大线程数定义，最大线程数对应于该队列上可以同时运行的最大程序数。如果队列中提交的作业大于最大线程数，则作业将等待线程可用。

有关队列中作业状态的详尽列表，请参阅队列中作业的可能状态。

作业队列上的操作通过以下专用端点处理。可以使用以下根 URL 从 Gapwalk 应用程序 URL 调用这些操作：`http://server:port/gapwalk-application/jobqueue`。

主题

列出可用的队列

- 支持的方法：GET

  需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`list-queues`
- 以 JSON 键值列表的形式返回可用队列列表及其状态。

示例响应:

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

作业队列的可能状态为：

STAND_BY

作业队列正在等待启动。

STARTED

作业队列已启动并正在运行。

UNKNOWN

无法确定作业队列状态。

启动或重新启动作业队列

- 支持的方法：POST

  需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN
- 路径：`/restart/{name}`
- 参数：要启动/重新启动的队列的名称（必需，字符串）。输入验证：队列名称不得为空且不能超过 255 个字符。
- 端点不返回任何内容，而是依赖 http 状态来指示 start/restart 操作的结果：

  HTTP 200

  start/restart 操作进展顺利：给定的任务队列现已启动。

HTTP 404

作业队列不存在。

HTTP 503

start/restart 尝试期间出现异常（应检查服务器日志以找出问题所在）。

提交要启动的作业

- 支持的方法：POST

  需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/submit`
- 参数：（必需，请求体）`com.netfective.bluage.gapwalk.rt.jobqueue.SubmitJobMessage` 对象的 JSON 序列化。有关更多信息，请参阅 提交作业和计划作业输入。
- 返回包含原始 `SubmitJobMessage` 的 JSON 和指示作业是否已提交的日志。

列出所有已提交的作业

- 支持的方法：GET

  需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/list-jobs?status={status}&size={size}&page={page}&sort={sort}`
- 参数：
  - page：要检索的页码（默认值 = 1）
  - size：页面的大小（默认值 = 50，最大值 = 300）
  - sort：作业的顺序。（默认值 =“executionId”）。“executionId”是目前支持的唯一值
  - status：（可选）如果提供，它将根据状态执行筛选。
- 以 JSON 字符串的形式返回所有计划作业的列表。有关响应示例，请参阅计划作业响应列表。

释放所有“暂停”作业

- 支持的方法：POST

  需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER

- 路径：`/release-all`
- 返回一条消息，指示释放尝试操作的结果。这里有两种可能的情况：
  - HTTP 200 和消息“所有作业均已成功释放！”（如果所有作业均已成功释放）。
  - HTTP 503 和消息“作业未释放。出现未知错误。有关更多详细信息，请参阅日志”（如果释放尝试出现问题）。

释放给定作业名称下所有处于“暂停”状态的作业

对于给定的作业名称，可以提交多个具有不同作业编号的作业（作业运行的唯一性由一组 <job name, job number> 来保证）。此端点将尝试释放具有给定作业名称的所有处于“暂停”状态的作业提交。

- 支持的方法：POST

  需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/release/{name}`
- 参数：要查找的作业名称，为字符串。必需。
- 返回一条消息，指示释放尝试操作的结果。这里有两种可能的情况：
  - HTTP 200 和消息 “组 <name> (<number of released jobs>) 中的作业已成功释放！”（如果作业已成功释放）。
  - HTTP 503 和消息“组 <name> 中的作业未释放。出现未知错误。有关更多详细信息，请参阅日志”（如果释放尝试出现问题）。

释放具有给定作业编号的作业

此端点将尝试释放具有给定 <job name, job number> 且处于“暂停”状态的唯一作业提交。

- 支持的方法：POST

  需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/release/{name}/{number}`
- 参数：

  name

    参数：要查找的作业名称，为字符串。必需。

数字

要查找的作业编号，以整数表示。必需。

返回

一条消息，指示释放尝试操作的结果。这里有两种可能的情况：

- HTTP 200 和消息“作业 <name/number> 已成功释放！”（如果该作业已成功释放）。
- HTTP 503 和消息“作业 <name/number>> 未释放。出现未知错误。有关更多详细信息，请参阅日志”（如果释放尝试出现问题）。

按重复计划提交作业

安排将按重复计划执行的作业。

- 支持的方法：POST

  需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/schedule`
- 参数：请求正文必须包含 `com.netfective.bluage.gapwalk.rt.jobqueue.SubmitJobMessage` 对象的 JSON 序列化。

列出所有已提交的重复作业

- 支持的方法：GET

  需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN
- 路径：`/schedule/list?status={status}&size={size}&page={page}&sort={sort}`
- 参数：

  1. page：要检索的页码（默认值 = 1）
  2. size：页面的大小（默认值 = 50，最大值 = 300）
  3. sort：作业的顺序。（默认值 =“id”）。“id”是目前支持的唯一值。
  4. status：（可选）如果提供，它将根据状态执行筛选。可能的值是第 1 部分中提到的值。
  5. status：（可选）如果提供，它将根据状态执行筛选。可能的值是第 1 部分中提到的值。
  6. 以 JSON 字符串的形式返回所有计划作业的列表。

取消重复作业的计划

删除按重复计划创建的作业。作业计划状态设置为“INACTIVE”。

- 支持的方法：POST

  需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN
- 路径：`/schedule/remove/{schedule_id}`
- 参数：`schedule_id`，要删除的计划作业的标识符。

Blusam应用程序控制台 REST 端点

在本节中，您可以了解Blusam应用程序控制台，这是一款旨在简化现代化 VSAM 数据集管理的 API。BlusamWeb 应用程序的端点使用根路径`/bac`。

主题

- 数据集相关端点
- 批量处理数据集相关端点
- 记录
- 掩码
- 其他
- BAC 用户管理端点

数据集相关端点

使用以下端点创建或管理特定的数据集。

主题

- 创建数据集
- 上传文件
- 加载数据集（POST）
- 加载数据集（GET）
- 从 Amazon S3 存储桶加载数据集
- 将数据集导出到 Amazon S3 存储桶
- 清除数据集

- 删除数据集
- 统计数据集记录数

创建数据集

可使用此端点创建数据集定义。

- 支持的方法：POST
- 需要身份验证和 ROLE_ADMIN 角色。
- 路径：`/api/services/rest/bluesamservice/createDataSet`
- 参数：

  名称

    （必需，字符串）：数据集的名称。

  type

    （必需，字符串）：数据集类型。可能的值为：`ESDS`、`KSDS`、`RRDS`。

  recordSize

    （可选，字符串）：数据集每条记录的最大大小。

  fixedLength

    （可选，布尔值）：表示记录长度是否固定。

  compression

    （可选，布尔值）：表示数据集是否压缩。

  cacheEnable

    （可选，布尔值）：表示是否为数据集启用缓存。

  alternativeKeys

    （可选，键列表）：

    - offset（必需，数字）
    - length（必需，数字）
    - name（必需，数字）
- 返回一个代表新创建的数据集的 JSON 文件。

示例请求：

```
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
}
```

示例响应:

```
{
    "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
  }
```


上传文件

可使用此端点将文件上传到服务器。文件存储在与每个特定用户相对应的临时文件夹中。每次需要上传文件时都使用此端点。

- 支持的方法：POST
- 需要身份验证和 ROLE_ADMIN 角色。
- 路径：`/api/services/rest/bluesamservice/upload`
- 参数：

  file

    （必需，多部分/形式数据）：要上传的文件。
- 返回表示上传状态的布尔值

加载数据集（POST）

使用 `createDataSet` 创建数据集定义后，可以将与上传的文件关联的记录加载到特定的数据集中。

- 支持的方法：POST
- 需要身份验证和 ROLE_ADMIN 角色。
- 路径：`/api/services/rest/bluesamservice/loadDataSet`
- 参数：

  名称

    （必需，字符串）：数据集的名称。
- 返回请求的状态和已加载的数据集。

加载数据集（GET）

- 支持的方法：GET
- 需要身份验证和 ROLE_ADMIN 角色。
- 路径：`/api/services/rest/bluesamservice/loadDataSet`
- 参数：

  名称

    （必需，字符串）：数据集的名称。

  dataset file

    （必需，字符串）：数据集文件名称。
- 返回请求的状态和已加载的数据集。

从 Amazon S3 存储桶加载数据集

使用 Amazon S3 存储桶中的 listcat 文件加载数据集。

- 支持的方法：GET
- 需要身份验证和 ROLE_ADMIN 角色。
- 路径：`/api/services/rest/bluesamservice/loadDataSetFromS3`
- 参数：

listcatFileS3Location

（必需，字符串）：listcat 文件在 Amazon S3 中的位置。

datasetFileS3Location

（必需，字符串）：数据集文件在 Amazon S3 中的位置。

区域

（必填，字符串）：存储文件的 Amazon S3 AWS 区域 。

- 返回新创建的数据集

示例请求：

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

将数据集导出到 Amazon S3 存储桶

将数据集导出到指定的 Amazon S3 存储桶。

- 支持的方法：GET
- 需要身份验证和 ROLE_ADMIN 角色。
- 路径：`/api/services/rest/bluesamservice/exportDataSetToS3`
- 参数：

  s3Location

  （必需，字符串）：将数据集文件导出到 Amazon S3 中的位置。

  datasetName

  （必需，字符串）：要导出的数据集的名称。

  区域

  （必填，字符串）：Amazon S3 存储桶的。 AWS 区域

  kmsKeyId

  （可选，字符串）：用于加密导出到 Amazon S3 存储桶的数据集的 AWS KMS ID。

- 返回导出的数据集

示例请求：

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

清除数据集

清除数据集中的所有记录。

- 支持的方法：GET、POST
- 需要身份验证和 ROLE_ADMIN 角色。
- 路径：`/api/services/rest/bluesamservice/clearDataSet`
- 参数：

  名称

    （必需，字符串）：要清除的数据集的名称。
- 返回请求的状态。

删除数据集

删除数据集定义和记录。

- 支持的方法：POST
- 需要身份验证和 ROLE_ADMIN 角色。
- 路径：`/api/services/rest/bluesamservice/deleteDataSet`
- 参数：

  名称

    （必需，字符串）：要删除的数据集的名称。
- 返回请求的状态和已删除的数据集。

统计数据集记录数

此端点会返回与数据集关联的记录数。

- 支持的方法：POST
- 需要身份验证和 ROLE_USER 角色。
- 路径：`/api/services/rest/bluesamservice/countRecords`
- 参数：

  名称

    （必需，字符串）：数据集的名称。
- 返回：记录数

批量处理数据集相关端点

使用以下端点同时创建或管理多个数据集。

主题

- [导出数据集（GET）]()
- [导出数据集（POST）]()
- [创建多个数据集]()
- [列出所有数据集]()
- [直接列出所有数据集]()
- [按页直接列出所有数据集]()
- [流式传输数据集]()
- [删除所有数据集]()
- [从 listcat 文件中获取数据集定义]()
- [从上传的 listcat 文件中获取数据集定义]()
- [获取数据集]()
- [从 JSON 文件加载 listcat]()

导出数据集（GET）

- 支持的方法：GET
- 需要身份验证和 ROLE_USER 角色。
- 路径：`/api/services/rest/bluesamservice/exportDataSet`

- 参数：

  datasetName

  （必需，字符串）：要导出的数据集的名称。

  datasetOutputFile

  （必需，字符串）：服务器上要用于存储导出的数据集的文件夹路径。

  rdw

  （必需，布尔值）：是否要在导出的记录中包含记录描述字（RDW）。如果数据集中的记录具有固定的长度，则会忽略此参数的值。

- 返回请求的状态和包含导出数据集（如有）的文件的路径。如果响应中的数据集为空，则表示系统无法找到具有给定名称的数据集。

导出数据集（POST）

- 支持的方法：POST
- 需要身份验证和 ROLE_USER 角色。
- 路径：`/api/services/rest/bluesamservice/exportDataSet`
- 参数：

  dumpParameters

  （必填， BACRead参数）：Bluesam 读取参数。

- 返回导出的数据集的状态。

创建多个数据集

- 支持的方法：POST
- 需要身份验证和 ROLE_ADMIN 角色。
- 路径：`/api/services/rest/bluesamservice/createAllDataSets`
- 参数：
  - 数据集列表

    名称

    （必需，字符串）：数据集的名称。

type

（必需，字符串）：数据集类型。可能的值为：`ESDS`、`KSDS`、`RRDS`。

recordSize

（可选，字符串）：数据集每条记录的最大大小。

fixedLength

（可选，布尔值）：表示记录长度是否固定。

compression

（可选，布尔值）：表示数据集是否压缩。

cacheEnable

（可选，布尔值）：表示是否为数据集启用缓存。

- 返回：请求的状态和新创建的数据集。

列出所有数据集

- 支持的方法：GET
- 需要身份验证和 ROLE_USER 角色。
- 路径：`/api/services/rest/bluesamservice/listDataSet`
- 参数：无
- 返回：请求的状态和数据集列表。

直接列出所有数据集

- 支持的方法：GET
- 需要身份验证和 ROLE_USER 角色。
- 路径：`/api/services/rest/bluesamservice/directListDataSet`
- 参数：无
- 返回：请求的状态和数据集列表。

按页直接列出所有数据集

- 支持的方法：GET
- 需要身份验证和 ROLE_USER 角色。
- 路径：`/api/services/rest/bluesamservice/directListDataSetByPage`
- 参数：

  datasetName

  （必需，字符串）：数据集的名称。

  pageNumber

  （必需，整数）：页码。

  pageSize

  （必需，整数）：页面大小。
- 返回：请求的状态和数据集列表。

流式传输数据集

- 支持的方法：GET
- 需要身份验证和 ROLE_ADMIN 角色。
- 路径：`/api/services/rest/bluesamservice/streamDataset`
- 参数：

  datasetName

  （必需，字符串）：数据集的名称。
- 返回：所请求数据集的流。

删除所有数据集

- 支持的方法：POST
- 需要身份验证和 ROLE_ADMIN 角色。
- 路径：`/api/services/rest/bluesamservice/removeAll`
- 参数：无
- 返回：代表请求状态的布尔值。

从 listcat 文件中获取数据集定义

- 支持的方法：POST
- 需要身份验证和 ROLE_ADMIN 角色。
- 路径：`/api/services/rest/bluesamservice/getDataSetsDefinitionFromListcat`
- 参数：

  paramFilePath

  （必需，字符串）：listcat 文件的路径。
- 返回：数据集列表

从上传的 listcat 文件中获取数据集定义

- 支持的方法：POST
- 需要身份验证和 ROLE_ADMIN 角色。
- 路径：`/api/services/rest/bluesamservice/getDataSetsDefinitionFromUploadedListcat`
- 参数：无
- 返回：数据集列表

获取数据集

- 支持的方法：GET
- 需要身份验证和 ROLE_USER 角色。
- 路径：`/api/services/rest/bluesamservice/getDataSet`
- 参数：

  名称

  （必需，字符串）：数据集的名称。
- 返回请求的数据集。

从 JSON 文件加载 listcat

- 支持的方法：GET

- 需要身份验证和 ROLE_ADMIN 角色。
- 路径：`/api/services/rest/bluesamservice/loadListcatFromJsonFile`
- 参数：

  filePath

    （必需，字符串）：listcat 文件的路径。
- 返回：数据集列表

记录

使用以下端点在数据集中创建或管理记录。

主题

- 创建记录
- 读取数据集
- 删除记录
- 更新记录
- 保存记录
- 验证记录
- 获取记录树

创建记录

可使用此端点创建新记录。

- 支持的方法：POST
- 需要身份验证和 ROLE_USER 角色。
- 路径：`/api/services/rest/crud/createRecord`
- 参数：

  数据集

    （必填， DataSet）：数据集对象

  mask

    （必需，掩码）：掩码对象。

- 返回：请求的状态和创建的记录。

读取数据集

可使用此端点读取数据集。

- 支持的方法：POST
- 需要身份验证和 ROLE_USER 角色。
- 路径：`/api/services/rest/crud/readDataSet`
- 参数：

  数据集

    （必填， DataSet）：数据集对象。
- 返回请求的状态和带有记录的数据集。

删除记录

可使用此端点从数据集中删除记录。

- 支持的方法：POST
- 需要身份验证和 ROLE_USER 角色。
- 路径：`/api/services/rest/crud/deleteRecord`
- 参数：

  数据集

    （必填， DataSet）：数据集对象

  记录

    （必需，记录）：要删除的记录
- 返回删除状态。

更新记录

可使用此端点更新与数据集关联的记录。

- 支持的方法：POST

- 需要身份验证和 ROLE_USER 角色。
- 路径：`/api/services/rest/crud/updateRecord`
- 参数：

  数据集

    （必填， DataSet）：数据集对象

  记录

    （必需，记录）：要更新的记录
- 返回请求的状态和带有记录的数据集。

保存记录

可使用此端点将记录保存到数据集中并使用掩码。

- 支持的方法：POST
- 需要身份验证和 ROLE_USER 角色。
- 路径：`/api/services/rest/crud/saveRecord`
- 参数：

  数据集

    （必填， DataSet）：数据集对象

  记录

    （必需，记录）：要保存的记录
- 返回请求的状态和带有记录的数据集。

验证记录

可使用此端点验证记录。

- 支持的方法：POST
- 需要身份验证和 ROLE_USER 角色。
- 路径：`/api/services/rest/crud/validateRecord`
- 参数：

数据集

（必填， DataSet）：数据集对象

- 返回请求的状态和带有记录的数据集。

获取记录树

可使用此端点获取记录的分层树。

- 支持的方法：POST
- 需要身份验证和 ROLE_USER 角色。
- 路径：`/api/services/rest/crud/getRecordTree`
- 参数：

  数据集

  （必填， DataSet）：数据集对象

  记录

  （必需，记录）：要获取的记录
- 返回请求的状态和所请求记录的分层树。

掩码

使用以下端点对数据集加载或应用掩码。

主题

- 加载掩码
- 应用掩码
- 应用掩码过滤器

加载掩码

可使用此端点检索与特定数据集关联的所有掩码。

- 支持的方法：POST
- 需要身份验证和 ROLE_USER 角色。

- 路径：`/api/services/rest/crud/loadMasks`
- 路径变量：

  recordSize: .../loadMasks/{recordSize}

  （可选，数字）：记录大小，筛选条件加载的与此记录大小相匹配的掩码
- 参数：

  数据集

  （必填， DataSet）：数据集对象
- 返回请求的状态和掩码列表。

应用掩码

可使用此端点对特定数据集应用掩码。

- 支持的方法：POST
- 需要身份验证和 ROLE_USER 角色。
- 路径：`/api/services/rest/crud/applyMask`
- 参数：

  数据集

  （必填， DataSet）：数据集对象

  mask

  （必需，掩码）：数据集对象
- 返回请求的状态和已应用掩码的数据集。

应用掩码过滤器

可使用此端点对特定数据集应用掩码和筛选条件。

- 支持的方法：POST
- 需要身份验证和 ROLE_USER 角色。
- 路径：`/api/services/rest/crud/applyMaskFilter`
- 参数：

数据集

（必填， DataSet）：数据集对象

mask

（必需，掩码）：数据集对象

- 返回请求的状态以及已应用掩码和过滤器的数据集。

其他

使用以下端点管理数据集的缓存或检查数据集的特征

主题

- 检查预热缓存
- 检查是否已启用缓存
- 启用缓存
- 检查已分配的 RAM 缓存
- 检查持久性
- 检查支持的数据集类型
- 检查服务器运行状况

检查预热缓存

检查是否已为特定数据集启用预热缓存。

- 支持的方法：POST
- 需要身份验证和 ROLE_ADMIN 角色。
- 路径：`/api/services/rest/bluesamservice/warmupCache`
- 参数：

  名称

  （必需，字符串）：数据集的名称。
- 返回：如果已启用预热缓存，则返回 true，否则返回 false。

检查是否已启用缓存

检查是否已为特定数据集启用缓存。

- 支持的方法：GET
- 需要身份验证和 ROLE_USER 角色。
- 路径：`/api/services/rest/bluesamservice/isEnableCache`
- 参数：无
- 如果已启用缓存，则返回 true。

启用缓存

- 支持的方法：POST
- 需要身份验证以及 ROLE_ADMIN 和 ROLE_SUPER_ADMIN 角色。
- 路径：`/api/services/rest/bluesamservice/enableDisableCache/{enable}`
- 参数：

  enable

    （必需，布尔值）：如果设置为 true，将启用缓存。
- 不返回任何内容

检查已分配的 RAM 缓存

可使用此端点检索已分配的 RAM 缓存。

- 支持的方法：GET
- 需要身份验证和 ROLE_USER 角色。
- 路径：`/api/services/rest/bluesamservice/allocatedRamCache`
- 参数：无
- 返回：表示内存大小的字符串

检查持久性

- 支持的方法：GET

- 需要身份验证和 ROLE_USER 角色。
- 路径：`/api/services/rest/bluesamservice/persistence`
- 参数：无
- 返回：表示使用的持久性，以字符串形式表示。

检查支持的数据集类型

- 支持的方法：GET
- 路径：`/api/services/rest/bluesamservice/getDataSetTypes`
- 需要身份验证和 ROLE_USER 角色。
- 参数：无
- 返回：支持的数据集类型列表，以字符串列表形式显示。

检查服务器运行状况

- 支持的方法：GET
- 路径：`/api/services/rest/bluesamserver/serverIsUp`
- 参数：无
- 返回：无。HTTP 响应状态代码 200 表示服务器已启动并正在运行。

BAC 用户管理端点

使用以下端点管理用户互动。

主题

- 登录用户
- 验证系统中是否至少存在一个用户
- 记录新用户
- 获取用户信息
- 列出用户
- 删除用户
- 注销当前用户

登录用户

- 支持的方法：POST
- 路径：`/api/services/security/servicelogin/login`
- 参数：无
- 返回 `com.netfective.bluage.bac.entities.SignOn` 对象的 JSON 序列化，表示在当前请求中提供凭证的用户。在返回的对象中，密码隐藏不可见，并且会列出赋予用户的角色。

示例响应:

```
{
     "login": "some-admin",
     "password": null,
     "roles": [
       {
         "id": 0,
         "roleName": "ROLE_ADMIN"
       }
     ]
   }
```

验证系统中是否至少存在一个用户

- 支持的方法：GET
- 路径：`/api/services/security/servicelogin/hasAccount`
- 参数：无
- 如果除默认超级管理员用户之外，还创建了至少一个用户，则会返回布尔值 `true`。否则返回 `false`。

记录新用户

- 支持的方法：POST
- 需要身份验证和 ROLE_ADMIN 角色。
- 路径：`/api/services/security/servicelogin/recorduser`
- 参数：`com.netfective.bluage.bac.entities.SignOn` 对象的 JSON 序列化，表示要添加到存储中的用户。必须定义该用户的角色，否则用户可能无法使用 BAC 设施和端点。

- 如果成功创建用户，则返回布尔值 `true`。否则返回 `false`。
- 示例请求 JSON：

```
{
    "login": "simpleuser",
    "password": "simplepassword",
    "roles": [
      {
        "id": 2,
        "roleName": "ROLE_USER"
      }
    ]
  }
```

`roleName` 的两个有效值如下：

- `ROLE_ADMIN`: 可以管理Blusam资源和用户。
- `ROLE_USER`: 可以管理Blusam资源但不能管理用户。

获取用户信息

- 支持的方法：GET
- 路径：`/api/services/security/servicelogin/userInfo`
- 参数：无
- 返回当前连接的用户的用户名和角色

列出用户

- 支持的方法：GET
- 需要身份验证和 ROLE_ADMIN 角色。
- 路径：`/api/services/security/servicelogin/listusers`
- 参数：无
- 返回 `com.netfective.bluage.bac.entities.SignOn` 列表，采用 JSON 序列化。

删除用户

> ⚠ Important
>
> 并且无法撤消。已删除的用户将无法再次连接到 BAC 应用程序。

- 支持的方法：POST
- 需要身份验证和 ROLE_ADMIN 角色。
- 路径：`/api/services/security/servicelogin/deleteuser`
- 参数：`com.netfective.bluage.bac.entities.SignOn` 对象的 JSON 序列化，表示要从存储中删除的用户。
- 如果成功删除用户，则返回布尔值 `true`。

注销当前用户

- 支持的方法：GET
- 路径：`/api/services/security/servicelogout/logout`
- 参数：无
- 如果成功注销当前用户，则返回 JSON 消息 `{"success":true}`。相关的 HTTP 会话将失效。

在 AWS Blu Age 中管理 JICS 应用程序控制台

JICS 组件是 AWS Blu Age 对传统 CICS 资源现代化的支持。JICS 应用程序控制台 Web 应用程序专门用于管理 JICS 资源。借助以下端点，无需与 JAC 用户界面交互即可执行管理任务。端点需要身份验证时，请求须包含身份验证详细信息（根据基本身份验证要求，通常为用户名/密码）。JICS 应用程序控制台 Web 应用程序的端点使用根路径 `/jac/`。

主题

JICS 资源管理

以下所有端点都与 JICS 资源管理有关，可供 JICS 管理员用于每天处理资源。

主题

- 更新 TRANSACTION
- 更新 PROGRAM
- 更新 FILE
- 更新 TDQUEUE
- 更新 TSMODEL
- 更新元素
- 更新插入元素
- 检索元素
- JICS CRUD 操作

列出 JICS LIST 和 GROUP

LIST 和 GROUP 是 JICS 组件中拥有的主要容器资源。所有 JICS 资源都必须属于一个 GROUP。GROUP 可以属于 LIST，但这不是必需的。给定的 JICS 环境中甚至可能不存在 LIST，但大多数时候，LIST 可为资源提供额外的组织层。有关 CICS 资源组织的更多信息，请参阅 CICS 资源。

- 支持的方法：GET
- 需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/api/services/rest/jicsservice/listJicsListsAndGroups`
- 参数：无
- 返回：以 JSON 格式返回序列化 JicsContainer 对象的列表，包括列表和组。

示例响应:

```
[
    {
      "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
  }
]
```


检索 JICS 资源

- 支持的方法：POST
- 需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/api/services/rest/jicsservice/retrieveJicsResources`
- 参数：表示要检索的 JICS 资源的 JSON 有效载荷。这是 `com.netfective.bluage.jac.entities.request.RetrieveOperationRequest` 对象的 JSON 序列化。
- 返回：序列化 JicsResource 对象的列表。这些对象不按特定顺序返回，并且具有不同的类型（PROGRAM、TRANSACTION、FILE 等）。

列出 JICS GROUP

- 支持的方法：GET
- 需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/api/services/rest/jicsservice/listJicsGroups`
- 参数：无
- 以 JSON 形式返回序列化 JicsContainer 对象（组）列表。这些 GROUP 返回时不带有组自身的 LIST 信息。

示例响应:

```
[
    {
      "jacType": "JACGroup",
      "name": "MURACHS",
      "isActive": true,
      "children": []
    },
    {
      "jacType": "JACGroup",
      "name": "TEST",
      "isActive": true,
      "children": []
    }
  ]
```

列出给定 LIST 的 JICS 组

- 支持的方法：POST
- 需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/api/services/rest/jicsservice/listGroupsForList`
- 参数：JSON 有效载荷，表示要查找的 GROUP 所属的 JICS LIST。这是 `com.netfective.bluage.jac.entities.JACList` 对象的 JSON 序列化。

  示例请求：

```
{
    "jacType":"JACList",
    "name":"MURACHS",
    "isActive":true
  }
```

- 以 JSON 形式返回附加到给定列表的序列化 JicsContainer 对象 (GROUPS) 列表。这些 GROUP 返回时不带有组自身的 LIST 信息。

  示例响应:

```
[
    {
      "jacType": "JACGroup",
      "name": "MURACHS",
      "isActive": true,
```

```
        "children": []
      }
    ]
```

列出给定 GROUP 的 JICS 资源

- 支持的方法：POST
- 需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/api/services/rest/jicsservice/listResourcesForGroup`
- 参数：JSON 有效载荷，表示要查找的资源所属的 JICS GROUP。这是 `com.netfective.bluage.jac.entities.JACGroup` 对象的 JSON 序列化。您无需指定 GROUP 的所有字段，但名称是必需的。

  示例请求：

```
{
      "jacType":"JACGroup",
      "name":"MURACHS",
      "isActive":true
    }
```

- 返回给定 GROUP 拥有的序列化 JicsResource 对象的列表。这些对象不按特定顺序返回，并且具有不同的类型（PROGRAM、TRANSACTION、FILE 等）。

列出给定 GROUP（也可以使用名称）的 JICS 资源

- 支持的方法：POST
- 需要身份验证
- 路径：`/api/services/rest/jicsservice/listResourcesForGroupName`
- 参数：要查找的资源所属 GROUP 的名称。
- 返回：由给定 GROUP 拥有的序列化 JicsResource 对象的列表。这些对象不按特定顺序返回，并且具有不同的类型（PROGRAM、TRANSACTION、FILE 等）。

编辑多个 LIST 中的 GROUP

- 支持的方法：POST

- 需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/api/services/rest/jicsservice/editGroupsList`
- 参数：带有子 GROUP 的 LIST 集合，以 JSON 表示形式；

  示例请求：

```
[
  {
        "jacType": "JACList",
        "name": "MURACHS",
        "isActive": true,
        "children": [
          {
            "jacType": "JACGroup",
            "name": "MURACHS",
            "isActive": true,
            "children": []
          },
          {
            "jacType": "JACGroup",
            "name": "TEST",
            "isActive": true,
            "children": []
          }
        ]
  }
]
```

  在进行此编辑之前，仅名为“MURACHS”的 GROUP 属于名为“MURACHS”的 LIST。通过此编辑，我们将名为“TEST”的 GROUP“添加”到名为“MURACHS”的 LIST 中。

- 返回一个布尔值。如果该值为“true”，则表示这些 LIST 修改已成功持久保存到底层 JICS 存储中。

删除 LIST

- 支持的方法：POST
- 需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/api/services/rest/jicsservice/deleteList`
- 参数：JSON 有效载荷，表示要删除的 JICS LIST。这是 `com.netfective.bluage.jac.entities.JACList` 对象的 JSON 序列化。

- 返回一个布尔值。如果该值为“true”，则表示已成功从底层 JICS 存储中删除该 LIST。

删除 GROUP

- 支持的方法：POST
- 需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/api/services/rest/jicsservice/deleteGroup`
- 参数：JSON 有效载荷，表示要删除的 JICS GROUP。这是 `com.netfective.bluage.jac.entities.JACGroup` 对象的 JSON 序列化。
- 返回一个布尔值。如果该值为“true”，则表示已成功从底层 JICS 存储中删除该 GROUP。

删除 TRANSACTION

- 支持的方法：POST
- 需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/api/services/rest/jicsservice/deleteTransaction`
- 参数：JSON 有效载荷，表示要删除的 JICS TRANSACTION。这是 `com.netfective.bluage.jac.entities.JACTransaction` 对象的 JSON 序列化。
- 返回一个布尔值。如果该值为“true”，则表示已成功从底层 JICS 存储中删除该 TRANSACTION。

删除 PROGRAM

- 支持的方法：POST
- 需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/api/services/rest/jicsservice/deleteProgram`
- 参数：JSON 有效载荷，表示要删除的 JICS PROGRAM。这是 `com.netfective.bluage.jac.entities.JACProgram` 对象的 JSON 序列化。
- 返回一个布尔值。如果该值为“true”，则表示已成功从底层 JICS 存储中删除该 PROGRAM。

删除 FILE

- 支持的方法：POST
- 需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER

- 路径：`/api/services/rest/jicsservice/deleteFile`
- 参数：JSON 有效载荷，表示要删除的 JICS FILE。这是 `com.netfective.bluage.jac.entities.JACFile` 对象的 JSON 序列化。
- 返回一个布尔值。如果该值为“true”，则表示已成功从底层 JICS 存储中删除该 FILE。

删除 TDQUEUE

- 支持的方法：POST
- 需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/api/services/rest/jicsservice/deleteTDQueue`
- 参数：JSON 有效载荷，表示要删除的 JICS TDQUEUE。这是 `com.netfective.bluage.jac.entities 的 JSON 序列化。 JACTDQueue`对象。
- 返回一个布尔值。如果该值为“true”，则表示已成功从底层 JICS 存储中删除该 TDQUEUE。

删除 TSMODEL

- 支持的方法：POST
- 需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/api/services/rest/jicsservice/deleteTSModel`
- 参数：JSON 有效载荷，表示要删除的 JICS TSMODEL。这是 `com.netfective.bluage.jac.entities 的 JSON 序列化。 JACTSModel`对象。
- 返回一个布尔值。如果该值为“true”，则表示已成功从底层 JICS 存储中删除该 TSMODEL。

删除元素

- 支持的方法：POST
- 需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/api/services/rest/jicsservice/deleteElements`
- 参数：JSON 有效载荷，表示要删除的 JICS 元素。
- 返回一个布尔值，`true` 表示已成功在底层 JICS 存储中执行删除操作。

创建 LIST

- 支持的方法：POST

- 需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/api/services/rest/jicsservice/createList`
- 参数：JSON 有效载荷，表示要创建的 JICS LIST。这是 `com.netfective.bluage.jac.entities 的 JSON 序列化。 JACList`对象。
- 返回一个布尔值。如果该值为“true”，则表示已成功在底层 JICS 存储中创建该 LIST。

> **Note**
>
> LIST 创建时始终为空。将 GROUP 附加到 LIST 需要执行另一项操作。

创建 GROUP

- 支持的方法：POST
- 需要身份验证和以下角色：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/api/services/rest/jicsservice/createGroup`
- 参数：JSON 有效载荷，表示要创建的 JICS GROUP。这是 `com.netfective.bluage.jac.entities.JACGroup` 对象的 JSON 序列化。
- 返回一个布尔值。如果值为“true”，则该 GROUP 已在底层 JICS 存储中正确创建。

> **Note**
>
> GROUP 创建时始终为空。将 RESOURCE 附加到 GROUP 需要额外的操作（创建资源会自动将其附加到给定的 GROUP）。

创建 RESOURCE 的常见注意事项

以下所有端点均与创建 JICS RESOURCE 有关，并具有一些共同的约束条件：在要发送到端点的请求负载中，必须对 `groupName` 字段赋值。

GROUP 所有权限制：

创建资源时必须将其附加到现有 GROUP，并且端点会使用 groupName 来检索此资源将要附加到的 GROUP。`groupName` 必须是现有 GROUP 的名称。如果 `groupName` 未指向 JICS 底层存储中的现有 GROUP，则将发送一条带有 HTTP 状态代码 400 的错误消息。

GROUP 内的唯一性约束：

具有给定名称的给定资源在指定的 GROUP 内必须唯一。每个资源创建端点均会执行唯一性检查。如果给定的有效载荷不符合唯一性约束条件，则端点将发送带有 HTTP 状态代码 400（错误请求）的响应，请参阅下面的示例响应。

示例有效载荷：您尝试在“TEST”GROUP 中创建事务“ARIT”，但该 GROUP 中已存在同名的事务。

```
{
    "jacType":"JACTransaction",
    "name":"ARIT",
    "groupName":"TEST",
    "isActive":true
  }
```

您会收到以下错误响应：

```
{
    "timestamp": 1686759054510,
    "status": 400,
    "error": "Bad Request",
    "path": "/jac/api/services/rest/jicsservice/createTransaction"
  }
```

检查服务器日志可确认问题的根源：

```

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
```

创建 TRANSACTION

- 支持的方法：POST

- 需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/api/services/rest/jicsservice/createTransaction`
- 参数：JSON 有效载荷，表示要创建的 JICS TRANSACTION。这是 `com.netfective.bluage.jac.entities.JACTransaction` 对象的 JSON 序列化。
- 返回一个布尔值。如果该值为“true”，则表示已成功在底层 JICS 存储中创建该 TRANSACTION。

创建 PROGRAM

- 支持的方法：POST
- 需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/api/services/rest/jicsservice/createProgram`
- 参数：JSON 有效载荷，表示要创建的 JICS PROGRAM。这是 `com.netfective.bluage.jac.entities.JACProgram` 对象的 JSON 序列化。
- 返回一个布尔值。如果该值为“true”，则表示已成功在底层 JICS 存储中创建该 PROGRAM。

创建 FILE

- 支持的方法：POST
- 需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/api/services/rest/jicsservice/createFile`
- 参数：JSON 有效载荷，表示要创建的 JICS FILE。这是 `com.netfective.bluage.jac.entities.JACFile` 对象的 JSON 序列化。
- 返回一个布尔值。如果该值为“true”，则表示已成功在底层 JICS 存储中创建该 FILE。

创建 TDQUEUE

- 支持的方法：POST
- 需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/api/services/rest/jicsservice/createTDQueue`
- 参数：JSON 有效载荷，表示要创建的 JICS TDQUEUE。这是 `com.netfective.bluage.jac.entities.JACTDQueue` 对象的 JSON 序列化。
- 返回一个布尔值。如果该值为“true”，则表示已成功在底层 JICS 存储中创建该 TDQUEUE。

创建 TSMODEL

- 支持的方法：POST
- 需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/api/services/rest/jicsservice/createTSModel`
- 参数：JSON 有效载荷，表示要创建的 JICS TSMODEL。这是 `com.netfective.bluage.jac.entities.JACTSModel` 对象的 JSON 序列化。
- 返回一个布尔值，`true` 表示已成功在底层 JICS 存储中执行元素创建操作。

创建元素

- 支持的方法：POST
- 需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/api/services/rest/jicsservice/createElements`
- 参数：JSON 有效载荷，表示要创建的 JICS 元素。
- 返回一个布尔值。如果该值为“true”，则表示已成功在底层 JICS 存储中创建这些元素。

更新 LIST

- 支持的方法：POST
- 需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/api/services/rest/jicsservice/updateList`
- 参数：JSON 有效载荷，表示要更新的 JICS LIST。这是 `com.netfective.bluage.jac.entities.JACList` 对象的 JSON 序列化。无需提供该 LIST 的子项，LIST 更新机制不会考虑子项。
- 返回一个布尔值。如果该值为“true”，则表示已成功在底层 JICS 存储中更新该 LIST。

更新 LIST“isActive”标志将传播到 LIST 中所有拥有的元素，即 LIST 拥有的所有 GROUP 以及这些 GROUP 拥有的所有 RESOURCE。这是一种通过单个操作在多个 GROUP 中停用大量资源的便捷方法。

更新 GROUP

- 支持的方法：POST

- 需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/api/services/rest/jicsservice/updateGroup`
- 参数：JSON 有效载荷，表示要更新的 JICS GROUP。这是 `com.netfective.bluage.jac.entities.JACGroup` 对象的 JSON 序列化。无需提供 GROUP 的子项，GROUP 更新机制也不会考虑这一点。
- 返回一个布尔值。如果该值为“true”，则表示已成功在底层 JICS 存储中更新该 GROUP。

> **Note**
>
> 更新 GROUP“isActive”标志将传播到 GROUP 中所有拥有的元素，即 GROUP 拥有的所有 RESOURCE。这是一种通过单个操作在给定 GROUP 中停用大量资源的便捷方法。

RESOURCE 更新常见注意事项

以下所有端点均与更新 JICS RESOURCE 有关。使用 `groupName` 字段，您可以更改任何 JICS RESOURCE 所属的 GROUP，前提是该字段值指向底层 JICS 存储中的现有 GROUP [否则，您将收到来自端点的错误请求响应（HTTP STATUS 400）]。

更新 TRANSACTION

- 支持的方法：POST
- 需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/api/services/rest/jicsservice/updateTransaction`
- 参数：JSON 有效载荷，表示要更新的 JICS TRANSACTION。这是 `com.netfective.bluage.jac.entities.JACTransaction` 对象的 JSON 序列化。
- 返回一个布尔值。如果该值为“true”，则表示已成功在底层 JICS 存储中更新该 TRANSACTION。

更新 PROGRAM

- 支持的方法：POST
- 需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/api/services/rest/jicsservice/updateProgram`
- 参数：JSON 有效载荷，表示要更新的 JICS PROGRAM。这是 `com.netfective.bluage.jac.entities.JACProgram` 对象的 JSON 序列化。

- 返回一个布尔值。如果该值为“true”，则表示已成功在底层 JICS 存储中更新该 PROGRAM。

更新 FILE

- 支持的方法：POST
- 需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/api/services/rest/jicsservice/updateFile`
- 参数：JSON 有效载荷，表示要更新的 JICS FILE。这是 `com.netfective.bluage.jac.entities.JACFile` 对象的 JSON 序列化。
- 返回一个布尔值。如果该值为“true”，则表示已成功在底层 JICS 存储中更新该 FILE。

更新 TDQUEUE

- 支持的方法：POST
- 需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/api/services/rest/jicsservice/updateTDQueue`
- 参数：JSON 有效载荷，表示要更新的 JICS TDQUEUE。这是 `com.netfective.bluage.jac.entities.JACTDQueue` 对象的 JSON 序列化。
- 返回一个布尔值。如果值为 “true”，则表示 TDQueue 已在底层 JICS 存储中成功更新。

更新 TSMODEL

- 支持的方法：POST
- 需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/api/services/rest/jicsservice/updateTSModel`
- 参数：JSON 有效载荷，表示要更新的 JICS TSMODEL。这是 `com.netfective.bluage.jac.entities.JACTSModel` 对象的 JSON 序列化。
- 返回一个布尔值。如果该值为“true”，则表示已成功在底层 JICS 存储中更新该 TSMODEL。

更新元素

- 支持的方法：POST
- 需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/api/services/rest/jicsservice/updateElements`

- 参数：JSON 有效载荷，表示要更新的元素。
- 返回一个布尔值，`true` 表示已成功在底层 JICS 存储中执行元素更新操作。

更新插入元素

- 支持的方法：POST
- 需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/api/services/rest/jicsservice/upsertElements`
- 参数：JSON 有效载荷，表示要更新插入的元素。
- 返回一个布尔值，`true` 表示已成功在底层 JICS 存储中执行元素更新插入操作。

检索元素

- 支持的方法：GET
- 需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/api/services/rest/jicsservice/retrieveElements`
- 参数：无
- 返回所有序列化 JICS 资源的列表。

JICS CRUD 操作

- 支持的方法：POST
- 需要身份验证和以下角色之一：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路径：`/api/services/rest/jicsservice/jicsCrudOperation`
- 参数：JSON 有效载荷，表示要查找的资源所在的 JICS 资源。这是 `com.netfective.bluage.jac.entities.request.JicsCrudOperationRequest` 对象的 JSON 序列化。
- 返回表示响应的 JSON 有效载荷。这是 `com.netfective.bluage.jac.entities.request.JicsCrudOperationResponse` 对象的 JSON 序列化。

其他

主题

- JICS 服务器运行状况

JICS 服务器运行状况

- 支持的方法：GET
- 路径：`/api/services/rest/jicsserver/serverIsUp`
- 参数：无
- 返回：无。HTTP STATUS 200 响应表示服务器已启动并正在运行。

JAC 用户管理端点

使用以下端点管理用户互动。

主题

- 以用户身份登录
- 测试系统中是否至少存在一个用户
- 记录新用户
- 用户信息
- 列出用户
- 删除用户
- 注销当前用户

以用户身份登录

- 支持的方法：POST
- 路径：`/api/services/security/servicelogin/login`
- 参数：无
- 返回 `com.netfective.bluage.jac.entities.SignOn` 对象的 JSON 序列化，表示在当前请求中提供凭证的用户。在返回的对象中，密码隐藏不可见，并且会列出赋予用户的角色。

示例响应:

```
{
```

```
    "login": "some-admin",
    "password": null,
    "roles": [
      {
        "id": 0,
        "roleName": "ROLE_ADMIN"
      }
    ]
  }
```

测试系统中是否至少存在一个用户

- 支持的方法：GET
- 路径：`/api/services/security/servicelogin/hasAccount`
- 参数：无
- 如果除默认超级管理员用户之外，还创建了至少一个用户，则会返回布尔值 `true`。否则返回 `false`。

记录新用户

- 支持的方法：POST
- 需要身份验证和 ROLE_ADMIN 角色。
- 路径：`/api/services/security/servicelogin/recorduser`
- 参数：`com.netfective.bluage.jac.entities.SignOn` 对象的 JSON 序列化，表示要添加到存储中的用户。应定义的用户角色，否则用户可能无法使用 JAC 设施和端点。
- 如果成功创建用户，则返回布尔值 `true`。否则返回 `false`。

示例请求：

```
{
    "login": "simpleuser",
    "password": "simplepassword",
    "roles": [
      {
        "id": 2,
        "roleName": "ROLE_USER"
      }
    ]
```

```
  }
```

记录新用户时，仅能使用以下角色：

- ROLE_ADMIN：可以管理 JICS 资源和用户。
- ROLE_USER：可以管理 JICS 资源，但不能管理用户。

用户信息

- 支持的方法：GET
- 路径：`/api/services/security/servicelogin/userInfo`
- 参数：无
- 返回当前连接用户的用户名和角色。

列出用户

- 支持的方法：GET
- 需要身份验证和 ROLE_ADMIN 角色。
- 路径：`/api/services/security/servicelogin/listusers`
- 参数：无
- 返回 `com.netfective.bluage.jac.entities.SignOn` 列表，采用 JSON 序列化。

删除用户

- 支持的方法：POST
- 需要身份验证和 ROLE_ADMIN 角色。
- 路径：`/api/services/security/servicelogin/deleteuser`
- 参数：`com.netfective.bluage.jac.entities.SignOn` 对象的 JSON 序列化，表示要从存储中删除的用户。
- 如果成功删除用户，则返回布尔值 `true`。

> **Important**
>
> 并且无法撤消。已删除的用户将无法再次连接到 JAC 应用程序。

注销当前用户

- 支持的方法：GET
- 路径：`/api/services/security/servicelogout/logout`
- 参数：无
- 如果成功注销当前用户，则返回 JSON 消息 `{"success":true}`。相关的 HTTP 会话将失效。

AWS Blu Age 用户的数据结构

你可以在下一节中了解 AWS Blu Age 引擎的各种数据结构。

主题

作业执行详细信息消息结构

每个作业执行的详细信息都将包含以下字段：

scriptId

被调用脚本的标识符。

caller

调用方的 IP 地址。

identifier

唯一的作业执行标识符。

startTime

作业执行开始日期和时间。

endTime

作业执行结束日期和时间。

status

作业执行的状态。可能为以下值之一：

- DONE：作业执行正常结束。
- TRIGGERED：作业执行已触发但尚未启动。
- RUNNING：作业执行正在运行。
- KILLED：作业执行已被终止。
- FAILED：作业执行失败。

executionResult

用于汇总作业执行结果的消息。如果作业执行尚未完成，则此消息可以是简单消息，也可以是包含以下字段的 JSON 结构：

- exitCode：数字退出码；负值表示故障情况。
- program：该任务启动的最新程序。
- status：可能为以下值之一：
  - Error：当 exitCode = -1 时；该值表示作业执行期间发生（技术）错误。
  - Failed：当 exitcode = -2 时；该值表示服务程序执行期间发生故障（例如 ABEND 情况）。
  - Succeeded：当 exitCode >= 0 时；
- stepName：作业中最近执行的步骤的名称。

executionMode

可以是 SYNCHRONOUS 或 ASYNCHRONOUS，具体取决于作业的启动方式。

示例输出：

```
{
    "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"
  }
```

事务启动结果结构

该结构可以包含以下字段：

outCome

代表事务执行结果的字符串。可能的值有：

- `Success`：事务执行正常结束。
- `Failure`：事务执行未能正常结束，遇到了一些问题。

commarea

表示 COMMAREA 最终值（byte64 编码字节数组）的字符串，可以是空字符串。

containerRecord

（可选）表示容器的记录内容（byte64 编码字节数组）的字符串。

serverDescription

可能包含有关服务于请求的服务器信息（用于调试目的），可以是空字符串。

abendCode

（可选）如果启动的事务所引用的程序中止，则在此字段中将以字符串形式返回中止代码值。

示例响应：

成功

```
{
    "outCome": "Success",
    "commarea": "",
    "serverDescription": ""
  }
```

Failure

```
{
    "outCome": "Failure",
```

```
    "commarea": "",
    "serverDescription": "",
    "abendCode": "AEIA"
  }
```

事务启动记录结果结构

该结构可以包含以下字段：

recordContent

表示 COMMAREA 的记录内容（byte64 编码字节数组）的字符串。

containerRecord

表示容器的记录内容（byte64 编码字节数组）的字符串。

serverDescription

可能包含有关服务于请求的服务器信息（用于调试目的），可以是空字符串。

示例响应：

成功

```
{
    "recordContent": "",
    "serverDescription": ""
}
```

队列中作业的可能状态

在队列中，作业可具有以下状态：

ACTIVE

作业目前正在队列中运行。

EXECUTION_WAIT

作业正在等待线程可用。

SCHEDULED

作业计划在特定的日期和时间执行。

HOLD

作业正在等待释放后再运行。

COMPLETED

作业已成功执行。

FAILED

作业执行失败。

UNKNOWN

状态未知。

提交作业和计划作业输入

提交作业和计划作业输入是 `com.netfective.bluage.gapwalk.rt.jobqueue.SubmitJobMessage` 对象的 JSON 序列化。以下示例输入显示了此类 Bean 的所有字段。

提交作业的示例输入：

```
{
    "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
}
```

计划作业的示例输入：

```
{
```

```
    "scheduleCron": "*/2 * * * * ?",
    "programName":"LOGPGM",
    "programParams": {
        "cl_sbmjob_param_json": "[\"./output/schedule-job-log.txt\", \"Every 2 
seconds!\"]"
    },
    "localDataAreaValue":"",
    "userName":"PVO",
    "jobName":"LOGGERJOB",
    "jobPriority":5,
    "jobQueue":"queue1",
    "scheduleMisfirePolicy": 4,
    "startTime": "2003/05/04 07:00:00.000 GMT-06:00",
    "endTime": "2003/05/04 07:00:07.000 GMT-06:00"
}
```

jobNumber

如果作业号为 0，则将使用作业编号序列中的下一个编号自动生成作业编号。该值应设置为 0（测试目的除外）。

jobPriority

AS400 中的默认任务优先级为 5。有效范围为 0 – 9，0 为最高优先级。

jobOnHold

如果作业被暂停提交，则不会立即执行，而仅在有人将其“释放”时才会执行。可以使用 REST API（/release 或 /releasease-all）释放作业。

scheduleDate 和 scheduleTime

如果这些值不为空，则作业将在指定的日期和时间执行。

Date

可以提供 format MMddyy 或 ddMMyyyy （输入的大小将决定使用哪种格式）

Time

可以提供格式 HHmm 或 HHmmss （输入的大小将决定使用哪种格式）

programParams

将作为映射传递给程序。

scheduleMisfirePolicy

定义触发器失效时使用的策略。有以下可能值：

1. 释放第一个失效并丢弃其他失效。
2. 提交因第一个失效而暂停的作业并丢弃其他失效。
3. 丢弃失效。
4. 释放所有失效。作业队列将运行所有作业。

计划作业响应列表

以下是列表作业的作业队列端点的结构。用于提交该作业的提交作业消息是响应的一部分。此端点结构可用于追踪或测试/重新提交。作业完成后，开始日期和结束日期也会填充。

```
[
  {
    "jobName": "PTA0044",
    "userName": "USER1",
    "jobNumber": 9,
    "jobPriority": 5,
    "status": "HOLD",
    "jobDelay": 0,
    "startDate": null,
    "endDate": null,
    "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,
      "scheduleCron": null,
      "save": false,
      "scheduleMisfirePolicy": 4,
```

```
      "omitdates": null
    },
    "executionId": 1,
    "jobScheduledId": 0,
    "jobScheduledAt": null
  },
  {
    "jobName": "PTA0044",
    "userName": "USER1",
    "jobNumber": 9,
    "jobPriority": 5,
    "status": "COMPLETED",
    "jobDelay": 0,
    "startDate": "2022-10-13T22:48:34.025+00:00",
    "endDate": "2022-10-13T22:52:54.475+00:00",
    "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,
      "scheduleCron": "*/20 * * * * ?",
      "save": false,
      "scheduleMisfirePolicy": 4,
      "omitdates": null
    },
    "executionId": 2,
    "jobScheduledId": 0,
    "jobScheduledAt": null
  }
]
```

重复作业响应列表

这是the /schedule/list作业队列端点的结构。


```
[
  {
    "id": 1,
    "status": "ACTIVE",
    "jobNumber": 1,
    "userName": "PVO",
    "msg": {
      "messageQueueName": null,
      "scheduleDate": null,
      "scheduleTime": null,
      "startTime": "2024/03/07 21:12:00.000 UTC",
      "endTime": "2024/03/07 21:13:59.000 UTC",
      "programName": "LOGPGM",
      "programParams": {"cl_sbmjob_param_json": "[\"./output/schedule-job-log.txt\",
 \"Every 20 seconds!\"]"},
      "localDataAreaValue": "",
      "userName": "PVO",
      "jobName": "LOGGERJOB",
      "jobNumber": 1,
      "jobScheduleId": 1,
      "jobPriority": 5,
      "executionDate": null,
      "jobQueue": "queue1",
      "jobOnHold": false,
      "scheduleCron": "*/20 * * * * ?",
      "save": false,
      "scheduleMisfirePolicy": 4,
      "omitdates": null
    },
    "lastUpdatedAt": "2024-03-07T21:11:13.282+00:00",
    "lastUpdatedBy": ""
  }
]
```


AWS Blu Age 运行时实用工具

大型机 JCL 脚本使用客户业务程序和系统实用程序。 AWS Blu Age Runtime 包括对常用系统实用程序的支持，以实现对 Groovy 的 JCL 适当现代化，以及其他便捷实用程序。

本文档的目的是：

- 解释如何配置和部署实用程序；
- 列出现有的支持的系统实用程序，并详细说明其用法和用途。

配置和部署实用程序 Web 应用程序

中提供了实用程序`gapwalk-utility-pgm-<version>.war`（Blu `<version>` Age 运行时版本在哪里）。与应用程序一起部署`gapwalk-application`并实现现代化，以允许从现代化的 JCL 脚本访问实用程序。

所有支持的实用程序都在 “程序注册表” 中注册the section called “运行和调用程序”为程序（有关注册程序以及在脚本或其他程序中使用它们的详细信息，请参阅此处）。因此，它们可以由其他程序或现代化的作业脚本调用。

实用程序 Web 应用程序有自己的名为的配置文件`application-utility-pgm.yml`，其内容详见此处。the section called “为托管应用程序配置对实用程序的访问权限”

可用的实用程序

根据其应用领域列出的可用实用程序：

主题

数据集实用工具

BLUESAMCOPY/BLUESAMCREATE/BLUESAMDELETE/BLUESAMCLEAR

这些实用程序在 Blusam 数据集上提供了基本的数据集操作功能。

标bypassBluesamStatus志（存储在 JobContext）决定是否绕过常规数据集状态验证，该验证可防止联机进程和批处理进程之间的并发访问冲突。

当 bypassBluesamStatus 为 true 时：

- 在操作期间跳过常规数据集状态检查
- 允许访问通常可能被屏蔽的数据集

BLUESAMCOPY

BLUESAMCOPY 程序将数据和元数据结构从一个 Bluesam 数据集复制到另一个数据集。

参数

BLUESAMCOPY 程序采用两个参数：

- FROM=<source dataset name>-要复制的源数据集
- TO=<target dataset name>-要复制到的目标数据集

行为

- 如果目标数据集不存在，则使用源的元数据结构创建它
- 如果目标存在，则在复制之前将其清除（覆盖现有数据）
- 当联机进程未使用数据集或设置bypassBluesamStatus为 true 时，操作将继续
- 保留密钥、压缩、记录长度和其他元数据属性

错误处理

如果源数据集不存在或者在线进程正在使用其中一个数据集，则返回代码 1。

示例用法

```
import ...

mpr =
 applicationContext.getBean("com.netfective.bluage.gapwalk.rt.call.ExecutionController",MainPro

// copy bluesam dataset
res =
 mpr.withArguments("FROM=MYBLUESAMTABLE","TO=MYBLUESAMTABLE2").runProgram("BLUESAMCOPY")
```

BLUESAMCREATE

BLUESAMCREATE 程序使用指定的参数创建一个 Bluesam 数据集。

参数

BLUESAMCREATE 程序需要七个参数：

- name=<dataset name>-要创建的数据集（必填）
- compress=<true/false>-启用压缩（可选，默认值：false）
- fixedLength=<true/false>-固定长度的记录状态（可选，默认值：false）
- recordLength=<integer>-以字节为单位的记录长度（可选，默认值：-1）
- primaryKey=<key id>,duplicates=<true/false>,fields=<offset,length,...>-主键规范（可选）
- key=<altkey id>,duplicates=<true/false>,fields=<offset,length,...>-备用密钥（可以有多个或 0）
- clearExisting=true/false-清除现有数据集（可选默认值：true）

具体按键参数说明：

- duplicates: 给定密钥是否允许重复值；
- fields: 字段位置（从 1 开始）和定义密钥的长度；

行为

- 如果数据集不存在，则使用指定的参数创建它
- 如果数据集存在，则在以下情况下将其清除 clearExisting=true
- 当联机进程未使用数据集或设置bypassBluesamStatus为时，操作将继续 true

错误条件

在所有情况下都返回代码 0。

示例用法

```
import ...

mpr =
 applicationContext.getBean("com.netfective.bluage.gapwalk.rt.call.ExecutionController",MainPro

// create bluesam dataset
```

```
mpr.withArguments(
      "name=MYBLUESAMTABLE",
      "compress=FALSE",
      "fixedLength=true",
      "recordLength=54",
      "primaryKey=MYKEY_PK,duplicates=FALSE,fields=0,6")
    .runProgram("BLUESAMCREATE")
```

BLUESAMDELETE

BLUESAMDELETE 程序会删除 Bluesam 数据集。

参数

BLUESAMDELETE 程序需要 1 个或多个参数：

- `<dataset name>`-要删除的数据集（可以有多个）

行为

- 当联机进程未使用数据集或设置`bypassBluesamStatus`为时，操作将继续 `true`
- 如果数据集存在，请将其删除

错误条件

在所有情况下都返回代码 0。

示例用法

```
import ...

mpr =
 applicationContext.getBean("com.netfective.bluage.gapwalk.rt.call.ExecutionController",MainPro

// delete bluesam datasets
res =
 mpr.withArguments("MYBLUESAMTABLE","MYBLUESAMTABLE2","MYBLUESAMTABLE3").runProgram("BLUESAMDEL
```

BLUESAMCLEAR

BLUESAMCLEAR 程序从现有数据集中删除所有数据，同时保留其结构和元数据。

参数

BLUESAMCLEAR 程序需要 1 个或多个参数：

- `<dataset name>`-要清除的数据集（可以有多个）

行为

- 当联机进程未使用数据集或设置`bypassBluesamStatus`为时，操作将继续 `true`
- 如果数据集存在，请将其清除

错误条件

在所有情况下都返回代码 0。

示例用法

```
import ...

mpr =
 applicationContext.getBean("com.netfective.bluage.gapwalk.rt.call.ExecutionController",MainPro

// clear bluesam datasets
res = mpr.withArguments("MYBLUESAMTABLE","MYBLUESAMTABLE2").runProgram("BLUESAMCLEAR")
```

BPXWDYN

该实用程序模拟 z/OS BPXWDYN 服务，用于动态数据集分配和解除分配。在现代应用程序中，文件分配是通过`FileConfigurationUtils`生成器 api 在 Groovy 脚本中完成的。该程序允许在没有 Groovy 的情况下动态管理这些分配，这对于现代化应用程序中的交互式或条件文件处理至关重要。

参数

保留了旧的参数格式。需要一个包含一个 2 字节大小的标头和命令字符串的参数。

用于分配的命令字符串格式：`ALLOC DD(<name>) DSN('<dsn>') SHR?`

用于取消分配的命令字符串格式：`FREE DD(<name>)`

错误处理

- 为成功设置错误代码 0，为失败设置错误代码 1

- RuntimeException 对无效的命令或参数进行抛出

示例用法

以下是 BPXWDYN 程序的 Java 示例用法，该程序源于通过 AWS 转换对 COBOL 进行现代化改造：

COBOL 代码：

```
   01  WK-AREA.
       03  DS-ALLOC-STRING.
           05  DS-LENGTH          PIC S9(004) COMP VALUE 100.
           05  DS-TEXT            PIC  X(100) VALUE
            "ALLOC DD(INFILE) DSN('A.B.JCLLIB(XYZ470)') SHR".
...
           01  RC-RETURN-CODE-AREA.
       03  RC-RETURN-CODE         PIC S9(008) COMP.
...
     CALL  'BPXWDYN'  USING  DS-ALLOC-STRING RETURNING RC-RETURN-CODE.
```

以及相应的 Java 现代化：

```
private final Group dsAllocString = new Group(root).named("DS-ALLOC-STRING");
private final Elementary dsLength = new Elementary(dsAllocString,new BinaryType(4, 0,
 "STD", false, false, true),Short.valueOf("100")).named("DS-LENGTH");
private final Elementary dsText = new Elementary(dsAllocString,new
 AlphanumericType(100),"ALLOC DD(INFILE) DSN('A.B.JCLLIB(XYZ470)') SHR").named("DS-
TEXT");
...
private final Group root = new Group(getData()).named("RC-RETURN-CODE-AREA");
private final Elementary rcReturnCode = new Elementary(root,new BinaryType(8, 0, "STD",
 false, false, true)).named("RC-RETURN-CODE");
...
    // Call to utility program  BPXWDYN
ctrl.callSubProgram(
       "BPXWDYN",

 CallBuilder.newInstance().byReference(ctx.getWkArea().getDsAllocStringReference()).getArgument
 ctx);

ctx.getRcReturnCodeArea().setRcReturnCode(NumberUtils.convert(ctx.getProgramReturned()).intValu
```

GDGUTILS

GDGs （生成数据组）允许应用程序使用版本化数据集，其中每次执行都会创建新一代数据集，同时保持对前几代数据的访问权限。此实用程序创建并管理这些数据集的生成。此实用程序本应在 groovy 脚本中调用。

参数

参数顺序无关紧要

- `action=<create|refreshevents>`-要执行的操作（必须）
- `gdgname=<name>`-GDG 基地名称（必填）
- `storageProvider=<filesystem|bluesam>`-存储后端
- `relativeGeneration=<+integer>`-相对世代号（例如，+1 表示下一代）
- `absoluteGeneration=integer`-绝对世代号
- `recordLength=<integer>`-数据集中的记录大小
- `fixedLength=<true/false>`-指定数据集中记录的长度是固定的
- `ownerPath=<File system path>`-存储数据集的路径（特定于文件系统，在这种情况下是必需的）
- `compress=<true/false>`-表示如果在数据存储中压缩数据，则数据应在内存中保持压缩状态（特定于 bluesam）（可选，默认值为 false）
- `catalog`-指定要对数据集进行编目（可选）
- `warmUp`-表示数据集在打开时应加载到内存中（特定于 bluesam）（可选）

relativeGeneration 或 AbsoluteGeneration：需要设置其中一个属性。

操作

- 创建：根据 GDG 元数据生成新的 GDG 数据集（同时处理文件系统和 Bluesam 存储）
- refresheevents：在不创建新数据集的情况下调整世代号（Gdg 元数据）（在重启已创建数据集的失败作业时使用）

错误处理

- 为成功设置错误代码 0，为失败设置错误代码 1
- RuntimeException 对无效的命令或参数进行抛出

示例用法

Gdg 创建操作：以下代码为数据集创建 bluesam 生成 (43) `IC.PLI.GDGTEST`

```
import ...

mpr =
 applicationContext.getBean("com.netfective.bluage.gapwalk.rt.call.ExecutionController",MainPro

Map params = new Hashtable()
params.put("jobContext", jobContext)
Object[] args
 =["action=create","gdgname=IC.PLI.GDGTEST","absoluteGeneration=43","storageProvider=bluesam","
mpr.withParameters(params).withArguments(args).runProgram("GDGUTILS")
```

Gdg 刷新事件操作：只有 relativeGeneration 与此操作相关。以下代码更新`IC.PLI.GDGTEST`数据集的世代号（与当前一代相比 +1）

```
import ...

mpr =
 applicationContext.getBean("com.netfective.bluage.gapwalk.rt.call.ExecutionController",MainPro

Map params = new Hashtable()
params.put("jobContext", jobContext)
Object[] args
 =["action=refreshevents","gdgname=IC.PLI.GDGTEST","relativeGeneration=1","storageProvider=blue
mpr.withParameters(params).withArguments(args).runProgram("GDGUTILS")
```

ICEGENER/SYNCGENR

该实用程序模仿了 z/OS 系统实用程序 ICEGENER 的行为，它将数据集从输入（数据集）复制到输出（SYSUT1 SYSUT2 数据集）。此 Java 实现提供了支持文件系统和 Bluesam 存储的同等功能。

参数

没有争论

必需的数据集

- SYSUT1: 输入数据集/文件
- SYSUT2: 输出数据集/文件

SYSUT2 数据集的处置处理

- 新增：创建新的数据集/文件
- OLD/SHR: Use existing dataset/file（必须存在）
- MOD：修改，如果缺少则创建，如果存在则追加

错误处理

- 如果复制成功则设置错误代码 0，如果复制失败则设置为 1
- 因使用 Bluse `IllegalStateException` am 数据集的无效而被抛出

示例用法

以下是 ICEGENER 计划的 Groovy 使用示例，该计划源于 JCL 通过 AWS 转型进行现代化改造：

JCL 的代码：

```
//STEP01    EXEC PGM=ICEGENER
//SYSUT1   DD DSN=POI.INPUT,DISP=SHR
//SYSUT2   DD DSN=POI.OUTPU,
//            DISP=(,CATLG,DELETE),
//            UNIT=3490,
//            DCB=(RECFM=FB,LRECL=100)
/*
```

以及相应的 Groovy 现代化改造：

```
mpr
    .withFileConfigurations(new FileConfigurationUtils()
        .withJobContext(jobContext)
        .bluesam("SYSUT1")
        .dataset("POI.INPUT")
        .disposition("SHR")
        .build()
        .bluesam("SYSUT2")
        .dataset("POI.OUTPU")
        .normalTermination("CATLG")
        .abnormalTermination("DELETE")
        .build()
        .getFileConfigurations())
```

```
    .withParameters(params)
    .runProgram("ICEGENER")
```

IDCAMS/KQCAMS

该实用程序模仿传统程序 IDCAMS 的行为，IDCAMS 是一种用于 VSAM（虚拟存储访问方法）文件操作的大型机数据管理工具。它处理传统的 IDCAMS 命令，保持与原始 SYSIN 输入相同的语法。

上下文

程序行为可以通过以下两个参数进行配置`application-utility-pgm.yml`：

- jclType：JCL 类型标识符（vse 或 mvs）。如果非 vse jcl 文件为空，则 IDCAMS 实用程序 PRINT/REPRO 命令返回 4
- forcedCharsetIdcams: IDCAMS 处理的可选字符集替换

参数

没有争论。操作通过 SYSIN 数据集传递。

必需的数据集

- SYSIN-包含 IDCAMS 命令语句
- 输入/输出数据集-如IDCAMS命令中所述（取决于IDCAMS声明）

主要功能/支持的命令

此处未提供有关 SYSIN 控制卡中的 IDCAMS 命令的详细信息，但应从现有的相关传统平台文档中获取。

- 定义-创建 VSAM 群集和数据集
- 删除-移除数据集（支持通配符）
- REPRO-在数据集之间复制数据
- 打印-显示数据集内容
- 验证-验证数据集的存在和完整性
- ALTER-修改数据集属性（重命名）
- ALLOC-动态分配数据集
- SET-管理条件代码 (LASTCC/MAXCC)

- IF-THEN-ELSE -有条件的命令执行
- 取消-终止作业执行

错误处理

- 如果最后一个命令成功则设置错误代码 0，如果失败则设置为 -1
- SET (LASTCC) 命令可以用来覆盖错误代码，例如 `SET LASTCC = 0`

示例用法

以下是 IDCAMS 计划的 Groovy 使用示例，该计划源于 JCL 通过 AWS 转型进行现代化改造：

JCL 的代码：

```
//STEP15 EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=*
//ACCTDATA DD DISP=SHR,
//          DSN=AWS.M2.CARDDEMO.ACCTDATA.PS
//ACCTVSAM DD DISP=SHR,
//          DSN=AWS.M2.CARDDEMO.ACCTDATA.VSAM.KSDS
//SYSIN    DD   *
  REPRO INFILE(ACCTDATA) OUTFILE(ACCTVSAM)
/*
```

以及相应的 Groovy 现代化改造：

```
mpr.withFileConfigurations(new FileConfigurationUtils()
    .withJobContext(jobContext)
    .systemOut("SYSPRINT")
    .output("*")
    .build()
    .bluesam("ACCTDATA")
    .dataset("AWS.M2.CARDDEMO.ACCTDATA.PS")
    .disposition("SHR")
    .build()
    .bluesam("ACCTVSAM")
    .dataset("AWS.M2.CARDDEMO.ACCTDATA.VSAM.KSDS")
    .disposition("SHR")
    .build()
    .fileSystem("SYSIN")
    .stream("REPRO INFILE(ACCTDATA) OUTFILE(ACCTVSAM)", getEncoding())
```

```
    .build()
    .getFileConfigurations())
.withParameters(params)
.runProgram("IDCAMS")
```

IEBGENER/JSDGENER

该程序复制了 IEBGENER 实用程序。它用于复制和操作顺序数据集。此实现通过支持用于高级数据处理操作的 IEBGENER 控制语句扩展了基本的复制功能。

参数

没有争论。操作通过 SYSIN 数据集传递。

必需的数据集

- SYSIN：包含控制语句（可选，如果未定义，IEBGENER 程序与 ICEGENER 程序相同）
- SYSUT1: 输入数据集/文件
- SYSUT2: 输出数据集/文件

主要特性/支持的 IEBGENER 控制语句

此处未提供有关 SYSIN 控制卡中的 IEBGENER 控制语句的详细信息，但应从现有的相关传统平台文档中获取。

- 生成-通过指定要处理的最大记录类型 (MAXNAME) 和字段 (MAXFLDS) 数来定义数据操作过程的整体结构
- 记录-通过指定将从输入数据集中复制或使用特定值生成的各个字段的位置、长度和格式来定义每种记录类型的实际布局和内容
- LABEL/MEMBER/EXIT不支持

示例：

```
GENERATE MAXNAME=3,MAXFLDS=5
RECORD TYPE=1,
       FIELD=(1,1,CH,VALUE='H'),
       FIELD=(2,30,CH,VALUE='EMPLOYEE REPORT 2024         ')
RECORD TYPE=2,
       FIELD=(1,1,CH,VALUE='D'),
       FIELD=(2,10,CH),             /* Name */
```

```
        FIELD=(12,8,CH),             /* Birth date */
        FIELD=(20,8,CH,VALUE='ACTIVE')
RECORD TYPE=3,
        FIELD=(1,1,CH,VALUE='F'),
        FIELD=(2,30,CH,VALUE='END OF REPORT              ')
```

错误处理

如果复制成功，则设置错误代码 0，如果复制失败，则设置错误代码 1。

示例用法

以下是 IEBGENER 计划的 Groovy 使用示例，该计划源于 JCL 通过 AWS 转型进行现代化改造：

JCL 的代码：

```
//GENDATA  EXEC PGM=IEBGENER
//SYSUT1   DD DSN=INPUT.EMPLOYEE.DATA,
//            DISP=SHR
//SYSUT2   DD DSN=OUTPUT.EMPLOYEE.FILE,
//            DISP=(NEW,CATLG,DELETE),
//            SPACE=(TRK,(1,1)),
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=27920)
//SYSIN    DD *
  GENERATE MAXNAME=2,MAXFLDS=5
  RECORD TYPE=1,
         FIELD=(1,1,CH,VALUE='H'),
         FIELD=(2,30,CH,VALUE='EMPLOYEE SALARY REPORT 2024    ')
  RECORD TYPE=2,
         FIELD=(1,1,CH,VALUE='D'),
         FIELD=(2,10,CH),              /* Name from input */
         FIELD=(12,8,CH),             /* Birth date from input */
         FIELD=(20,8,CH),             /* Dept from input */
         FIELD=(28,8,CH)              /* Salary from input */
/*
```

以及相应的 Groovy 现代化改造：

```
mpr
    .withFileConfigurations(new FileConfigurationUtils()
        .withJobContext(jobContext)
        .systemOut("SYSPRINT")
        .output("*")
        .build()
```

```
        .bluesam("SYSUT1")
        .dataset("INPUT.EMPLOYEE.DATA")
        .disposition("SHR")
        .build()
        .bluesam("SYSUT2")
        .dataset("OUTPUT.EMPLOYEE.FILE")
        .disposition("NEW")
        .normalTermination("CATLG")
        .abnormalTermination("DELETE")
        .build()
        .fileSystem("SYSIN")
        .stream( """GENERATE MAXNAME=2,MAXFLDS=5
  RECORD TYPE=1,
         FIELD=(1,1,CH,VALUE='H'),
         FIELD=(2,30,CH,VALUE='EMPLOYEE SALARY REPORT 2024    ')
  RECORD TYPE=2,
         FIELD=(1,1,CH,VALUE='D'),
         FIELD=(2,10,CH),               /* Name from input */
         FIELD=(12,8,CH),              /* Birth date from input */
         FIELD=(20,8,CH),              /* Dept from input */
         FIELD=(28,8,CH)               /* Salary from input */""", getEncoding())
        .build()
        .getFileConfigurations())
    .withParameters(params)
    .runProgram("IEBGENER")
```

IEFBR14

IEFBR14 是一个 “什么都不做” 的程序，返回码为 0（零）。它的主要用途是通过 DD 语句进行数据集分配、删除或目录维护，而无需执行任何实际的数据处理。

参数

没有争论

错误处理

始终设置错误代码 0。

示例用法

以下是该 IEFBR14 程序的 Groovy 使用示例，该示例是通过 AWS 转型实现的 JCL 现代化改造：

创建新的顺序数据集的 JCL 代码：

```
//STEP1    EXEC PGM=IEFBR14
//NEWSEQ   DD DSN=USER.NEW.SEQ.DATA,
//            DISP=(NEW,CATLG,DELETE),
//            SPACE=(TRK,(10,5)),
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=27920)
```

以及相应的 Groovy 现代化改造：

```
mpr
    .withFileConfigurations(new FileConfigurationUtils()
        .withJobContext(jobContext)
        .bluesam("NEWSEQ")
        .dataset("USER.NEW.SEQ.DATA")
        .disposition("NEW")
        .normalTermination("CATLG")
        .abnormalTermination("DELETE")
        .build()
        .getFileConfigurations())
    .withParameters(params)
    .runProgram("IEFBR14")
```

删除现有数据集的 JCL 代码：

```
//STEP3    EXEC PGM=IEFBR14
//DELDD    DD DSN=USER.OLD.DATASET,
//            DISP=(OLD,DELETE,DELETE)
```

以及相应的 Groovy 现代化改造：

```
mpr
    .withFileConfigurations(new FileConfigurationUtils()
        .withJobContext(jobContext)
        .bluesam("DELDD")
        .dataset("USER.OLD.DATASET")
        .disposition("OLD")
        .normalTermination("DELETE")
        .abnormalTermination("DELETE")
        .build()
        .getFileConfigurations())
    .withParameters(params)
    .runProgram("IEFBR14")
```

JCLBCICS

该程序管理数据集的状态，它基于配置 enables/disables 的数据集，同时支持单个文件和通配符模式：它更改 JICS 表 file_table 中的 STATUS 字段。

参数

没有争论。操作通过`DatasetsConfiguration`对象传递，例如

```
mpr.withDatasetsConfiguration(new DatasetsConfiguration().close(<Dataset Name>)
```

上下文

程序行为可以通过两个参数进行配置：

In `application-utility-pgm.yml`:

- `jclbcics.ddname.size`（默认为 8）：它全局配置数据集名称大小；如果设置了此值且数据集名称长度小于此值，则数据集名称将被截断。

在 Groovy 文件的各个步骤中调用程序

- `JCLBCICS_OVERRIDDEN_SIZE`:它会覆盖全局数据集名称大小：

```
TreeMap stepMapTransfo = [:]
Map stepParams = ["MapTransfo":stepMapTransfo]
stepParams["MapTransfo"]["JCLBCICS_OVERRIDDEN_SIZE"] = '6'
```

如果调整后的 dd 名称大小（截断后）小于 8，则 dd 名称将被视为通配符，并且该函数适用于以此 dd 名称开头的所有数据集。

主要功能

支持的操作有：

- 打开：将数据集设置为 “已启用” 状态
- 关闭：将数据集设置为禁用状态

这些操作是通过`DatasetsConfiguration`生成器类声明的：

```
new DatasetsConfiguration().close(<DD name>).open(<DD name>)
```

DD 名称：数据集的名称，如果 dd 名称大小小于最大 dd 名称大小 (8)，则接受通配符 *。

错误处理

将错误代码设置为 0

示例用法

以下是 JCLBCICS 程序的 Groovy 示例用法：

它禁用数据集`UFOLJ3P`，启用所有以 AX 开头的数据集，启用所有以 DX 开头的数据集

```
import com.netfective.bluage.gapwalk.rt.call.MainProgramRunner
import com.netfective.bluage.gapwalk.rt.call.ProgramExecutionResult
import com.netfective.bluage.gapwalk.rt.io.support.DatasetsConfiguration

MainProgramRunner mpr =
 applicationContext.getBean("com.netfective.bluage.gapwalk.rt.call.ExecutionController",MainPro
def TreeMap stepMapTransfo = [:]
def Map stepParams = ['MapTransfo':stepMapTransfo]
stepParams['MapTransfo']['JCLBCICS_OVERRIDDEN_SIZE'] = '7'

ProgramExecutionResult res = mpr
.withDatasetsConfiguration(new
 DatasetsConfiguration().close("UFOLJ3P").open("AX*").open("DX"))
.withParameters(stepParams)
.runProgram("JCLBCICS")
```

数据库实用程序

本节介绍的是与数据库相关的实用程序。在传统平台上，这些程序通常在 DB2 数据库上运行，而在现代环境中，则会选择其他数据库（AWS Aurora 是一种流行的选择）。

DSNTEP2/DSNTEP4

DSNTEP2/DSNTEP4 目的

DNSTEP 实用程序从输入数据集执行 SQL 查询，并将结果写入输出数据集。它会自动为现代化的目标数据库（超越 DB2）重写旧版 SQL 查询，并支持多种`SYSIN`数据集类型：流、Blusam 数据集、平面文件、GDG 生成和数据集串联。

DSNTEP2/DSNTEP4 签名

该实用程序不带任何参数，使用两个数据集：

- SYSIN: 包含 SQL 语句的输入数据集
- SYSPRINT: 查询结果的输出数据集（仅限SELECT查询）

DSNTEP2/DSNTEP4 相关配置参数

该实用程序的行为受以下配置参数的影响：

- unload.sqlCodePointShift
- unload.noPad
- unload.nbi.whenNull
- unload.nbi.whenNotNull
- unload.useDatabaseConfiguration
- unload.format.date
- unload.format.time
- unload.format.timestamp
- hasGraphic
- forcedDate
- frozenDate

有关配置这些参数[the section called “可选 Web 应用程序的可用属性”]的详细信息，请参阅。

DSNTEP2/DSNTEP4 检查/错误处理

- 如果由于任何原因在查询运行期间发生异常，则将记录一条错误消息并抛出，从而导致当前运行单元停止。StopRunUnitException
- 如果SYSIN是各种数据集的串联而其中任何一个数据集不受支持，则会抛出 a RuntimeException。当前，当与 /4 一起用作输入时，仅支持将平面文件和 GDG 世代类型作为连接文件的一部分。 DSNTEP2

DSNTEP2/用法DSNTEP4 示例

以下是 JCL 的用法 DNSTEP4示例：

```
//*****************************************************************
//* RETRIEVE DATA FROM TABLE AP_JBI7_INVOICE                    *
//*****************************************************************
//*
//DSNTEP03 EXEC PGM=DSNTEP4,DYNAMNBR=20
//SYSPRINT DD DSN=output(out012.txt),
//            DISP=SHR,DCB=(RECFM=FB,LRECL=1152)
//SYSIN    DD *
  SELECT * FROM BUR000.AP_JBI7_INVOICE WITH UR;
```

以及匹配的现代化 groovy 脚本片段：

```
def stepDSNTEP03(Object shell, Map params, Map programResults){
    shell.with {
        if (checkValidProgramResults(programResults)) {
            return execStep("DSNTEP03", "DSNTEP4", programResults, {
                    mpr
                        .withFileConfigurations(new FileConfigurationUtils()
                        .withJobContext(jobContext)
                        .fileSystem("SYSPRINT")
                        .path("output(out012.txt)").recordSize(1152)
                        .disposition("SHR")
                        .build()
                        .fileSystem("SYSIN")
                        .stream(
"""  SELECT * FROM BUR000.AP_JBI7_INVOICE WITH UR;
  """, getEncoding())
                        .build()
                        .getFileConfigurations(fcmap))
                        .withParameters(params)
                    .runProgram("DSNTEP4")
                })
        }
    }
}
```

请注意，SQL 查询是 “按原样” 提供给 DSNTEP2 /4 实用程序的，不做任何修改。

DSNUTILB

DSNUTILB 目的

用于加载、复制和管理数据的数据库实用程序。通常在旧版 DB2 数据库上运行；现代化版本支持 AWS Aurora 和其他目标数据库。

DSNUTILB 签名

从本质上讲，DSNUTILB本来就是要用现代化的作业脚本来调用的。

不带任何参数；从`SYSIN`数据集中读取命令。

支持的命令有：

- `TEMPLATE`（数据集的动态分配）
- `LISTDEF`（将数据库对象分组为列表，可供其他命令使用）
- `COPY`（创建数据库对象的副本）
- `LOAD`（将记录加载到表中）
- `DISCARD`（从表中删除记录）

有关命令的更多详细信息，请查阅相关的相应旧版文档。

DSNUTILB 相关的配置参数

该实用程序的行为受以下配置参数的影响：

- `unload.useDatabaseConfiguration`
- `load.format.localDate`
- `load.format.dbDate`
- `load.format.localTime`
- `load.format.dbTime`
- `load.sqlCodePointShift`
- `convertGraphicDataToFullWidth`

有关配置这些参数[the section called “可选 Web 应用程序的可用属性”](#)的详细信息，请参阅。

DSNUTILB 检查/错误处理

- 如果SYSIN数据集不包含任何可用的命令，则RuntimeException会抛出 a。
- 如果在数据库操作期间发生任何异常，则会记录一条错误消息，返回码将设置为 8，然后抛出 aStopRunUnitException（停止当前的运行单元）。

DSNUTILB 用法示例

以下是 JCL 脚本中使用 DSNUTILB 的示例：

```
//*********************************************************************
//* LOAD DATA IN TABLE AP_JBI7_INVOICE.                             *
//*********************************************************************
//DSN01  EXEC PGM=DSNUTILB,DYNAMNBR=20
//SYSREC DD  DSN=input(input021.data),
//           DISP=SHR
//           DCB=(RECFM=FB,LRECL=76)
//SYSIN  DD  DSN=input(dsn01.card),
//           DISP=SHR
```

使用命令卡片内容 (dsn01.card)--用于将数据从旧平台导出的平面文件加载到数据库中--：

```
  LOAD         DATA
               INDDN         SYSREC
               RESUME        NO
               LOG           YES
               NOCOPYPEND
               SORTDEVT      SYSDA
               SORTNUM       12
               SORTKEYS      100000
               DISCARDS      0
               INTO TABLE    BUR000.AP_JB17_INVOICE
               WHEN (76:76) = 'L'
 ( IDENTIFIER        POSITION(1:1)        SMALLINT
  ,CUST_ID           POSITION(10)         VARCHAR
                                          NULLIF(39) = '?'
  ,CUST_KD           POSITION(40:43)      CHAR
  ,INVC_AMNT         POSITION(44:49)      NUMERIC
  ,INVC_DAT          POSITION(50:75)      TIMESTAMP EXTERNAL(26)
 )
```

以及配套的 groovy 现代化脚本片段：

```
// STEP DSN01 - PGM - DSNUTILB****************************************************
def stepDSN01(Object shell, Map params, Map programResults){
    shell.with {
        if (checkValidProgramResults(programResults)) {
            return execStep("DSN01", "DSNUTILB", programResults, {
                mpr
                    .withFileConfigurations(new FileConfigurationUtils()
                        .withJobContext(jobContext)
                        .fileSystem("SYSREC")
                        .path("input(input021.data)").recordSize(76)
                        .disposition("SHR")
                        .build()
                        .fileSystem("SYSIN")
                        .path("input(dsn01.card)")
                        .disposition("SHR")
                        .build()
                        .getFileConfigurations(fcmap))
                    .withParameters(params)
                    .runProgram("DSNUTILB")
                    })
            }
    }
}
```

INFUTILB /INZUTILB

INFUTILB 目的

INFUTILB/INZUTILB 是用于从数据库中提取数据——卸载——（DB2 在传统环境中）并将其转换为各种输出格式的实用程序。

系统会自动调整传统的 SQL 查询 on-the-fly以满足现代目标数据库的需求（支持的引擎：PostgreSQL、Oracle 和）。 DB2

INFUTILB /INZUTILB Signature

可以使用以下程序别名（并与相应的旧版排序实用程序名称相匹配）：

- `INFUTILB`
- `INZUTILB`

该实用程序不接受任何参数，而是从SYSIN数据集（“控制卡”）中读取要运行的命令。从数据库中提取的记录将写入SYSREC数据SYSPUNCH集，可选的数据集用于存储可用于重新加载数据的控制卡（the section called “DSNUTILB”例如，使用其他实用程序）。

从本质 INFUTILB/INZUTILB 上讲，主要由现代化的作业脚本调用。

要获取有关支持的命令的详细信息，请参阅相应的旧版文档。 INFUTILB/INZUTILB 他们正在 “按原样” 使用传统的 “控制卡” 数据集。

INFUTILB /INZUTILB 相关的配置参数

该实用程序的行为受以下配置参数的影响：

- `unload.sqlCodePointShift`
- `unload.noPad`
- `unload.nbi.whenNull`
- `unload.nbi.whenNotNull`
- `unload.useDatabaseConfiguration`
- `unload.format.date`
- `unload.format.time`
- `unload.format.timestamp`
- `unload.columnFiller`
- `unload.varCharIsNull`
- `unload.DFSIGDCB`
- `hasGraphic`
- `forcedDate`
- `frozenDate`

有关配置这些参数the section called “可选 Web 应用程序的可用属性”的详细信息，请参阅。

INFUTILB/INZUTILB 检查/错误处理

- 如果目标数据库不是支持的数据库引擎（PostgreSQL、Oracle DB2 和）的一部分，则程序返回码将设置为 8 并`UnsupportedOperationException`抛出。
- 如果程序未能删除临时文件，则返回码将设置为 4，将记录一条错误消息，但程序运行不会中断。
- 对于以下所有情况，程序返回码将设置为 4 或 8，并抛出（停止程序运行）：`AbendException`

- 如果SYSREC数据集不是支持的类型（GDG 或基于文件系统的数据集）；返回代码 4；
- 如果SYSPUNCH数据集不是支持的类型（GDG 或基于文件系统的数据集或DUMMY）；返回代码 4；
- 如果程序无法检索SYSREC数据集记录大小（数据集目录中未设置或未定义）；返回代码 8；
- 如果程序无法检索SYSPUNCH数据集记录大小（数据集目录中未设置或未定义）；返回代码 8；
- 如果用于创建SYSREC数据集内容的查询无效（错误的查询将被记录下来）；返回代码 4；
- 如果在从数据库中获取数据时出现任何异常；返回代码 8；
- 如果SYSIN数据集中缺少该OUTDDN命令，则执行卸载任务；返回代码 8；
- 如果在SYSIN数据集中找不到有效的命令，则返回代码 8；

INFUTILB /INZUTILB 用法示例

以下是旧版 jcl 脚本片段示例：

```
//*********************************************************************
//* UNLOAD DATA FROM TABLE AP_JBI7_INVOICE.                        *
//*********************************************************************
//INF1   EXEC PGM=INFUTILB
//SYSREC DD  DSN=output(out032.data),
//           DISP=SHR
//           DCB=(RECFM=FB,LRECL=90)
//SYSIN  DD  DSN=input(inf12.card),
//           DISP=SHR
```

它使用以下命令卡 (inf12.card) 从数据库中卸载一些数据（在这里，记录是根据其日期选择的）：

```
  UNLOAD
      SELECT * FROM BUR000.AP_JB17_INVOICE
      WHERE INVC_DAT >=
      CONCAT(STRIP(CHAR(YEAR(CURRENT DATE - 100 YEAR))),'-01-01-00.00.00.000000')
      AND INVC_DAT >=
      CONCAT('2025-01-01-',CONCAT((CURRENT TIME),'.000000'))
      AND INVC_DAT >= (CURRENT TIMESTAMP - 100 YEAR)
      ORDER BY identifier ASC
        OUTDDN (SYSREC)
        FORMAT DSNTIAUL
```

以及匹配的 groovy 脚本片段，这是 jcl 自动现代化的结果：

```
// STEP INF1 - PGM - INFUTILB*****************************************************
def stepINF1(Object shell, Map params, Map programResults){
    shell.with {
        if (checkValidProgramResults(programResults)) {
            return execStep("INF1", "INFUTILB", programResults, {
                mpr
                    .withFileConfigurations(new FileConfigurationUtils()
                        .withJobContext(jobContext)
                        .fileSystem("SYSREC")
                        .path("output(out032.data)").recordSize(90)
                        .disposition("SHR")
                        .build()
                        .fileSystem("SYSIN")
                        .path("input(inf12.card)")
                        .disposition("SHR")
                        .build()
                        .getFileConfigurations(fcmap))
                    .withParameters(params)
                    .runProgram("INFUTILB")
                })
        }
    }
}
```

JXHDBCLR

JXHDBCLR 目的

JXHDBCLR 是一个数据库清除实用程序，可在 GS21 平台上找到，它根据命令卡中提供的特定语句截断表。

JXHDBCLR 签名

它不接受任何参数，而是从SYSIN数据集（命令卡）中读取语句。

由于其性质，它主要被现代化的作业脚本调用。

JXHDBCLR 相关的配置参数

该实用程序的行为受以下配置参数的影响：

- `unload.sqlCodePointShift`

- `unload.noPad`
- `unload.nbi.whenNull`
- `unload.nbi.whenNotNull`
- `unload.useDatabaseConfiguration`
- `unload.format.date`
- `unload.format.time`
- `unload.format.timestamp`
- `hasGraphic`
- `forcedDate`
- `frozenDate`

有关配置这些参数the section called “可选 Web 应用程序的可用属性”的详细信息，请参阅。

JXHDBCLR 检查/错误处理

如果找不到要截断的表，则会记录一条警告消息，但程序运行不会中断。

如果在表截断期间出现一些故障，则程序返回码将设置为 4，将记录错误消息，但程序运行不会中断。

对于以下任何一种`AbendException`情况，程序返回码将设置为 8 并抛出（停止程序运行）：

- 如果命令卡片内容为空；
- 在解析命令处理过程中是否出现任何异常；

JXHDBCLR 用法示例

带有内联命令卡的 JXHDBCLR 用法示例：

```
//*******************************************************************
//**  Step 1 - JXHDBCLR UTILITY - DBCLEAR TYPE=2/3
//*******************************************************************
//STEP01  EXEC PGM=JXHDBCLR,REGION=256K,PARM='LINECNT=0'
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
    DBCLEAR    SCHEMA=MUSICSCH,TYPE=3,CHECK=YES
    DEFINE     RANGE=(SINGERDEST)
END
```

以及匹配的现代化 groovy 脚本片段：

```
// STEP STEP01 - PGM - JXHDBCLR***************************************************
def stepSTEP01(Object shell, Map params, Map programResults){
 shell.with {
        if (checkValidProgramResults(programResults)) {
            return execStep("STEP01", "JXHDBCLR", programResults, {
                mpr
                    .withFileConfigurations(new FileConfigurationUtils()
                        .withJobContext(jobContext)
                        .systemOut("SYSPRINT")
                        .output("*")
                        .build()
                        .fileSystem("SYSIN")
                        .stream(
"""    DBCLEAR    SCHEMA=MUSICSCH,TYPE=3,CHECK=YES
    DEFINE     RANGE=(SINGER)
END""", getEncoding())
                        .build()
                        .getFileConfigurations())
                    .withArguments(getParm("LINECNT=0"))
                    .withParameters(params)
                    .runProgram("JXHDBCLR")
                })
        }
    }
}
```

旧版的内联卡片使用直播保持 “原样”。

命令实用程序

本节与实用程序有关，其作用是处理使用控制卡提供的用户命令。

IKJEFT1A/IKJEFT1B/KEQEFT01/IKJEFT01/DSNDBTCH

IKJEFT1一个目的

IKJEFT1A 及其别名在批处理作业中执行 TSO（T ime S haring O 选项）命令，无需交互式 TSO 会话，桥接非交互式批处理和交互式 TSO 环境。

模仿传统的 IKJEFT1 A 行为，但存在特定于环境的差异：例如，在现代环境中，T DB2 ERM (inate) 命令会被忽略（记录为信息）。

> **Note**
>
> 此处未详细介绍命令语法- IKJEFT1 A 使用未更改的旧命令数据集。有关 TSO 命令的详细信息，请参阅旧版平台文档。

IKJEFT1一个签名

使用SYSTSIN数据集进行命令输入（通过 JCL DD 指令定义）。现代化版本使用相同的数据集，保持不变。仅在现代化的 Groovy 作业脚本中调用。

IKJEFT1A 相关的配置参数

该实用程序的行为受以下配置参数的影响：

- `systin.encoding`

有关配置此参数the section called “可选 Web 应用程序的可用属性”的详细信息，请参阅。

IKJEFT1A. Checks/Error 处理

当 SYSTSIN 数据集中存在不支持的命令时，就会抛出 a `RuntimeException`。

IKJEFT1A 示例用法

旧版 JCL 脚本（使用SYSTSIN数据集的内联内容，使用`DD *`指令）

```
//*********************************************************************
//* READ THE TEMPORARY INPUT FILE                                     *
//*********************************************************************
// IF IDCM00032.RC = 1 THEN
//067FILEKEY  EXEC PGM=IKJEFT01,DYNAMNBR=20
//SYSTSIN   DD *
 DSN SYSTEM(DB2P)
 RUN  PROGRAM(067-fileKey) PLAN(FILEKEYPLAN)    PARM('RT')
 END
/*
//TEMPVSAM  DD DSN=IDXVIDEO.TEMPVSAM,DISP=SHR
//FKOUT     DD DSN=output(out067.txt),DISP=(NEW,CATLG)
// ENDIF
//*********************************************************************
```

还有匹配的现代化脚本（在 groovy 中）—— SYSTSIN 数据集的内联内容由 “流” 表示 —

```
// STEP 067FILEKEY - PGM - IKJEFT01*********************************************
def step067FILEKEY(Object shell, Map params, Map programResults){
    shell.with {
        if (checkValidProgramResults(programResults)) {
            return execStep("067FILEKEY", "IKJEFT01", programResults, {
                mpr
                    .withFileConfigurations(new FileConfigurationUtils()
                        .withJobContext(jobContext)
                        .fileSystem("SYSTSIN")
                        .stream(
""" DSN SYSTEM(DB2P)
 RUN  PROGRAM(067-fileKey) PLAN(FILEKEYPLAN)    PARM('RT')
 END
""", getEncoding())
                        .build()
                        .bluesam("TEMPVSAM")
                        .dataset("IDXVIDEO.TEMPVSAM")
                        .disposition("SHR")
                        .build()
                        .fileSystem("FKOUT")
                        .path("output(out067.txt)")
                        .disposition("NEW")
                        .normalTermination("CATLG")
                        .build()
                        .getFileConfigurations())
                    .withParameters(params)
                    .runProgram("IKJEFT01")
                })
        }
    }
}
```

QCMDEXC

QCMDEXC 目的

该实用程序模拟 AS/400 系统实用程序 QCMDEXC 的行为，用于在运行时以动态方式运行系统命令。

功能：

- 动态解析和执行命令。

- 支持命名和位置参数格式。

QCMDEXC 签名

接受 1-2 个参数：

- 第一：包含要执行的命令的字母数字数据
- 第二（可选）：从第一个参数读取的长度（以字节为单位）（默认为全长）

QCMDEXC 处理 Checks/Error

对于以下情况，`RuntimeException`将抛出 a：

- 当提供的参数数量不是一两个时；
- 如果 commands 参数为空或者要运行的命令无法识别；
- 如果命令由于任何原因运行失败；此外，程序的返回代码将设置为 1。

QCMDEXC 示例用法

以下是旧版 COBOL 程序的用法示例：数据项`CL-COMMANDX`在用作 QCMDEXC 程序调用的参数之前先装有命令：

```
        INITIALIZE CL-COMMANDX.

        STRING
            "CPYF FROMFILE(PLAYERS) TOFILE(" DELIMITED BY SIZE,
            "NEWFILE"                      DELIMITED BY SIZE,
            ") MBROPT(*REPLACE) FMTOPT(*NOCHK) CRTFILE(*YES)"
                                           DELIMITED BY SIZE
            INTO CL-COMMANDX
        END-STRING.

        CALL "QCMDEXC"                  USING CL-COMMANDX
                                              CLENGTHX.
```

一旦现代化为 Java 代码，它就会变成：

```
   DataUtils.initialize(ctx.getClCommandx().getClCommandxReference());

StringConcatenationBuilder.newInstance(ctx.getClCommandx().getClCommandxReference())
```

```
            .addDelimitedBySize("CPYF FROMFILE(PLAYERS) TOFILE(")
            .addDelimitedBySize("NEWFILE")
            .addDelimitedBySize(") MBROPT(*REPLACE) FMTOPT(*NOCHK) CRTFILE(*YES)")
            .end();
        ctrl.callSubProgram("QCMDEXC", CallBuilder.newInstance()
            .byReference(ctx.getClCommandx().getClCommandxReference())
            .byReference(ctx.getClengthx().getClengthxReference())
            .getArguments(), ctx);
```

请注意，旧版命令内容在现代化代码中 “按原样” 使用，未经任何修改。

排序实用工具

排序实用程序依赖于其引用对实用程序具有特殊含义的数据集。因此，现代化方面的所有排序实用程序在groovy脚本中都具有相同的调用模式：

```
mpr.withFileConfigurations(<FILE CONFIGURATIONS DETAILS>...)
   .withParameters(params)
   .runProgram("<SORT UTILITY ALIAS>")
```

其中mpr是MainProgramRunner实例（请参阅本页中的 “调用程序的脚本” 部分）。以下各节给出了排序实用程序的别名。

JCL 脚本中的DD（数据集定义）条目被现代化为文件配置详细信息中的条目，这些条目充当该方法的mpr.withFileConfigurations()参数。有关该概念的插图，请参阅下面的示例。

SORT/SYNCSORT/ICEMAN

用途

该程序根据提供的标准模拟各种大型机排序实用程序，这些实用程序用于数据集中的sort/merge/copy数据。可以使用以下程序别名（并与相应的旧版排序实用程序名称相匹配）：

- SORT
- SYNCSORT
- ICEMAN

此处未提供有关控制卡中的 SORT/MERGE 指令和传统排序实用程序功能的详细信息，但应从现有的相关遗留平台文档中获取。

签名

该程序不接受任何参数，而是依赖特定的数据集引用：

- SYSIN数据集（又名 “控制卡”）保存控制语句 sort/merge
- 可选SYMNAMES数据集在 SYSIN 内容中保存变量替换指令
- 可选的SORTXDUP或SORTXSUM数据集可用于存储重复的记录
- 以要处理的记录为前缀SORTIN或SORTDBIN保存要处理的数据集（输入）
- SORTOUT数据集保存程序的结果（输出）
- 在某些旧版作业脚本中找到的 SORT WORK 数据集的SORTWK定义会被忽略（并且不在现代化调用中表示）；在现代环境中，排序工作数据集将始终动态分配
- DD 以SORTJN（前缀）开头的两个数据集包含连接键指令（用于在排序过程中连接数据集）所涉及的记录

例如，考虑以下联接键指令：

```
JOINKEYS FILE=F1,FIELDS=(13,5,A)
JOINKEYS FILE=F2,FIELDS=(24,5,A)
```

在这里，连接键的长度为 5，起始于：

- 数据集中记录的位置 13SORTJNF1（SORTJN前缀和文件的串联）F1
- 数据集中记录的位置 24SORTJNF2（SORTJN前缀和文件的串联）F2

检查/错误处理

- 如果输入文件 (SORTIN) 有SHR处置但无法找到，则会记录一条错误消息，程序返回码设置为 1，程序运行暂停（不会进行排序，也不会生成任何输出）

对于以下情况，将抛出RuntimeException保留专用消息：

- 如果程序调用需要连接到数据库（使用SORTDBIN数据集时，不是SORTIN）但找不到有效的数据源
- 如果输出文件 (SORTOUT) 定义不正确
- 如果在控制卡中找到的命令无法理解或不受支持
- 如果不完全是，则为一个SORT JOINKEYS案例提供了两个输入文件

示例用法

合并示例

以下是来自作业脚本ICEMAN片段的调用示例：

控制卡是内联的，命令用于合并输入文件中的字段（参见SYSIN条目）

```
//*
//PASOSO03 EXEC PGM=ICEMAN,REGION=0M
//SORTIN01 DD DSN=input(input809a.data),DISP=SHR,LRECL=10
//SORTIN02 DD DSN=input(input809b.data),DISP=SHR,LRECL=10
//SORTOUT  DD DSN=output(out809.txt),DISP=(,PASS),LRECL=10
//SORTWK01 DD SPACE=(281,(156300,156300),RLSE),AVGREC=U
//SORTWK02 DD SPACE=(281,(156300,156300),RLSE),AVGREC=U
//SYSIN    DD *
   MERGE  FIELDS=(1,6,PD,A,7,2,CH,A)
   END
/*
```

还有匹配的现代化 groovy 脚本片段——请注意，如前所述，在现代化过程中不会考虑这些SORTWK条目，而且内联控制卡与旧版控制卡内容完全匹配。

```
// STEP PASOSO03 - PGM - ICEMAN***************************************************
def stepPASOSO03(Object shell, Map params, Map programResults){
    shell.with {
        if (checkValidProgramResults(programResults)) {
            return execStep("PASOSO03", "ICEMAN", programResults, {
                mpr
                    .withFileConfigurations(new FileConfigurationUtils()
                        .withJobContext(jobContext)
                        .fileSystem("SORTIN01")
                        .path("input(input809a.data)").recordSize(10)
                        .disposition("SHR")
                        .build()
                        .fileSystem("SORTIN02")
                        .path("input(input809b.data)").recordSize(10)
                        .disposition("SHR")
                        .build()
                        .fileSystem("SORTOUT")
                        .path("output(out809.txt)").recordSize(10)
                        .normalTermination("PASS")
                        .build()
```

```
                        .fileSystem("SYSIN")
                        .stream(
"""   MERGE  FIELDS=(1,6,PD,A,7,2,CH,A)
   END                                                                              """,
 getEncoding())
                        .build()
                        .getFileConfigurations())
                        .withParameters(params)
                    .runProgram("ICEMAN")
                })
        }
    }
}
```

简单排序示例

一个带有内联控制卡的简单传统排序步骤（作业脚本片段），从 carddemo 示例应用程序中获取：

```
//*********************************************************************
//* CREATE COPY OF TRANSACT FILE WITH CARD NUMBER AND TRAN ID AS KEY
//*********************************************************************
//STEP010  EXEC PGM=SORT
//SORTIN   DD  DISP=SHR,DSN=AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SORTOUT  DD  DSN=AWS.M2.CARDDEMO.TRXFL.SEQ,
//             DISP=(NEW,CATLG,DELETE),UNIT=SYSDA,
//             DCB=(LRECL=350,BLKSIZE=3500,RECFM=FB),
//             SPACE=(CYL,(1,1),RLSE)
//SYSIN    DD *
  SORT FIELDS=(263,16,CH,A,1,16,CH,A)
  OUTREC FIELDS=(1:263,16,17:1,262,279:279,50)
/*
```

以及匹配的现代化 groovy 脚本片段：

```
// STEP STEP010 - PGM - SORT******************************************************
def stepSTEP010(Object shell, Map params, Map programResults){
    shell.with {
        if (checkValidProgramResults(programResults)) {
            return execStep("STEP010", "SORT", programResults, {
                mpr
                    .withFileConfigurations(new FileConfigurationUtils()
```

```
                        .withJobContext(jobContext)
                        .bluesam("SORTIN")
                        .dataset("AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS")
                        .disposition("SHR")
                        .build()
                        .systemOut("SYSPRINT")
                        .output("*")
                        .build()
                        .systemOut("SYSOUT")
                        .output("*")
                        .build()
                        .fileSystem("SORTOUT")
                        .path("AWS.M2.CARDDEMO.TRXFL.SEQ").recordSize(350)
                        .disposition("NEW")
                        .normalTermination("CATLG")
                        .abnormalTermination("DELETE")
                        .build()
                        .fileSystem("SYSIN")
                        .stream(
"""  SORT FIELDS=(263,16,CH,A,1,16,CH,A)
  OUTREC FIELDS=(1:263,16,17:1,262,279:279,50)""", getEncoding())
                        .build()
                        .getFileConfigurations())
                    .withParameters(params)
                    .runProgram("SORT")
                })
        }
    }
}
```

请注意，内联控制卡按 “原样” 使用，未对旧版控制卡内容进行任何修改。

ICETOOL

用途

ICETOOL 实用程序用于在单个作业步骤中对数据集执行多项操作（数据操作、排序和分析）。

支持以下核心运算符：

- COPY-将数据从输入文件复制到输出文件
- SORT-使用指定的排序卡/标准对数据进行排序
- SELECT-根据条件筛选和选择特定记录

- SPLICE-来自多个来源 Merges/joins 的数据
- COUNT-计算符合指定标准的记录
- OCCUR-分析数据中的出现模式

对于SPLICE操作员，该实用程序将使用基于数据分块策略的多线程方法来确保优化的性能。

有关运算符的详细信息应从相应的旧平台文档中获取。

签名

该ICETOOL实用程序不采用任何参数，但依赖于特定的数据集：

- TOOLIN数据集包含要由实用程序处理的控制语句
- TOOLMSG而且现代化ICETOOL实用程序暂时不使用DFSMSG数据集（已忽略）
- IN是输入数据集（要处理的记录）的前缀
- OUT是输出数据集（处理产生的记录）的前缀
- 控制卡中的控制语句可能会引用其他数据集

检查/错误处理

对于以下情况，RuntimeException将抛出带有相关消息的 a：

- 如果不支持其中一个控制语句中使用的运算符
- 对于任何运算符，如果提供了不支持的指令

示例用法

ICETOOL 排序示例

以下是使用 ICETOOL 进行排序的传统 jcl 示例：

- 每个 SORT 运算符控制语句都使用专用的控制卡，其引用通过 USING 关键字指定
- 所有控制卡都是在定义之后定义的，并且是内联的（参见SEL1CNTL下面的*CNTL条目）TOOLIN

```
//SAMPL052 EXEC PGM=ICETOOL,REGION=1024K
//TOOLMSG  DD SYSOUT=*
//DFSMSG   DD SYSOUT=*
```

```
//IN1      DD DSN=input(input846a.data),DISP=SHR
//            DCB=(RECFM=F,LRECL=8)
//IN2      DD DSN=input(input846b.data),DISP=SHR
//            DCB=(RECFM=F,LRECL=8)
//OUT1     DD DSN=output(out846a.txt),DISP=(,CATLG)
//            DCB=(RECFM=F,LRECL=8)
//OUT2     DD DSN=output(out846b.txt),DISP=(,CATLG)
//            DCB=(RECFM=V)
//OUT3     DD DSN=output(out846c.txt),DISP=(,CATLG)
//            DCB=(RECFM=V)
//TOOLIN   DD *
  SORT FROM(IN1) TO(OUT1) USING(SEL1)
  SORT FROM(IN2) TO(OUT1) USING(SEL2)
  SORT FROM(IN1) TO(OUT2) USING(SEL3)
  SORT FROM(IN2) TO(OUT2) USING(SEL4)
  SORT FROM(IN1) TO(OUT3) USING(SEL5)
  SORT FROM(IN2) TO(OUT3) USING(SEL6)
/*
//SEL1CNTL DD *
    OPTION COPY
    OUTFIL BUILD=(1,8,JFY=(SHIFT=LEFT,TRAIL=C'*'))
//*
//SEL2CNTL DD *
   OPTION COPY,SKIPREC=1
   OUTFIL BUILD=(1,8,JFY=(SHIFT=LEFT,TRAIL=C'*'))
//*
//SEL3CNTL DD *
    OPTION COPY
    OUTFIL BUILD=(1,8,JFY=(SHIFT=LEFT,TRAIL=C'*')),
    FTOV,VLTRIM=C' '
//*
//SEL4CNTL DD *
    OPTION COPY,SKIPREC=1
    OUTFIL BUILD=(1,8,JFY=(SHIFT=LEFT,TRAIL=C'*')),
    FTOV,VLTRIM=C' '
//*
//SEL5CNTL DD *
    OPTION COPY
    OUTFIL BUILD=(1,8,JFY=(SHIFT=LEFT,TRAIL=C'*')),
    FTOV,VLTRIM=C' '
//*
//SEL6CNTL DD *
    OPTION COPY,SKIPREC=1
    OUTFIL BUILD=(1,7,JFY=(SHIFT=LEFT,TRAIL=C'*')),
```

```
    FTOV,VLTRIM=C' '
//*
```

现代化后，匹配的 groovy 脚本片段如下所示：

```
// STEP SAMPL052 - PGM - ICETOOL*****************************************************
def stepSAMPL052(Object shell, Map params, Map programResults){
    shell.with {
        if (checkValidProgramResults(programResults)) {
            return execStep("SAMPL052", "ICETOOL", programResults, {
                mpr
                    .withFileConfigurations(new FileConfigurationUtils()
                        .withJobContext(jobContext)
                        .systemOut("TOOLMSG")
                        .output("*")
                        .build()
                        .systemOut("DFSMSG")
                        .output("*")
                        .build()
                        .fileSystem("IN1")
                        .path("input(input846a.data)").recordSize(8)
                        .disposition("SHR")
                        .build()
                        .fileSystem("IN2")
                        .path("input(input846b.data)").recordSize(8)
                        .disposition("SHR")
                        .build()
                        .fileSystem("OUT1")
                        .path("output(out846a.txt)").recordSize(8)
                        .normalTermination("CATLG")
                        .build()
                        .fileSystem("OUT2")
                        .path("output(out846b.txt)").rdw(true)
                        .normalTermination("CATLG")
                        .build()
                        .fileSystem("OUT3")
                        .path("output(out846c.txt)").rdw(true)
                        .normalTermination("CATLG")
                        .build()
                        .fileSystem("TOOLIN")
                        .stream(
"""  SORT FROM(IN1) TO(OUT1) USING(SEL1)
  SORT FROM(IN2) TO(OUT1) USING(SEL2)
```

```
  SORT FROM(IN1) TO(OUT2) USING(SEL3)
  SORT FROM(IN2) TO(OUT2) USING(SEL4)
  SORT FROM(IN1) TO(OUT3) USING(SEL5)
  SORT FROM(IN2) TO(OUT3) USING(SEL6)               """, getEncoding())
                         .build()
                         .fileSystem("SEL1CNTL")
                         .stream(
"""    OPTION COPY
    OUTFIL BUILD=(1,8,JFY=(SHIFT=LEFT,TRAIL=C'*')) """, getEncoding())
                         .build()
                         .fileSystem("SEL2CNTL")
                         .stream(
"""   OPTION COPY,SKIPREC=1
   OUTFIL BUILD=(1,8,JFY=(SHIFT=LEFT,TRAIL=C'*'))""", getEncoding())
                         .build()
                         .fileSystem("SEL3CNTL")
                         .stream(
"""    OPTION COPY
    OUTFIL BUILD=(1,8,JFY=(SHIFT=LEFT,TRAIL=C'*')),
    FTOV,VLTRIM=C' '""", getEncoding())
                         .build()
                         .fileSystem("SEL4CNTL")
                         .stream(
"""    OPTION COPY,SKIPREC=1
    OUTFIL BUILD=(1,8,JFY=(SHIFT=LEFT,TRAIL=C'*')),
    FTOV,VLTRIM=C' '""", getEncoding())
                         .build()
                         .fileSystem("SEL5CNTL")
                         .stream(
"""    OPTION COPY
    OUTFIL BUILD=(1,8,JFY=(SHIFT=LEFT,TRAIL=C'*')),
    FTOV,VLTRIM=C' '""", getEncoding())
                         .build()
                         .fileSystem("SEL6CNTL")
                         .stream(
"""    OPTION COPY,SKIPREC=1
    OUTFIL BUILD=(1,7,JFY=(SHIFT=LEFT,TRAIL=C'*')),
    FTOV,VLTRIM=C' '""", getEncoding())
                         .build()
                         .getFileConfigurations())
                     .withParameters(params)
                     .runProgram("ICETOOL")
                 })
         }
```

```
    }
}
```

备注：

- 内联控制卡 “按原样” 使用；从旧控制卡中没有任何转换
- TOOLMSG和DFSMSG在现代化版本中定义，但在运行时将被忽略
- 在旧版本和现代版本中，控制卡都使用CNTL后缀定义，但在数据集的指令中引用时不带后缀：例如SORT FROM(IN1) TO(OUT1) USING(SEL1)，在中，USING(SEL1)是指TOOLIN数据集定义SEL1CNTL

ICETOOL 复制样本

这是另一个使用 COPY 运算符的 ICETOOL 示例。在 TOOLIN jcl 脚本片段中已内联：

```
//SAMPLO51 EXEC PGM=ICETOOL,REGION=1024K
//TOOLMSG  DD SYSOUT=*
//DFSMSG   DD SYSOUT=*
//IN1      DD DSN=input(input831.data),DISP=SHR
//            DCB=(RECFM=F,LRECL=12)
//OUT1     DD DSN=output(out831a.txt),DISP=OLD
//            DCB=(RECFM=F,LRECL=12)
//OUT2     DD DSN=output(out831b.txt),DISP=OLD
//            DCB=(RECFM=F,LRECL=12)
//TOOLIN   DD *
  COPY FROM(IN1) TO(OUT1,OUT2) USING(SEL1)
/*
//SEL1CNTL DD *
  OPTION COPY
  OUTFIL INCLUDE=(7,2,CH,EQ,C'10')
//*
```

以下是匹配的现代化 groovy 脚本片段：

```
// STEP SAMPLO51 - PGM - ICETOOL*****************************************************
def stepSAMPLO51(Object shell, Map params, Map programResults){
    shell.with {
        if (checkValidProgramResults(programResults)) {
            return execStep("SAMPLO51", "ICETOOL", programResults, {
                mpr
                    .withFileConfigurations(new FileConfigurationUtils()
```

```
                        .withJobContext(jobContext)
                        .systemOut("TOOLMSG")
                        .output("*")
                        .build()
                        .systemOut("DFSMSG")
                        .output("*")
                        .build()
                        .fileSystem("IN1")
                        .path("input(input831.data)").recordSize(12)
                        .disposition("SHR")
                        .build()
                        .fileSystem("OUT1")
                        .path("output(out831a.txt)").recordSize(12)
                        .disposition("OLD")
                        .build()
                        .fileSystem("OUT2")
                        .path("output(out831b.txt)").recordSize(12)
                        .disposition("OLD")
                        .build()
                        .fileSystem("TOOLIN")
                        .stream(
"""  COPY FROM(IN1) TO(OUT1,OUT2) USING(SEL1)
  COPY FROM(IN1) TO(OUT3,OUT4)
  COPY FROM(IN1) TO(OUT4)
  COPY FROM(IN1) TO(OUT5,OUT6)                       """, getEncoding())
                        .build()
                        .fileSystem("SEL1CNTL")
                        .stream(
"""  OPTION COPY
  OUTFIL INCLUDE=(7,2,CH,EQ,C'10')""", getEncoding())
                        .build()
                        .getFileConfigurations())
                    .withParameters(params)
                    .runProgram("ICETOOL")
                })
        }
    }
}
```

MFSORT

用途

该实用程序旨在模仿 Micro Focus 环境中名为 MFSORT 的排序实用程序的行为（它通常从命令行或旧版环境中的脚本中调用）。

在内部，该程序将实际的排序操作委托给the section called “SORT/SYNCSORT/ICEMAN”实用程序。

签名

仅支持以下传统语法：`mfsort take <control card>`

不支持直接指令调用。`mfsort <instructions>`

它不接受任何参数；take 指令使用引用为的数据集进行模拟*TAKE*，该数据集包含 MFSORT 要操作的命令。

检查/错误处理

- 如果`TAKE`数据集缺失或无效，则`RuntimeException`会抛出
- 考虑到从 MFSORT 到 SORT 的授权，这里也the section called “检查/错误处理”适用

示例用法

以下命令调用显示了 MFSORT 的用法示例：

```
mfsort take TESTSRT1.CTL
```

以下是匹配的现代化改编版 groovy 脚本片段：

```
mpr.withFileConfigurations(new FileConfigurationUtils()
.fileSystem("TAKE")
.path("input(TESTSRT1.CTL)")
.build()
.getFileConfigurations())
.withArguments("input") // relative path for use and give files
.runProgram("MFSORT");
```

其他/杂项实用工具

本节涵盖了各种公用事业计划，这些计划具有其他用途，无法附加到现有类别中

在以下段落中，PICTURE应理解为 COBOL PICTURE 关键字，用于描述字段的数据类型和格式。

CBL_ _NOT AND/CBL_OR/CBL_XOR/CBL_EQ/CBL_IMP/CBL

用途

这与对数据项的按位逻辑运算的支持有关，这种运算存在于某些 COBOL 方言（Micro Focus）中。请注意，使用 AWS Transform 自动重构可以处理按位运算符用法从 Cobol MF 方言到 java 的转换（参见下面的示例用法）。为了使现代化的代码正常运行，必须同时部署实用程序应用程序。

支持的运算符：

- 一元运算符：

一元运算符	details
CBL_NOT	按位补码（~target）

- 二进制运算符：

二元运算符	details
CBL_AND	按位 AND（源和目标）
CBL_OR	按位或（源 | 目标）
CBL_XOR	按位排他或（源 ^ 目标）
CBL_EQ	按位等价（~（源 ^ 目标））
CBL_IMP	按位含义（~source | target）

签名

除了只有一个操作数（目标）的 CBL_NOT 之外，所有其他操作都至少有两个操作数，即数据项。第一个操作数是源，第二个操作数是目标。另一个可选参数（length）给出了要处理的字节数（从左到

右）。如果省略长度，则默认为源和目标大小的最小值（以字节为单位）。操作结果存储在目标中。该程序的返回码为 0（除非发生异常）。

检查/错误处理

- 对于每个可用的运算符，都会检查所需参数的数量。如果未满足所需参数的最小数量，则`IllegalArgumentException`会抛出。
- 检查可选的长度整数参数是否为正数。如果提供的长度为负，则日志中将显示一条消息，并且不会应用运算符。

示例用法

- 一元运算符示例：

以下是 CBL_NOT 的 cobol 用法：

```
* TARGET:       00000101
* OPERATION:     NOT
* EXPECTED:     11111010 = 64000 dec (2nd byte unchanged - all 0)
     MOVE X'0500' to TARGET
     CALL "CBL_NOT" USING TARGET BY VALUE 1.
```

以及相应的 java 现代化：

```
/*
TARGET:       00000101
OPERATION:     NOT
EXPECTED:     11111010 = 64000 dec (2nd byte unchanged - all 0) */
ctx.getTarget().getTargetReference().setBytes(new byte[]{ 5, 0 });
ctrl.callSubProgram("CBL_NOT", CallBuilder.newInstance()
    .byReference(ctx.getTarget().getTargetReference())
    .byValue(1)
    .getArguments(), ctx);
```

- 二进制运算符示例：

例如，参见以下使用 CBL_AND 的 COBOL 代码：

```
* SOURCE:       00000011
```

```
  * OPERATION:     AND
  * TARGET:       00000101
  * EXPECTED:     00000001 = 256 dec (2nd byte unchanged - all 0)
       MOVE X'0300' to SRC
       MOVE X'0500' to TARGET
       CALL "CBL_AND" USING SRC TARGET BY VALUE 1.
```

以及它的 java 现代化：

```
/*
SOURCE:       00000011
OPERATION:     AND
TARGET:       00000101
EXPECTED:     00000001 = 256 dec (2nd byte unchanged - all 0) */
ctx.getSrc().getSrcReference().setBytes(new byte[]{ 3, 0 });
ctx.getTarget().getTargetReference().setBytes(new byte[]{ 5, 0 });
ctrl.callSubProgram("CBL_AND", CallBuilder.newInstance()
    .byReference(ctx.getSrc().getSrcReference())
    .byReference(ctx.getTarget().getTargetReference())
    .byValue(1)
    .getArguments(), ctx);
```

CEE3ABD

用途

该实用程序模仿同名旧程序的行为，其作用是终止带有Abend（异常结束）代码和可选的清理时机的程序。AWS 转换引擎可处理对 CEE3 ABD 的调用的自动现代化。为了使现代化的代码正常运行，必须同时部署实用程序应用程序。

该程序使用提供的中止代码停止当前运行的单元。一条信息消息将打印到日志中，指定已使用给定代码调用了用户退出。目前，计时参数没有被考虑在内，但出于现代化兼容性的考虑，它被当作一个论据。

签名

该程序接受 0 或 2 个参数。这两个论点是：

- Abend 代码（必须可解释为正整数值的数据项）
- 清理时间（必须可解释为正整数值的数据项）--忽略

如果未提供任何参数，则使用以下默认值：

- 晚间代码：0
- 清理时间：0

检查/错误处理

- 检查是否提供了 0 或 2 个参数，否则`IllegalArgumentException`将抛出。
- 如果提供了两个参数，请检查：abend 代码介于 0 和 4095 之间（均包括在内）；时序介于 0 和 5 之间（均包括在内）

这些测试的任何失败都将触发`IllegalArgumentException`.

示例用法

以下是来自 Carddemo 应用程序的 COBOL 程序中的示例用法：

```
  9999-ABEND-PROGRAM.
      DISPLAY 'ABENDING PROGRAM'
      MOVE 0 TO TIMING
      MOVE 999 TO ABCODE
      CALL 'CEE3ABD'.
```

以及相应的现代化 Java 代码：

```
public void _9999AbendProgram(final Cbtrn03cContext ctx, final ExecutionController
 ctrl) {
    DisplayUtils.display(ctx, ctrl, LOGGER, "ABENDING PROGRAM");
    ctx.getTiming().setTiming(0);
    ctx.getAbcode().setAbcode(999);
    ctrl.callSubProgram("CEE3ABD", CallBuilder.newInstance()
        .getArguments(), ctx);
}
```

割让

用途

CEEDATE 实用程序使用提供的格式将表示莉莲日期（自 1582 年 10 月 15 日以来的天数）的数字转换为基于字符的日期表示形式。

它模仿同名的旧系统实用程序的行为。

签名

考虑到其性质，CEEDATE 实用程序更适合从程序中调用。

它需要三个或四个参数（最后一个参数是可选的）：

- 第一个必填参数是一个数字数据项，其值将被解释为 Lilian 日期
- 第二个强制参数是一个字母数字数据项，其中包含用于将日期转换为字符的PICTURE字符串
- 必需的第三个参数是目标字母数字数据项，它使用第二个参数作为保存第一个参数的转换结果 PICTURE
- 第四个可选参数是用于存储来自实用程序的反馈代码的数据项

检查/错误处理

- 如果传递给实用程序的参数数量不是三或四个，则BluageWrapperException会抛出 a
- 如果提供的第一个数字参数无法正确计算为 Lilian 日期（越界），则会记录一条错误消息。可选的反馈代码持有人（如果有）将收到严重性为 3 的反馈代码和编号为 2512 的反馈代码
- 如果在日期转换期间由于提供的内容无效而发生异常PICTURE，则会记录一条错误消息。可选的反馈代码持有人（如果有）将收到严重性为 3 的反馈代码和编号为 2518 的反馈代码
- 如果由于任何原因无法正常进行转换，则输出数据项将用空白填充
- 如果转换成功，则将向可选的反馈代码持有者提供严重性为 0 的反馈代码（且没有消息）

示例用法

以下是来自 COBOL 程序的示例调用（反馈代码持有者 FC 的结构包含省略号，因为包含数百个未在此处显示的条件（级别 88）条目）：

```
WORKING-STORAGE SECTION.
01  LILIANS              PIC S9(9) BINARY.
01  TIMESTAMP-OUT               PIC X(80).
01  MASK.
    05  MASK-LEN         PIC S9(4) BINARY.
    05  MASK-STR.
        10  MASK-CHR     PIC X OCCURS 0 TO 256
                            DEPENDING ON MASK-LEN.
01  ROUTINE-NAMES.
    05 CEESECS-ROUTINE    PIC X(08) VALUE 'CEESECS '.
    05 CEELOCT-ROUTINE    PIC X(08) VALUE 'CEELOCT '.
```

```
        05 CEEDATE-ROUTINE    PIC X(08) VALUE 'CEEDATE '.
    01  FC.
    ...
    * lilian date for 4 June 1990
        MOVE SPACES TO MASK-STR
        MOVE 148887 TO LILIANS.
        MOVE 23                  TO MASK-LEN
        MOVE 'YYYY-MM-DD-HH:MI:SS.999' TO MASK-STR
        CALL CEEDATE-ROUTINE USING LILIANS
                                   MASK
                                   TIMESTAMP-OUT
                                   FC.
```

CEELOCT

用途

CEELOCT 实用程序用于以三种格式返回本地数据 date/time ：

- 莉莲日期（自 1582 年 10 月 14 日以来的天数）
- Lilian seconds（自 1582 年 10 月 14 日 00:00:00 以来的秒数）
- 格里高利字符串（格式`YYYYMMDDHHMISS999`为）

它模仿同名的旧系统实用程序的行为。

签名

考虑到其性质，CEELOCT 实用程序更适合从程序中调用。

它需要三个或四个参数（最后一个参数是可选的）：

- 第一个必填参数是一个数据项，用于存储 Lilian 日期
- 第二个强制参数是一个数据项，用于存储 Lilian 秒
- 必填的第三个参数是一个数据项，用于使用上面给出的形式存储公历日期
- 可选的第四个参数是用于存储来自实用程序的反馈代码的数据项

检查/错误处理

- 如果传递给实用程序的参数数量不是三或四个，则`BluageWrapperException`会抛出 a

- 如果在处理从本地 date/time 到任何输出格式的转换过程中出现任何异常：第一个和第二个参数将设置为 0，第三个参数将保持不变；将记录错误消息；（可选）将向反馈代码持有者提供严重性为 3 的反馈代码和消息编号 2531
- 成功后，所有三个参数都将填充正确的内容，可选的反馈代码将使用严重性为 0 的代码

示例用法

以下是 COBOL 示例片段，显示了 CEELOCT 实用程序的用法。反馈代码持有人结构 FC 没有完整给出，因为它包含数百个条件（88 级）条目。

```
WORKING-STORAGE SECTION.
01  LILIANS              PIC S9(9) BINARY.
01  GREGORN              PIC X(80).
01  SECONDS              COMP-2.
01  FC.
...
01  ROUTINE-NAMES.
    05 CEESECS-ROUTINE    PIC X(08) VALUE 'CEESECS '.
    05 CEELOCT-ROUTINE    PIC X(08) VALUE 'CEELOCT '.
    05 CEEDATE-ROUTINE    PIC X(08) VALUE 'CEEDATE '.
...

    CALL CEELOCT-ROUTINE USING LILIANS
                               SECONDS
                               GREGORN
                               FC.
```

CEERAN0

用途

调用 CEERAN0 程序使用指定的种子生成介于 0.0 和 1.0 之间的伪随机数。它基于乘法同余法算法，该算法需要用户指定的种子。使用 0 作为种子会触发一种特定的模式，在这种模式下，种子实际上是根据格林威治平均时间（程序运行时）计算的。否则，将按原样使用种子。伪随机序列是可预测的。

签名

CEERAN0 程序采用三个参数：

- 种子（输入参数），可以解释为正整数的数据项（包括 0）

- 随机数（输出参数），可以解释为双精度浮点数的数据项（其值将介于 0.0 和 1.0 之间，不包括）；它是程序的结果
- 可选的反馈码（输出参数），一个 12 字节的数据项，用于存储来自程序的有关随机数计算的反馈

检查/错误处理

- 如果参数的数量不是 2 或 3，则`IllegalArgumentException`会抛出
- 种子值必须介于 0 和 2147483646 之间（含）。如果种子值超出这些范围，则将记录一条错误消息，并将反馈代码设置为严重性 3，消息编号设置为 2524。生成的随机数将设置为 -1.0（与传统行为类似）
- 如果种子值设置为 0，但系统无法检索格林威治平均时间（出于任何原因），则将使用值 1 作为后备进行计算，反馈代码将设置为严重性 1，消息编号设置为 2523。随机数计算将继续使用后备种子值（类似于传统行为）

示例用法

这是一个 java 示例，演示了如何使用 CEERAN0 程序，使用包括反馈代码在内的所有参数，以及来自多个层（实体、服务）的位：

```
//Entity layer
public class Randomin extends RecordEntity {
    private final Group root = new Group(getData()).named("RANDOMIN");
    private final Elementary randomin = new Elementary(root,new BinaryIntegerType(4,
 true),new BigDecimal("0")).named("RANDOMIN");
    ...

public class Randomout extends RecordEntity {
    private final Group root = new Group(getData()).named("RANDOMOUT");
    private final Elementary randomout = new Elementary(root,new
 DoubleFloatingPointType(),new BigDecimal("0")).named("RANDOMOUT");
    ...
public class Returncode1 extends RecordEntity {
    private final Group root = new Group(getData()).named("RETURNCODE");
    private final Elementary returncode1 = new Elementary(root,new
 AlphanumericType(12)," ").named("RETURNCODE");
    ...

// Service layer
    CallHandler.newInstance(ctrl, ctx, ctx.getErrorContext())
        .byReference(ctx.getRandomin().getRandominReference(),
```

```
        ctx.getRandomout().getRandomoutReference(),
        ctx.getReturncode1().getReturncode1Reference())
        .call("CEERAN0");
```

CEESECS

用途

CEESECS 实用程序将时间戳字符串表示形式转换为 Lilian 秒（自 1582 年 10 月 14 日 00:00:00 以来的秒数）。

签名

考虑到其性质，CEESECS 实用程序更倾向于从程序中调用。

它需要三个或四个参数（最后一个参数是可选的）：

- 第一个必填参数是一个数据项，其值将被解释为时间戳
- 强制性的第二个参数是一个字母数字字数据项，其中包含用于指定如何解释第一个参数的`PICTURE`字符串
- 强制性的第三个参数是数据项，它保存了使用第二个参数作为第一个参数的转换结果 `PICTURE`
- 第四个可选参数是用于存储来自实用程序的反馈代码的数据项

检查/错误处理

- 如果传递给实用程序的参数数量不是三或四个，则`BluageWrapperException`会抛出 a
- 如果作为参数传递给实用程序的时间戳无效，则将记录一条错误消息，并且可以选择向反馈代码持有者提供严重性为 3 的反馈代码和编号为 2513 的反馈代码
- 如果作为参数传递给实用程序的图片无效，则将记录一条错误消息，并且可以选择向反馈代码持有者提供严重性为 3 的反馈代码和编号为 2518 的反馈代码
- 如果出于任何原因无法计算 lilian seconds 的输出，则第三个参数（输出）将设置为 0

示例用法

以下是 COBOL 程序中对 CEESECS 实用程序的调用示例：

```
WORKING-STORAGE SECTION.
01  SECONDS              COMP-2.
01  TIMESTAMP-IN.
```

```
        05  TIMESTAMP-IN-LEN        PIC S9(4) BINARY.
        05  TIMESTAMP-IN-STR.
            10  TIMESTAMP-IN-CHAR   PIC X OCCURS 0 TO 256
                                DEPENDING ON TIMESTAMP-IN-LEN.
    01  MASK.
        05  MASK-LEN         PIC S9(4) BINARY.
        05  MASK-STR.
            10  MASK-CHR     PIC X OCCURS 0 TO 256
                                DEPENDING ON MASK-LEN.
    01  FC.
    ...
    01  ROUTINE-NAMES.
        05 CEESECS-ROUTINE    PIC X(08) VALUE 'CEESECS '.
        05 CEELOCT-ROUTINE    PIC X(08) VALUE 'CEELOCT '.
        05 CEEDATE-ROUTINE    PIC X(08) VALUE 'CEEDATE '.
    ...
    ...
    * date for lilian second 12,799,191,601.123
        MOVE '1988-5-16-19:00:01.123' TO TIMESTAMP-IN-STR
        MOVE 23                  TO MASK-LEN
        MOVE 'YYYY-MM-DD-HH:MI:SS.999' TO MASK-STR
        CALL CEESECS-ROUTINE USING TIMESTAMP-IN
                                   MASK
                                   SECONDS
                                   FC.
```

ILBOABN0

用途

ILBOABN0 程序的目的是使用用户提供的终止（异常结束）代码，以受控的方式中断当前运行的单元。通常用于错误处理专用程序。

通过投掷 a 来中断当前运行的单元`StopRunUnitException`。

签名

ILBOABN0 程序采用一个强制性参数，该参数是一个包含中止代码的数据项（必须可以解释为整数）。

检查/错误处理

在抛出`StopRunUnitException`以中断当前运行单元运行时，程序会将返回码设置为作为第一个参数提供的值。此外，还将记录一条信息消息。

示例用法

以下是 ILBOABN0 程序的 Java 示例用法，该程序源于通过 AWS 转换对 COBOL 进行现代化改造：

```
   77  WS-ABND-CODE          COMP   PIC S9(4)      VALUE +1234.
...
...

   1970-ABNDIT.
       CALL 'ILBOABN0' USING WS-ABND-CODE.
```

以及匹配的 Java 现代化：

```
//Entity layer
private final Group root = new Group(getData());
private final Elementary wsAbndCode = new Elementary(root,new BinaryType(4, 0, "STD",
 false, false, true),Short.valueOf("1234"));
...

//Service layer
@Override
public void _1970Abndit(final MyPgmContext ctx, final ExecutionController ctrl) {
   ctrl.callSubProgram("ILBOABN0", CallBuilder.newInstance()
       .byReference(ctx.getWsAbndCode().getWsAbndCodeReference())
       .getArguments(), ctx);
```

设置 AWS Blu Age 运行时

本节介绍在 AWS 基础架构上设置 AWS Blu Age Runtime 的步骤。在为应用程序设置 AWS Blu Age Runtime 之前，请了解先决条件、区域和存储分区以及配置和管理运行时环境的 CloudWatch警报设置。

主题

- AWS 蓝光时代运行时必备条件
- 入门 AWS Blu Age 运行时
- AWS Blu Age Runtime 的基础架构设置要求
- AWS 蓝光时代运行时神器
- 在亚马逊上部署 AWS Blu Age 运行时 EC2

- 在亚马逊 ECS 和亚马逊 EKS 的容器上部署 AWS Blu Age Runtime
- 测试 PlanetsDemo 应用程序

AWS 蓝光时代运行时必备条件

AWS Blu Age Runtime 有多个the section called “AWS Blu Age 发行说明”发行版本可用。如果您正在进行现代化项目，可能需要运行时的增量版本用于实现和测试目的。要确定您的需求，请联系您的 AWS Blu Age 配送经理。

在开始使用 AWS Blu Age Runtime 之前，请执行以下操作：

- 请确保您有一个 AWS 帐户。
- 确保你有一个使用 Blu Age 重构的现代化应用程序。 AWS
- 选择一个 AWS 区域和一个 AWS Blu Age Runtime 支持的计算选项。
- 选择你要使用的 AWS Blu Age 运行时版本。
- 查看the section called “基础设施设置要求”并验证运行 AWS Blu Age 运行时所需的其他组件。

> **Note**
>
> 如果你想测试 AWS Blu Age Runtime 的功能，你可以使用演示应用程序`Planets Demo`，你可以从 PlanetsDemo-v1.zip 下载该应用程序。

入门 AWS Blu Age 运行时

> **Note**
>
> AWS Blu Age Runtime 访问权限已从基于门票的请求转变为通过 AW S Blu Insights 直接下载。我们引入了一种新的简化方式，可通过 AWS Blu Insights 中的Blu Age工具箱功能访问Blu Age产品。

首先，请创建一个AWS 支持案例，请求入职以访问 AWS Blu Age Runtime。在您的请求中包括您的 AWS 账户 ID、您要使用的 AWS 区域、计算选项以及 AWS Blu Age 运行时版本。如果您不确定需要哪个版本，请联系您的 AWS Blu Age 交付经理。如果您已经有了由 AWS 大型机现代化重构工具生成的应用程序的代码源，请将`gapwalk.version`标签值记入现代化代码`pom.xml`库中的文件中。

> Note
>
> AWS Blu Age Runtime 有两种主要版本：Alpha 预发行版和发行版。要确定要使用哪个版本，请查看the section called “AWS Blu Age 版本控制”或联系您的 AWS Blu Age 交付经理。

B AWS lu Age Runtime 的区域和存储分区

我们按地区和计算选择将 AWS Blu Age 运行时工件存储在不同的 Amazon S3 存储桶中。要访问你 AWS 区域 的 for AWS Blu Age Runtime 的存储桶，请使用下表中列出的名称。

AWS 区域	发行版存储桶	Alpha 预发行版存储桶
美国东部（俄亥俄州）	aws-bluage-runtime-artifacts-055777665268-us-east-2	aws-bluage-runtime-artifacts-dev-055777665268-us-east-2
美国东部（弗吉尼亚州北部）	aws-bluage-runtime-artifacts-139023371234-us-east-1	aws-bluage-runtime-artifacts-dev-139023371234-us-east-1
美国西部（北加利福尼亚）	aws-bluage-runtime-artifacts-788454048782-us-west-1	aws-bluage-runtime-artifacts-dev-788454048782-us-west-1
美国西部（俄勒冈州）	aws-bluage-runtime-artifacts-836771190483-us-west-2	aws-bluage-runtime-artifacts-dev-836771190483-us-west-2
加拿大（中部）	aws-bluage-runtime-artifacts-637423580979-ca-central-1	aws-bluage-runtime-artifacts-dev-637423580979-ca-central-1
欧洲地区（爱尔兰）	aws-bluage-runtime-artifacts-925278190477-eu-west-1	aws-bluage-runtime-artifacts-dev-925278190477-eu-west-1
欧洲地区（伦敦）	aws-bluage-runtime-artifacts-767397831990-eu-west-1	aws-bluage-runtime-artifacts-dev-767397831990-eu-west-1
欧洲地区（巴黎）	aws-bluage-runtime-artifacts-673009995881-eu-west-3	aws-bluage-runtime-artifacts-dev-673009995881-eu-west-3

AWS 区域	发行版存储桶	Alpha 预发行版存储桶
欧洲地区（法兰克福）	aws-bluage-runtime-artifacts-485196800481-eu-central-1	aws-bluage-runtime-artifacts-dev-485196800481-eu-central-1
欧洲地区（斯德哥尔摩）	aws-bluage-runtime-artifacts-654654484534-eu-north-1	aws-bluage-runtime-artifacts-dev-654654484534-eu-north-1
欧洲地区（米兰）	aws-bluage-runtime-artifacts-654654328338-eu-south-1	aws-bluage-runtime-artifacts-dev-654654328338-eu-south-1
欧洲（西班牙）	aws-bluage-runtime-artifacts-905417994954-eu-south-2	aws-bluage-runtime-artifacts-dev-905417994954-eu-south-2
南美洲（圣保罗）	aws-bluage-runtime-artifacts-737536804457-sa-east-1	aws-bluage-runtime-artifacts-dev-737536804457-sa-east-1
亚太地区（东京）	aws-bluage-runtime-artifacts-445578176276-ap-northeast-1	aws-bluage-runtime-artifacts-dev-445578176276-ap-northeast-1
亚太地区（首尔）	aws-bluage-runtime-artifacts-381492221498-ap-northeast-2	aws-bluage-runtime-artifacts-dev-381492221498-ap-northeast-2
亚太地区（大阪）	aws-bluage-runtime-artifacts-905418229615-ap-northeast-3	aws-bluage-runtime-artifacts-dev-905418229615-ap-northeast-3
亚太地区（新加坡）	aws-bluage-runtime-artifacts-767397774613-ap-southeast-1	aws-bluage-runtime-artifacts-dev-767397774613-ap-southeast-1
亚太地区（悉尼）	aws-bluage-runtime-artifacts-726160321909-ap-southeast-2	aws-bluage-runtime-artifacts-dev-726160321909-ap-southeast-2

AWS 区域	发行版存储桶	Alpha 预发行版存储桶
亚太地区（孟买）	aws-bluage-runtime-artifacts-905418353577-ap-south-1	aws-bluage-runtime-artifacts-dev-905418353577-ap-south-1
非洲（开普敦）	aws-bluage-runtime-artifacts-992382777663-af-south-1	aws-bluage-runtime-artifacts-dev-992382777663-af-south-1
以色列（特拉维夫）	aws-bluage-runtime-artifacts-471112516508-il-central-1	aws-bluage-runtime-artifacts-dev-471112516508-il-central-1

使用列 AWS CLI 出存储桶中的内容

登录后，您可以在终端中运行以下 AWS CLI 命令来列出存储桶的内容。

```
aws s3 ls bucket-name
```

bucket-name替换为上表 AWS 区域 中您的存储桶的名称。

此命令返回与不同版本的 AWS Blu Age Runtime 运行时相对应的文件夹列表，例如 Release 存储桶的以下文件夹：

```
PRE 4.0.0/
PRE 4.10.0/
```

或者以下列表对应于构建存储桶：

```
PRE 4.11.0/
PRE 4.13.0/
```

建议您使用最新可用版本。如果不能，请使用在应用程序重构阶段验证过的运行时版本。要列出特定版本的可用框架，请运行以下命令：

```
aws s3 ls s3://bucket-name/version/Framework/
```

bucket-name替换为你的存储桶version名称 AWS 区域 和你想要的版本。下面是两个示例：

对于发行版存储桶：

```
aws s3 ls s3://aws-bluage-runtime-artifacts-139023371234-us-east-1/4.10.0/Framework/
```

该命令返回 zip 文件列表，例如：

```
2025-04-08 16:11:19  152040176 aws-bluage-runtime-4.10.0.zip
2025-04-08 16:11:52  176518889 aws-bluage-webapps-4.10.0.zip
```

对于 Alpha 预发行版存储桶：

```
aws s3 ls s3://aws-bluage-runtime-artifacts-dev-139023371234-us-east-1/4.11.0/Framework/
```

该命令返回 zip 文件列表，例如：

```
2025-04-09 20:23:34  152304534 aws-bluage-runtime-4.11.0.zip
2025-04-09 20:24:07  176262381 aws-bluage-webapps-4.11.0.zip
```

下载框架

例如，您可以下载该框架来升级现有 Amazon EC2 实例上的 AWS Blu Age 运行时版本。

```
aws s3 cp s3://bucket-name/version/Framework/ folder-of-your-choice --recursive
```

其中：

folder-of-your-choice

要下载框架的文件夹路径。

例如：`aws s3 cp s3://aws-bluage-runtime-artifacts-139023371234-us-east-1/4.0.0/Framework/ . --recursive`

此命令将生成以下输出：

```
download: s3://aws-bluage-runtime-artifacts-139023371234-us-east-1/4.0.0/Framework/aws-bluage-webapps-4.0.0.zip to ./aws-bluage-webapps-4.0.0.zip
download: s3://aws-bluage-runtime-artifacts-139023371234-us-east-1/4.0.0/Framework/aws-bluage-runtime-4.0.0.zip to ./aws-bluage-runtime-4.0.0.zip
```

您可以使用如下命令列出框架文件：

```
ls -l
```

此命令将生成以下输出：

```
total 230928
-rw-rw-r-- 1 cloudshell-user cloudshell-user 152040176 Apr  8 16:11 aws-bluage-runtime-4.0.0.zip
-rw-rw-r-- 1 cloudshell-user cloudshell-user 176518889 Apr  8 16:11 aws-bluage-webapps-4.0.0.zip
```

> **Note**
> 出于安全考虑，对工件的访问可能会暂时中断，版本可能会被删除。我们强烈建议您将使用的工件存储在自己的账户中。在您的内部架构中应使用本地版本作为参考。

AWS Blu Age Runtime 的基础架构设置要求

本主题介绍运行 AWS Blu Age Runtime 所需的最低基础架构配置。以下过程描述了如何在您选择的计算机上设置 AWS Blu Age Runtime，以便在 B AWS lu Age Runtime 上部署现代化应用程序。您创建的资源必须位于具有专用于您的应用程序域的子网的 Amazon VPC 中。

主题

基础设施要求

创建安全组

如果您计划在 Amazon EKS 上使用亚马逊 EC2 实例，请跳过此过程，因为 Amazon EKS 集群创建过程会代表您创建一个安全组。请在以下过程中使用该安全组，而不是创建新的安全组。

1. 打开位于 https://console.aws.amazon.com/vpc/ 的 Amazon VPC 控制台。
2. 在左侧导航窗格中的安全下，选择安全组。

3. 在中央窗格中，选择创建安全组。
4. 在安全组名称字段中，输入 **`M2BluagePrivateLink-SG`**。
5. 在入站规则部分中，选择添加规则。
6. 对于类型，选择 HTTPS。
7. 对于源，请输入您的 VPC CIDR。
8. 在出站规则部分，选择添加规则。
9. 对于类型，选择 HTTPS。
10. 在目标位置字段，输入 **`0.0.0.0/0`**。
11. 选择创建安全组。

创建一个 IAM 角色

1. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在左侧导航窗格中的访问权限管理下，选择角色。
3. 在中央窗格中，选择创建角色。
4. 在用例部分，根据您的计算方案，选择以下选项之一：
   - EC2（适用于亚马逊 EC2 和亚马逊上的亚马逊 EKS EC2）
   - 弹性容器服务，然后是弹性容器服务的EC2角色（适用于亚马逊上的 Amazon ECS EC2）
   - Elastic Container Service，然后是 Elastic Container Service 任务（适用于由 Fargate 管理的 Amazon ECS）
5. 选择下一步。
6. 输入角色的名称，然后选择创建角色。

在亚马逊上运行 AWS Blu Age Runtime EC2

要创建 Amazon EC2 实例，请使用以下步骤。

创建一个 Amazon EC2 实例

1. 打开 Amazon EC2 控制台，网址为https://console.aws.amazon.com/ec2/。
2. 选择启动实例。
3. 对于实例类型，选择可 EC2 用的实例类型。
4. 在密钥对部分中，选择现有密钥对或创建新密钥对。

5. 在网络设置部分，选择选择现有安全组。
6. 对于通用安全组，请选择 M2-BluagePrivateLink SG。
7. 展开高级详细信息部分。
8. 对于 IAM 实例配置文件，选择您前面创建的 IAM 角色。
9. 选择启动实例。

在 Amazon EC2 实例上安装应用程序

1. 当 Amazon EC2 实例的状态更改为 “正在运行” 时，请连接到该实例。
2. 在实例上安装以下软件组件（请参阅中提及的版本the section called “AWS Blu Age 发行说明”）：
   - Java 运行时环境 (JRE)。
   - Apache Tomcat。
   - AWS Blu Age Runtime（在亚马逊上 EC2）。在 Apache Tomcat 安装文件夹的根目录下安装 AWS Blu Age 运行时（有些文件将被添加，而另一些文件将被覆盖）。

   要安装与 AWS Blu Age Runtime 存档一起提供的其他 Web 应用程序，请设置 Apache Tomcat 服务器的辅助实例，然后在该位置解压网页应用程序存档。有关详细说明，请参阅the section called “AWS 蓝光时代运行时神器”。

在亚马逊上的亚马逊 ECS 上运行 AWS Blu Age Runtime EC2

1. 创建一个 Amazon ECS 集群，将亚马逊 EC2 实例作为底层基础架构。请参阅亚马逊弹性容器服务开发者指南 EC2中的亚马逊上Windows入门。
2. 指定在前面的步骤中创建的 IAM 角色。
3. 选择 EC2 实例类型。
4. 在 Amazon EC2 实例的网络设置中，选择您在之前的步骤中创建的安全组。

在亚马逊 EKS 上运行 AWS Blu Age Runtime、亚马逊 EC2

1. 创建一个 Amazon EKS 集群。请参阅《Amazon EKS 用户指南》中的创建 Amazon EKS 集群。
2. 如前所述，系统会代表您创建一个安全组。
3. 创建节点组。指定在前面的步骤中创建的 IAM 角色。
4. 选择 EC2 实例类型。

5. Amazon EKS 会自动将安全组分配给生成的亚马逊 EC2实例。

在 Amazon ECS 上运行 AWS Blu Age Runtime 由 AWS Fargate

创建以 AWS Fargate（无服务器）作为底层基础设施的 Amazon ECS 集群。请参阅《Amazon Elastic Container Service 开发人员指南》中的 Fargate 入门。

AWS 蓝光时代运行时神器

AWS Blu Age Runtime 工件是用于部署和运行现代化应用程序的组件。本文档概述了不同类型的可用对象、它们的存储位置以及如何访问它们。

AWS 蓝光时代运行时神器

文物内容

在发行版和预发行版中，你会发现：

gapwalk-x.y.z.z.zip

此存档中 x.y.z 代表版本号（参见 AWS Blu Age 版本控制），包含执行 AWS Blu Age 应用程序所必需的核心 AWS Blu Age 运行时组件，包括：

Artifact	说明	Type	分发文件夹
gapwalk-application-x.y.z.war	Blu Age主网络应用程序 Gap walk	战争	webapps
*.jar	Blu Age运行时框架共享库 (Gapwalk)	罐子	共享
gapwalk-*-x.y.z.war	Blu Age其他 Web 应用程序（Utility-pgm、分层支持...）	战争	网络应用程序额外版
bac-x.y.z.war / jac-x.y.z.war	Blu Age管理控制台（非独立控制台）	战争	网络应用程序控制台

aws-bluage-webapps-x.y.z.zip

此存档中 x.y.z 遵循与上述相同的版本控制方案，包括Blu Age管理控制台（独立版）

- BAC（Blusam控制台）WAR 文件，用于监控Blusam数据库。
- JAC（JICS 控制台）WAR 文件，用于监视 JICS 数据库。
- 必要的支持库。

Package 签名验证

已发布的 zip 文件以数字签名的 ZIP 存档形式提供，以确保安全性和真实性。数字签名有助于验证您下载的文件是真实的、未经修改的，并且是由我们的组织正式发布的。这样可以防止篡改，并防范可能被恶意分发的恶意软件或受损文件。

要在使用 ZIP 文件之前验证其签名，可以使用以下命令：

```
jarsigner -verify -certs -verbose aws-bluage-runtime-x.y.z.zip
```

以下是认证文件的结果示例：

```
...
jar verified.
```

有关如何解决安全漏洞的详细信息，请参阅 AWS 大型机现代化重构和 AWS Blu Age 版本概述。

> **Note**
> 虽然我们努力发布不带新产品的产品 CVEs，但以后 CVEs 可能会出现新的产品。

在亚马逊上部署 AWS Blu Age 运行时 EC2

您可以通过本节中的主题学习如何在亚马逊 EC2上设置 AWS Blu Age Runtime、如何更新运行时版本、如何使用亚马逊 CloudWatch 警报监控您的部署，以及如何添加许可依赖项。这些说明适用于您创建亚马逊 EC2 实例，以及在亚马逊上使用 Amazon ECS EC2 或在亚马逊上使用 Amazon EKS 时 EC2。

从 alpha 版本 5.60.0 开始， AWS Blu Age Runtime 发行版包括一个包含全面的 README.md 文件的部署文件夹。本指南提供了 step-by-step Amazon EC2 部署说明、配置选项和疑难解答提示。

主题

- 在亚马逊上设置 AWS Blu Age Runtime EC2
- 在亚马逊上升级 AWS Blu Age Runtime EC2
- 设置 AWS Blu Age Runtime（在亚马逊上 EC2）亚马逊 CloudWatch 警报
- 在亚马逊的 AWS Blu Age Runtime 中设置许可依赖关系 EC2

在亚马逊上设置 AWS Blu Age Runtime EC2

本主题介绍如何在亚马逊上使用 AWS Blu Age Runtime 设置和部署 PlanetsDemo 示例应用程序 EC2。

主题

- 先决条件
- 设置
- 测试部署的应用程序

先决条件

在开始之前，请确保满足以下先决条件：

- AWS CLI 按照配置 AWS CLI 中的步骤进行配置。
- 完成 the section called “AWS 蓝光时代运行时必备条件”和the section called “入门 AWS Blu Age 运行时”。
- 使用支持的 EC2 实例类型之一创建 Amazon 实例。有关更多信息，请参阅 Amazon EC2 Linux 实例入门。
- 确保您可以成功连接到 Amazon EC2 实例，例如使用 SSM。

> **Note**
>
> 在本指南中，假设 Tomcat 的安装路径为。`/m2-anywhere/tomcat-gapwalk/velocity`在按照以下说明进行操作时，请确保使用此路径，或者根据您选择的路径调整以下指令。

- 将 AWS Blu Age Runtime（在亚马逊上 EC2）直接下载并解压到 velocity 目录`/m2-anywhere/tomcat-gapwalk/velocity`（Tomcat 安装文件夹）。有关如何检索 AWS Blu Age Runtime 工件（包括有关存储、访问和内容的信息）的说明，请参阅the section called “AWS 蓝光时代运行时神器”。

- 下载PlanetsDemo应用程序档案。
- 解压缩存档文件，并将应用程序上传到您选择的 Amazon S3 存储桶。
- 为 JICS 创建 Amazon Aurora PostgreSQL 数据库。 AWS Blu Age Runtime 将在首次启动时自动执行`PlanetsDemo-v1/jics/sql/initJics.sql`脚本。有关如何创建 Amazon Aurora PostgreSQL 数据库的信息，请参阅创建并连接到 Aurora PostgreSQL 数据库集群。

设置

要设置 PlanetsDemo 示例应用程序，请完成以下步骤。

1. 连接到您的亚马逊 EC2 实例，然后转到 Apache Tomcat 10 安装`conf`文件夹下的文件夹。打开 `catalina.properties` 文件进行编辑，将以 `common.loader` 开头的行替换为以下行。

```
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. 导航到 `/m2-anywhere/tomcat-gapwalk/velocity/webapps`文件夹。
3. 使用以下命令从 Amazon S3 存储桶中复制`PlanetsDemo-v1/webapps/`文件夹中可用的 PlanetsDemo 二进制文件。

```
aws s3 cp s3://path-to-demo-app-webapps/ . --recursive
```

> **Note**
>
> `path-to-demo-app-webapps`替换为之前解压缩存档的存储桶的正确 Amazon S3 URI。 PlanetsDemo

4. 将 `PlanetsDemo-v1/config/` 文件夹的内容复制到 `/m2-anywhere/tomcat-gapwalk/velocity/config/`。
5. 在 `/m2-anywhere/tomcat-gapwalk/velocity/config/application-main.yml` 文件的以下代码段中，提供作为先决条件一部分而创建的数据库的连接信息。有关更多信息，请参阅创建并连接到 Aurora PostgreSQL 数据库集群。

```
datasource:
   jicsDs:
     driver-class-name :
```

```
    url:
    username:
    password:
    type :
```

6. 启动您的 Apache Tomcat 服务器并验证日志。

```
/m2-anywhere/tomcat-gapwalk/velocity/startup.sh

tail -f /m2-anywhere/tomcat-gapwalk/velocity/logs/catalina.log
```

如果您发现以 C 后跟数字开头的错误代码（例如 CXXXX），请记下错误消息。例如，错误代码 C5102 是一个常见错误，表示基础设施配置不正确。

测试部署的应用程序

有关如何测试 PlanetsDemo 应用程序的示例，请参阅the section called "测试 PlanetsDemo 应用程序"。

在亚马逊上升级 AWS Blu Age Runtime EC2

本指南介绍如何在亚马逊上升级 AWS Blu Age Runtime EC2。

主题

先决条件

在开始之前，确保满足以下先决条件：

- 要查看是否有针对您的版本的特定说明，请参阅 the section called "升级 AWS 蓝光时代"。
- 完成 the section called "AWS 蓝光时代运行时必备条件"和the section called "入门 AWS Blu Age 运行时"。
- 确保您的亚马逊 EC2 实例已安装了 AWS Blu Age 运行时。有关更多信息，请参阅 Amazon EC2 Linux 实例入门。
- 确保您可以成功连接到 Amazon EC2 实例，例如，使用 SSM。

- 下载你想要升级到的 AWS Blu Age 运行时版本。该框架由两个存档文件组成：`gapwalk-x.y.z.zip`和`aws-bluage-webapps-x.y.z.zip`。有关更多信息，请参阅 the section called “AWS 蓝光时代运行时神器”。

在 Amazon EC2 实例中升级 AWS Blu Age 运行时

完成以下步骤以升级 AWS Blu Age 运行时间。

1. 通过运行以下命令连接到您的 Amazon EC2 实例并将用户更改为 s u。

   ```
   sudo su
   ```

   您需要超级用户权限才能运行本教程中的命令。
2. 使用以下命令提取二进制文件。运行每个文件夹中的命令。

   ```
   unzip gapwalk-x.y.z.zip
   unzip aws-bluage-webapps-x.y.z.zip
   ```
3. 使用以下命令以停止 Apache Tomcat 服务。

   ```
   systemctl stop tomcat.service
   systemctl stop tomcat-webapps.service
   ```
4. 将 `<your-tomcat-path>/shared/` 的内容替换为 `gapwalk-x.y.z/shared/` 的内容。
5. 将 `<your-tomcat-path>/webapps/gapwalk-application.war`替换为 `gapwalk-x.y.z/webapps/gapwalk-application.war`。
6. 将 `<your-tomcat-path>/webapps/` 中的 war 文件（即 `bac.war` 和 `jac.war`）替换为 `aws-bluage-webapps-x.y.z/velocity/webapps/` 中的相同文件。
7. 运行以下命令来启动 Apache Tomcat 服务。

   ```
   systemctl start tomcat.service
   systemctl start tomcat-webapps.service
   ```
8. 检查日志。

运行以下命令来查看部署的应用程序的状态。

```
curl http://localhost:8080/gapwalk-application/
```

将会出现以下消息。

```
Jics application is running
```

```
curl http://localhost:8181/jac/api/services/rest/jicsservice/
```

将会出现以下消息。

```
Jics application is running
```

```
curl http://localhost:8181/bac/api/services/rest/bluesamserver/serverIsUp
```

响应应为空。

AWS Blu Age 运行时已成功升级。

在容器中升级 AWS Blu Age 运行时间

完成以下步骤以升级 AWS Blu Age 运行时间。

1. 使用所需的 AWS Blu Age 运行时版本重建 Docker 镜像。有关说明，请参阅the section called “在亚马逊上设置 AWS Blu Age Runtime EC2”。
2. 将 Docker 映像推送到 Amazon ECR 存储库。
3. 停止并重启 Amazon ECS 或 Amazon EKS 服务。
4. 检查日志。

AWS 蓝光时代运行时间已成功升级。

设置 AWS Blu Age Runtime（在亚马逊上 EC2）亚马逊 CloudWatch 警报

您可以设置 CloudWatch 为接收应用程序日志，并添加警报以警告您可能出现的错误。这样，在部署的应用程序遇到异常时，您就可以收到更多可见的通知。以下各节可帮助您了解和了解 CloudWatch 日志和警报设置的配置。

启用 CloudWatch 日志记录

要启用 CloudWatch 日志记录，请使用下一节中提供的配置模板在 config 文件夹中创建一个`logback-cloudwatch.xml`文件。然后，通过取消注释以下几行来引用`application-main.yml`文件中的此文件：

```
# logging:
#  config: classpath:logback-cloudwatch.xml
```

取消注释这些行后，将激活 CloudWatch 日志记录，并且可以按照以下各节中的说明进行进一步配置。

CloudWatch 日志配置

使用以下模板创建`logback-cloudwatch.xml`文件：

```
<?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>
```

<appender name="cloudwatch"/> 元素之外的所有内容均为标准 logback 配置。此文件中有两个附加程序：一个用于向控制台发送日志的控制台附加程序和一个用于向其发送日志的 CloudWatch 附加程序。 CloudWatch

`root` 元素中的 `level` 属性指定了整个应用程序的日志记录级别。

标签 <appender name="cloudwatch"/> 中的必需值包括：

- <logGroup/>:在中设置日志组的 CloudWatch名称。如果未指定，则默认值为 `BluAgeRuntimeOnEC2-Logs`。如果日志组不存在，系统将自动创建。可以通过配置来更改此行为，具体内容将在下文中介绍。
- <logStream/>：在中设置 LogStream 的名称（在日志组内）。 CloudWatch

可选值：

- <region/>：覆盖将写入日志流的区域。默认情况下，日志会进入与 EC2 实例相同的区域。
- <layout/>：日志消息将使用的模式。
- <maxbatchsize/>：每个操作要发送到的最大日志消息 CloudWatch 数。
- <maxbatchtimemillis/>：允许写入 CloudWatch 日志的时间（以毫秒为单位）。
- <maxqueuewaittimemillis/>：尝试在内部日志队列中插入请求的时间（以毫秒为单位）。
- <internalqueuesize/>：内部队列的最大大小。
- <createlogdests/>：创建日志组和日志流（如果不存在）。
- <initialwaittimemillis/>: 您希望线程在启动时处于睡眠状态的时间。该初始等待时间允许日志的初始累积。
- <maxeventmessagesize/>：日志事件的最大大小。超过此大小的日志将不会被发送。
- <truncateeventmessages/>：截断过长的消息。
- <printrejectedevents/>：启用紧急 Appender。

CloudWatch 设置

为了使上述配置正确推送日志 CloudWatch，请更新您的 EC2 Amazon IAM 实例配置文件角色以授予其对 `BluAgeRuntimeOnEC2-Logs`日志组及其日志流的额外权限：

- `logs:CreateLogStream`
- `logs:DescribeLogStreams`
- `logs:CreateLogGroup`
- `logs:PutLogEvents`
- `logs:DescribeLogGroups`

警报设置

借助 CloudWatch 日志，您可以根据应用程序和需求配置不同的指标和警报。具体而言，您可以为使用情况设置主动警报，以便在出现可能使您的应用程序处于宽限期（最终使其完全无法运行）的错误时收到警告。为此，您可以在日志中添加一个与 “Error C5001” 字符串相关的指标，该指标会突出显示与 AWS Blu Age 控制系统连接中的错误。然后，您可以定义一个对此指标做出反应的警报。

在亚马逊的 AWS Blu Age Runtime 中设置许可依赖关系 EC2

本指南介绍如何设置其他许可依赖项，供亚马逊上的 AWS Blu Age Runtime 使用 EC2。

主题

- 先决条件
- 安装步骤

先决条件

在开始之前，请确保满足以下先决条件：

- 完成 the section called “AWS 蓝光时代运行时必备条件”和the section called “入门 AWS Blu Age 运行时”。
- 确保你有一个安装了 AWS Blu Age Runtime（在亚马逊上 EC2）的亚马逊 EC2 实例。有关更多信息，请参阅 Amazon EC2 Linux 实例入门。
- 确保您可以成功连接到 Amazon EC2 实例，例如，使用 SSM。
- 从相应源中获取以下依赖项。

Oracle 数据库

提供 Oracle 数据库驱动程序。我们使用版本 ojdbc11-23.3.0.23.09.jar 测试了 B AWS lu Age Runtime（在亚马逊上 EC2）的功能，但可能兼容更新的版本。

IBM MQ 连接

提供一个 IBM MQ 客户端。我们使用 com.ibm.mq.jakarta.client-9.3.4.1.jakarta.client-9.3.4.1.jar 版本测试了 Bl AWS u Age Runtime（在亚马逊上 EC2）的功能，但可能兼容更新的版本。

使用此依赖项版本时，还要提供以下传递依赖项：

- bcprov-jdk15to18-1.76.jar

- bcpkix-jdk15to18-1.76.jar
- bcutil-jdk15to18-1.76.jar

DDS 打印机文件

提供 Jasper 报告库（m https://com unity.jaspersoft。 com/download-jaspersoft/community-版）。我们使用 j asperreports-6.16.0.jar 测试了 AWS Blu Age Runtime（在亚马逊上 EC2）功能，但可能兼容更新的版本。

使用此依赖项版本时，还要提供以下传递依赖项：

- 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
- commons-beanutils-1.9.4.jar
- commons-collections-3.2.2.jar

安装步骤

要安装依赖项，请完成以下步骤。

1. 通过运行以下命令连接到您的 Amazon EC2 实例并将用户更改为 s u。

   ```
   sudo su
   ```

   您需要超级用户权限才能运行本教程中的命令。

2. 导航到 `<your-tomcat-path>/extra/`文件夹。

   ```
   cd <your-tomcat-path>/extra/
   ```

3. 根据需要将上述任何依赖项复制到此文件夹。

> **Note**
>
> 对于具有 Oracle 托管数据库的 JAC 和 BAC Web 应用程序，请确保在此文件夹中包含 Oracle 数据库驱动程序 (ojdbc)。

4. 通过运行以下命令停止和启动 tomcat.service。

```
systemctl stop tomcat.service
```

```
systemctl start tomcat.service
```

5. 检查该服务的状态，确保其正在运行。

```
systemctl status tomcat.service
```

6. 验证日志。

在亚马逊 ECS 和亚马逊 EKS 的容器上部署 AWS Blu Age Runtime

您可以使用本节中的主题来学习如何在容器上设置 AWS Blu Age Runtime 以将其部署在 Amazon ECS（由亚马逊管理 EC2 或 AWS Fargate）和由亚马逊管理的 Amazon EKS 上 EC2，如何更新运行时版本，如何使用亚马逊 CloudWatch 警报监控您的部署，以及如何添加许可依赖项。

> **Note**
>
> 这与由管理的 Amazon EKS 不兼容 AWS Fargate。

主题

在容器上设置 AWS Blu Age 运行时

本主题介绍如何在 docker 容器上使用 AWS Blu Age Runtime 设置和部署 PlanetsDemo 示例应用程序。

AWS 容器上的 Blu Age Runtime 适用于亚马逊管理的亚马逊 ECS EC2、由亚马逊管理的 Amazon ECS 和由 AWS Fargate亚马逊管理的 Amazon EKS EC2。它与由管理的 Amazon EKS 不兼容 AWS Fargate。

主题

- 先决条件
- 设置
- 测试部署的应用程序

先决条件

在开始之前，请确保满足以下先决条件：

- AWS CLI 按照配置 AWS CLI 中的步骤进行配置。
- 完成 the section called “AWS 蓝光时代运行时必备条件”和the section called “入门 AWS Blu Age 运行时”。
- 下载 AWS Blu Age 运行时二进制文件。有关说明，请参阅the section called “入门 AWS Blu Age 运行时”。
- 下载 Apache Tomcat 10 二进制文件。
- 下载PlanetsDemo应用程序档案。
- 为 JICS 创建一个 Amazon Aurora PostgreSQL 数据库，然后对其运行 `PlanetsDemo-v1/jics/sql/initJics.sql` 查询。有关如何创建 Amazon Aurora PostgreSQL 数据库的信息，请参阅创建并连接到 Aurora PostgreSQL 数据库集群。

设置

要设置 PlanetsDemo 示例应用程序，请完成以下步骤。

1. 下载 Apache Tomcat 二进制文件后，解压缩内容并转到 `conf` 文件夹。打开 `catalina.properties` 文件进行编辑，将以 `common.loader` 开头的行替换为以下行。

```
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. 使用 tar 命令压缩 Apache Tomcat 文件夹以构建 “tar.gz”归档。
3. 准备一个 Dockerfile，用于根据提供的运行时二进制文件和 Apache Tomcat 服务器二进制文件构建您的自定义映像。请见以下示例 Dockerfile。目标是安装 Apache Tomcat 10，然后安装在 Apache Tomcat 10 安装目录根目录中提取的 AWS Blu Age 运行时（适用于由管理的 Amazon ECS AWS Fargate），然后安装名为的示例现代化应用程序。 PlanetsDemo

> **Note**
>
> 在本示例 Dockerfile 中使用的 install-gapwalk.sh 和 install-app.sh 脚本的内容列在 Dockerfile 之后。

```
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-runtime-4.x.x.zip /usr/local/velocity/installation/gapwalk/
gapwalk.zip

# run relevant installation scripts
RUN ./install-gapwalk.sh
RUN ./install-app.sh

EXPOSE 8080
EXPOSE 8081
# ...

WORKDIR /bluage/tomcat.gapwalk/velocity
# Run Command to start Tomcat server
CMD ["sh", "-c", "sudo bin/catalina.sh run"]
```

以下是 `install-gapwalk.sh.` 的内容

```
# 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.zip ${TEMP_DIR}
sudo cp /usr/local/velocity/installation/tomcat.tar.gz ${TEMP_DIR}
# Create velocity dir
mkdir -p /bluage/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-10.x.x/* /bluage/tomcat.gapwalk/velocity
# Remove default webapps of Tomcat
rm -f /bluage-on-fargate/tomcat.gapwalk/velocity/webapps/*
# Extract Blu Age runtime at velocity dir
unzip ${TEMP_DIR}/gapwalk.zip -d /bluage/tomcat.gapwalk
# Remove temporary extraction dir
sudo rm -rf ${TEMP_DIR}
```

以下是 `install-app.sh` 的内容。

```
#!/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. 在 `application-main.yml` 文件（位于 `{TOMCAT_GAPWALK_DIR}/config` 文件夹中）的以下代码段中，提供作为先决条件一部分而创建的数据库的连接信息。有关更多信息，请参阅创建并连接到 Aurora PostgreSQL 数据库集群。

```
datasource:
   jicsDs:
     driver-class-name :
     url:
     username:
     password:
     type :
```

5. 构建映像推送到您的 Amazon ECR 存储库。有关说明，请参阅《Amazon Elastic Container Registry 用户指南》中的推送 Docker 映像。然后，根据您的情况，使用您的 Amazon ECR 映像创建一个 Amazon EKS 容器组或 Amazon ECS 任务定义，然后将其部署到您的集群中。有关创建这些项目的示例，请参阅《Amazon Elastic Container Service（Amazon ECS）开发人员指南》中的使用控制台创建任务定义和《Amazon EKS 用户指南》中的部署示例应用程序。
6. 具体而言，对于由 AWS Fargate管理的 Amazon ECS，创建任务定义时，请使用您在初始基础设施设置过程中创建的 IAM 角色。然后，在创建服务时，请展开网络部分，配置您在初始基础设施设置过程中创建的 VPC、子网和安全组。请参阅 A WS Blu Age 运行时的基础设施设置要求。

测试部署的应用程序

有关如何测试 PlanetsDemo 应用程序的示例，请参阅the section called “测试 PlanetsDemo 应用程序”。

升级容器上的 AWS Blu Age 运行时间

本指南介绍如何在容器上升级 AWS Blu Age 运行时。为此，你需要先完成一些先决条件，然后使用 Docker 镜像来升级 AWS Blu Age 运行时。

主题

- 先决条件
- 升级 AWS Blu Age 运行时间

先决条件

在开始之前，确保满足以下先决条件：

- 完成 the section called “AWS 蓝光时代运行时必备条件”和the section called “入门 AWS Blu Age 运行时”。
- 下载你想要升级到的 AWS Blu Age Runtime 版本。有关更多信息，请参阅 the section called “入门 AWS Blu Age 运行时”。该框架由两个二进制文件组成：`aws-bluage-runtime-x.y.z.zip` 和 `aws-bluage-webapps-x.y.z.zip`。

升级 AWS Blu Age 运行时间

完成以下步骤以升级 AWS Blu Age 运行时间。

1. 使用所需的 AWS Blu Age 运行时版本重建 Docker 镜像。有关说明，请参阅the section called “在容器上设置 AWS Blu Age 运行时”。
2. 将 Docker 映像推送到 Amazon ECR 存储库。
3. 停止并重启 Amazon ECS 或 Amazon EKS 服务。
4. 验证日志。

AWS 蓝光时代运行时间已成功升级。

在容器上为 AWS Blu Age Runtime 设置亚马逊 CloudWatch 警报

您可以设置 CloudWatch 为在部署的应用程序遇到异常时收到更多可见的通知。这可以帮助您监控重定向到的应用程序日志 CloudWatch，并添加警报以警告您可能出现的错误。

警报设置

借助 CloudWatch 日志，您可以根据您的应用程序和需求配置任意数量的指标和警报。

具体而言，您可以在创建集群期间直接为使用情况警报设置主动警报，以便在发生错误时收到通知。要突出显示与 AWS Blu Age 控制系统连接中的错误，请在日志中添加与字符串 “Error C” 相关的指标。然后，您可以定义一个对此指标做出反应的警报。

在容器上的 AWS Blu Age Runtime 中设置许可依赖关系

本主题介绍如何设置可与 AWS Blu Age 运行时（在容器上）搭配使用的其他许可依赖项。

主题

- 先决条件
- 概览

先决条件

在开始之前，请确保满足以下先决条件：

- 完成 the section called “AWS 蓝光时代运行时必备条件”和the section called “入门 AWS Blu Age 运行时”。
- 从相应源中获取以下依赖项。

Oracle 数据库

提供 Oracle 数据库驱动程序。例如，ojdbc11-23.3.0.23.09.jar。

IBM MQ 连接

提供一个 IBM MQ 客户端。例如，com.ibm.mq.jakarta.client-9.3.4.1.jar。

使用此依赖项版本时，还要提供以下传递依赖项：

- bcprov-jdk15to18-1.76.jar
- bcpkix-jdk15to18-1.76.jar
- bcutil-jdk15to18-1.76.jar

DDS 打印机文件

提供 Jasper 报告库（m https://com unity.jaspersoft。 com/download-jaspersoft/community-版）。例如 jasperreports-6.16.0.jar，不过可以兼容更新的版本。

使用此依赖项版本时，还要提供以下传递依赖项：

- 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
- commons-beanutils-1.9.4.jar
- commons-collections-3.2.2.jar

概览

要安装依赖项，请完成以下步骤。

1. 根据需要将上述任意依赖项复制到 Docker 映像构建文件夹中。
2. 如果您的 JICS 数据库托管在 Oracle 上，请在中提供 Oracle 数据库驱动程序。*your-tomcat-path*/extra
3. 在您的 Dockerfile 上，将这些依赖项复制到 *your-tomcat-path*/extra 中。
4. 构建您的 Docker 映像并将其推送至 Amazon ECR。
5. 停止并重启 Amazon ECS 或 Amazon EKS 服务。
6. 检查日志。

测试 PlanetsDemo 应用程序

要检查已部署 PlanetsDemo 应用程序的状态，请在替换`load-balancer-DNS-name``listener-port`、和之后`web-binary-name`使用正确的设置值运行以下命令。

```
curl http://load-balancer-DNS-name:listener-port/gapwalk-application/
```

如果应用程序正在运行，将显示以下输出消息：`Jics application is running`。

接下来，运行以下命令：

```
curl http://load-balancer-DNS-name:listener-port/jac/api/services/rest/jicsservice/
```

如果应用程序正在运行，将显示以下输出消息：Jics application is running。

```
Jics application is running
```

如果您已配置Blusam，则在运行以下命令时可能会出现空响应。

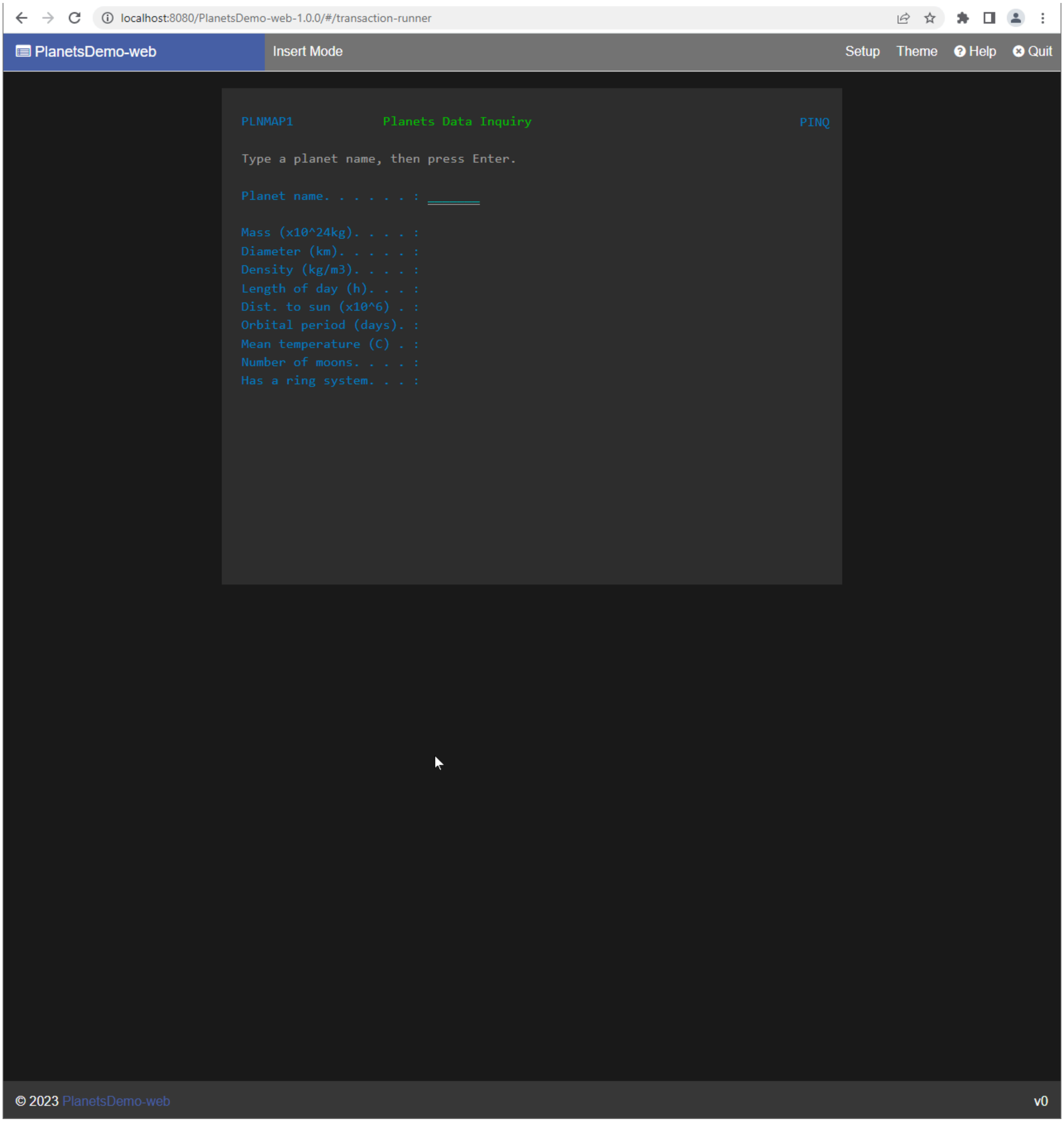
```
curl http://load-balancer-DNS-name:listener-port/bac/api/services/rest/bluesamserver/
serverIsUp
```

注意 Web 二进制文件的名称（PlanetsDemo-web-1.0.0，如果未更改）。要访问 PlanetsDemo 应用程序，请使用以下格式的 URL。

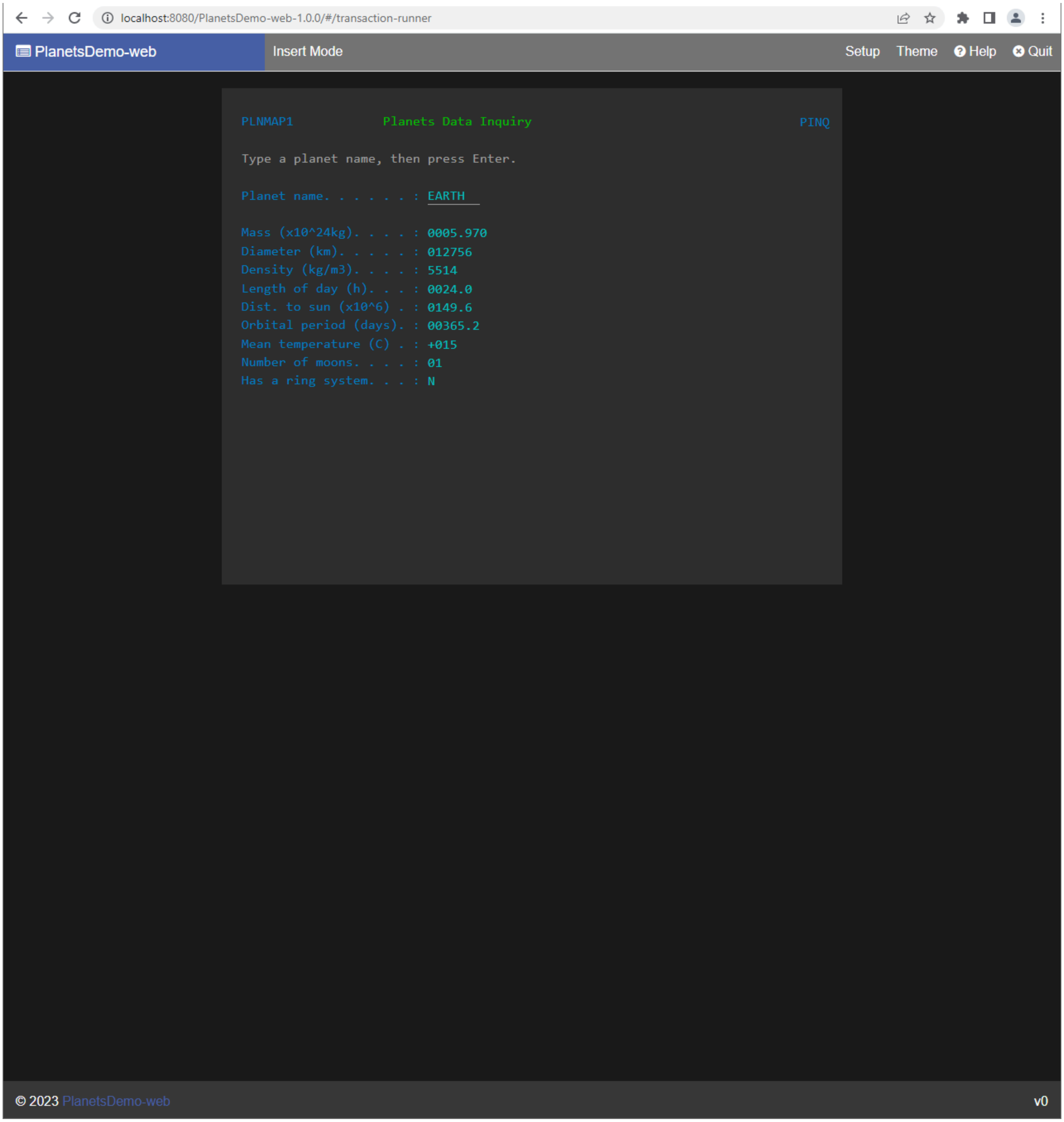
```
https://load-balancer-DNS-name:listener-port/web-binary-name
```

PlanetsDemo 应用程序启动后，将显示主页。

在文本框中，输入 PINQ，然后按 Enter。随后显示数据查询页面。



例如，在 PlanetsDemo 名称字段中输入 EARTH，然后按 Enter。随后将显示您输入的行星对应的页面。



使用 Blu Age Developer IDE 修改源代码

如果您使用的是 AWS托管的 AWS Blu Age 运行时引擎，则可以使用 Blu Age Developer 来修改生成的源代码。在以下情况下可能需要修改源代码：出于某种原因需要更新经过现代化改造的代码，

或者部分遗留源代码无法进行现代化改造。您可以通过亚马逊 WorkSpaces 应用程序访问 Blu Age Developer。本节介绍如何在 WorkSpaces 应用程序上设置 Blu Age Developer。它还说明了如何使用示例应用程序使用 Blu Age Developer 更新源代码 PlanetsDemo。

主题

教程：为 AWS Blu Age 开发者 IDE 设置 WorkSpaces 应用程序

AWS 大型机现代化通过 Amazon WorkSpaces 应用程序提供了多种工具。 WorkSpaces Applications 是一项完全托管的安全应用程序流服务，它允许您在不重写应用程序的情况下将桌面应用程序流式传输给用户。 WorkSpaces 应用程序使用户可以即时访问他们需要的应用程序，并在他们选择的设备上提供响应灵敏、流畅的用户体验。使用 WorkSpaces 应用程序托管特定于运行时引擎的工具，使客户应用程序团队能够直接通过其网络浏览器使用这些工具，与存储在 Amazon S3 存储桶或存储库中的应用程序文件进行交互。 CodeCommit

有关 WorkSpaces 应用程序中浏览器支持的信息，请参阅《亚马逊 WorkSpaces 应用程序管理指南》中的 “系统要求和功能支持（Web 浏览器）”。如果您在使用 WorkSpaces 应用程序时遇到问题，请参阅《Amazon WorkSpaces 应用程序管理指南》中的 AppStream2.0 用户问题疑难解答。

本文档介绍如何在 WorkSpaces 应用程序群中设置 AWS Blu Age 开发者 IDE。

主题

先决条件

对于初次使用的用户，请执行以下操作：

1. 在家中导航到 AppStream 2.0 https://console.aws.amazon.com/appstream2/主机。
2. 选择开始使用。
3. 选择 Skip。

> **Important**
>
> Amazon WorkSpaces Applications 使用 IAM 角色来管理您的 WorkSpaces 应用程序资源，并 AWS 将在您执行此操作时创建这些角色。

然后，下载存档文件，其中包含在 “ WorkSpaces 应用程序” 下设置 AWS Blu Age Developer IDE 所需的工件。

> **Note**
>
> 这是一个大文件。如果您在操作超时方面遇到问题，我们建议您使用 Amazon EC2 实例来提高上传和下载性能。有关启动和连接亚马逊 EC2 实例的更多信息，请参阅亚马逊入门 EC2。

步骤 1：创建 Amazon S3 存储桶

创建一个 Amazon S3 存储桶，该存储桶与您将要创建的 WorkSpaces 应用程序队列 AWS 区域 相同。该存储桶将包含完成本教程所需的构件。有关存储桶的更多信息，请参阅创建存储桶。

步骤 2：将策略附加到 S3 存储桶

将以下策略附加到您为本教程创建的存储桶。有关将策略附加到 S3 存储桶的更多信息，请参阅添加存储桶策略。

请务必将 `amzn-s3-demo-bucket` 替换为您创建的存储桶的实际名称。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Sid": "AllowAppStream2.0ToRetrieveObjects",
        "Effect": "Allow",
        "Principal": {
```

```
                "Service": "appstream.amazonaws.com"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
        }]
}
```

步骤 3：将文件上传到 Amazon S3 存储桶

解压缩您在先决条件中下载的文件，然后将该 `appstream` 文件夹上传到您的存储桶。上传此文件夹会在您的存储桶中创建正确的结构。有关更多信息，请参阅《Amazon S3 用户指南》中的上传对象。

第 4 步：下载 CloudFormation 模板

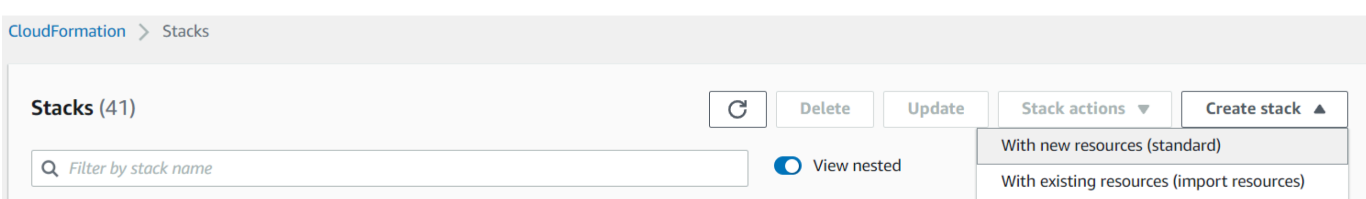
下载以下 CloudFormation 模板。您需要这些模板来创建和填充 WorkSpaces 应用程序队列。

- 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

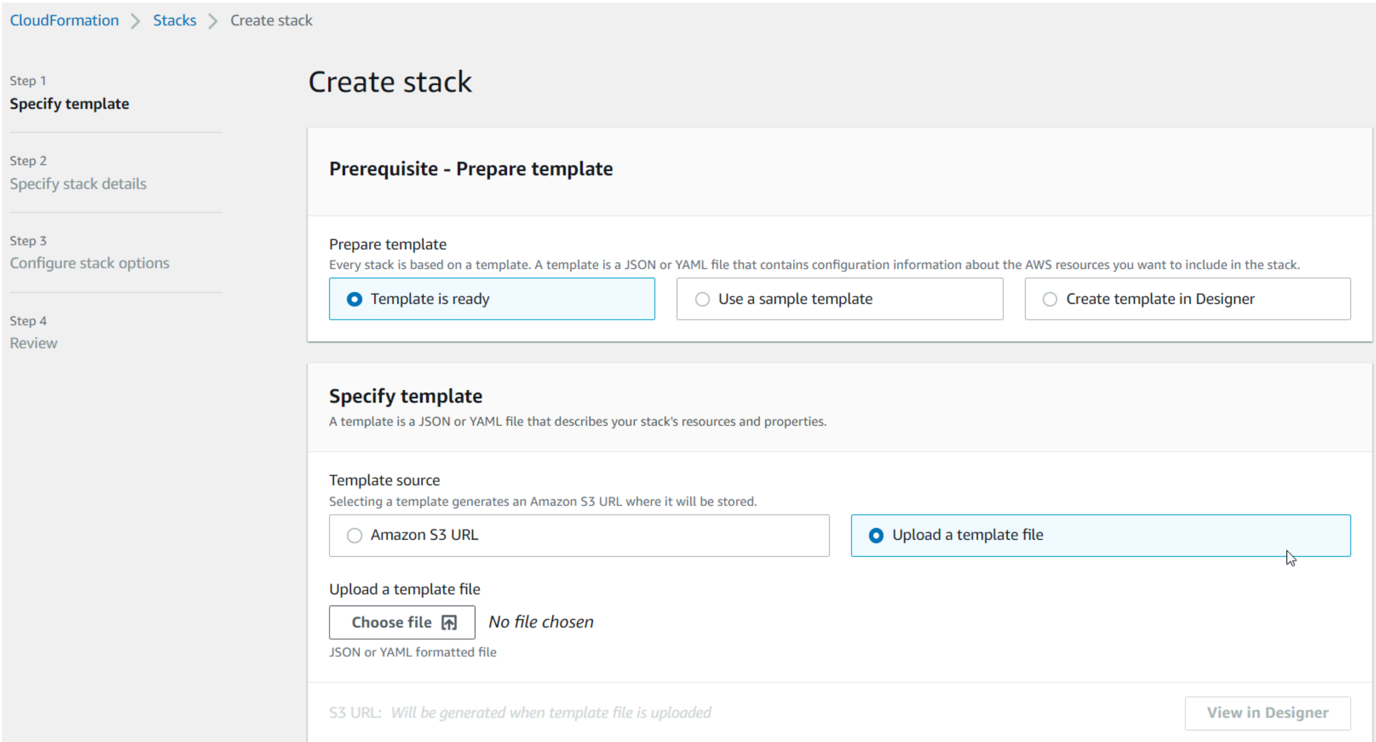
第 5 步：使用创建舰队 CloudFormation

在此步骤中，您将使用`cfn-m2-appstream-elastic-fleet-linux.yaml` CloudFormation 模板创建 WorkSpaces 应用程序队列和堆栈来托管 AWS Blu Age Developer IDE。创建队列和堆栈后，您将运行在上一步中下载的其他 CloudFormation 模板来安装开发者 IDE 和其他必需的工具。

1. 在 AWS 管理控制台 CloudFormation 中导航到，然后选择堆栈。
2. 在堆栈中，选择创建堆栈和使用新资源(标准)。



3. 在创建堆栈中，选择选择现有模板并上传模板文件：



4. 选择选择文件，并导航到文件 `cfn-m2-appstream-elastic-fleet-linux.yaml`。选择下一步。
5. 在指定堆栈集详细信息页面上，提供以下信息：

   - 堆栈的名称。
   - 您的默认安全组和该安全组的两个子网。

   > **Note**
   > 安全组的两个子网必须位于不同的可用区中。

6. 选择下一步。
7. 向下导航页面并选择我确认这 CloudFormation 可能会创建带有自定义名称的 IAM 资源。 。
8. 选择下一步。
9. 查看详细信息，然后选择 “提交”。
10. 创建队列后，使用所有其他下载的模板创建 CloudFormation 堆栈以完成应用程序的设置。确保BucketName每次都更新以指向正确的 S3 存储桶。你可以在 CloudFormation 控制台BucketName中编辑。或者，您可以直接编辑模板文件并更新 `S3Bucket` 属性。

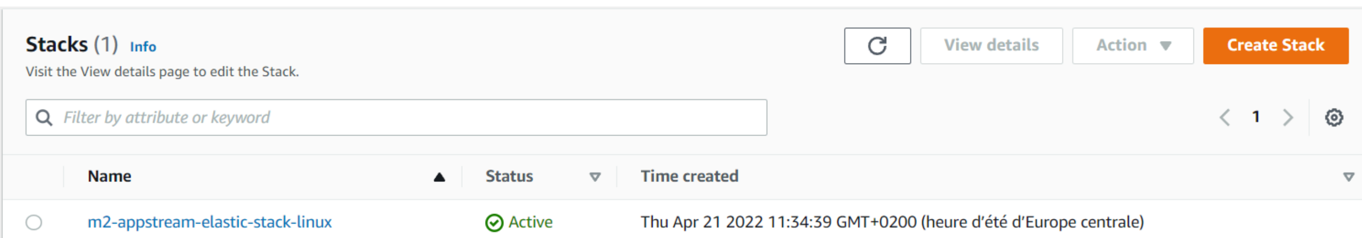
> **Note**
>
> 下载的模板预期使用名为 `appstream/bluage/developer-ide/` 的文件夹结构在 S3 存储桶中查找资产。存储桶必须与您创建的队列 AWS 区域 相同。

> **Important**
>
> 运行步骤 4 中下载的所有 CloudFormation 脚本以正确设置应用程序。

步骤 6：访问实例

创建并启动实例集后，您可以创建一个临时链接，以便通过本机客户端访问实例集。

1. 导航到中的 WorkSpaces 应用程序， AWS 管理控制台 然后选择之前创建的堆栈：



2. 在堆栈详细信息页面上，选择堆栈，然后选择关联舰队。
3. 在提示中，选择您之前创建并启动的舰队。
4. 选择关联 。
5. 选择关联堆栈，然后从 “操作” 菜单中选择 “创建直播 URL”，输入任意用户 ID 和 URL 过期时间，然后选择 “获取 URL”。您获取的 URL 可用于流式传输到浏览器或本机客户端。我们建议您流式传输到本机客户端。

清理 资源

有关清理已创建堆栈和队列的过程，请参阅[创建 WorkSpaces 应用程序队列和堆栈]。

删除 WorkSpaces 应用程序对象后，您或账户管理员还可以清理应用程序设置和主文件夹的 S3 存储桶。

> **Note**
>
> 给定用户的主文件夹在所有队列中都是唯一的，因此，如果同一账户中的其他 WorkSpaces 应用程序堆栈处于活动状态，则可能需要保留该文件夹。

您无法使用 WorkSpaces 应用程序控制台删除用户。而是必须使用服务 API 和 AWS CLI进行删除。有关更多信息，请参阅《Amazon WorkSpaces 应用程序管理指南》中的用户池管理。

教程：在 WorkSpaces 应用程序上使用 AWS Blu Age 开发者

本教程向您展示了如何在 Applications 上 WorkSpaces 访问 AWS Blu Age Developer 并将其与示例应用程序一起使用，以便您可以试用这些功能。完成本教程后，您可以在使用自己的应用程序的情况下，使用相同的步骤进行操作。

主题

步骤 1：创建数据库

在此步骤中，使用 Amazon RDS 创建托管 PostgreSQL 数据库，演示应用程序使用该数据库来存储配置信息。

1. 打开 Amazon RDS 控制台。

2. 选择数据库 > 创建数据库。
3. 选择标准创建 > PostgreSQL，保留默认版本，然后选择免费套餐。
4. 选择数据库实例标识符。
5. 对于凭证设置，选择管理 AWS Secrets Manager中的主凭证。有关更多信息，请参阅《Amazon RDS 用户指南》中的使用 Amazon RDS 和 AWS Secrets Manager管理密码。
6. 确保 VPC 与您用于 WorkSpaces 应用程序实例的 VPC 相同。您可以向管理员询问此值。
7. 对于 VPC 安全组，选择新建。
8. 将公共访问设置为是。
9. 保留所有其他默认值。审核这些值。
10. 选择创建数据库。

要使数据库服务器可以从您的实例访问，请在 Amazon RDS 中选择数据库服务器。在连接和安全性下，为数据库服务器选择 VPC 安全组。此安全组是之前创建的，其描述应与 RDS 管理控制台创建的安全组的描述类似。选择操作 > 编辑入站规则，选择添加规则，然后创建 PostgreSQL 类型的规则。对于规则来源，请使用安全组默认值。您可以开始在来源字段中输入来源名称，然后接受建议的 ID。最后，选择保存规则。

步骤 2：访问环境

在此步骤中，您将访问 “ WorkSpaces 应用程序” 上的 AWS Blu Age 开发环境。

1. 请联系您的管理员以获取访问您的 WorkSpaces 应用程序实例的正确方法。有关可能的客户端和配置的一般信息，请参阅《Amazon WorkSpaces 应用程序管理指南》中的 WorkSpaces 应用程序访问方法和客户端。请考虑使用本机客户端以获得最佳体验。
2. 在 “ WorkSpaces 应用程序” 中选择 “桌面”。

步骤 3：设置运行时

在此步骤中，您将设置 AWS Blu Age 运行时。您必须在首次启动时设置运行时，如果收到运行时升级的通知，则必须重新设置运行时。此步骤将填充您的 `.m2` 文件夹。

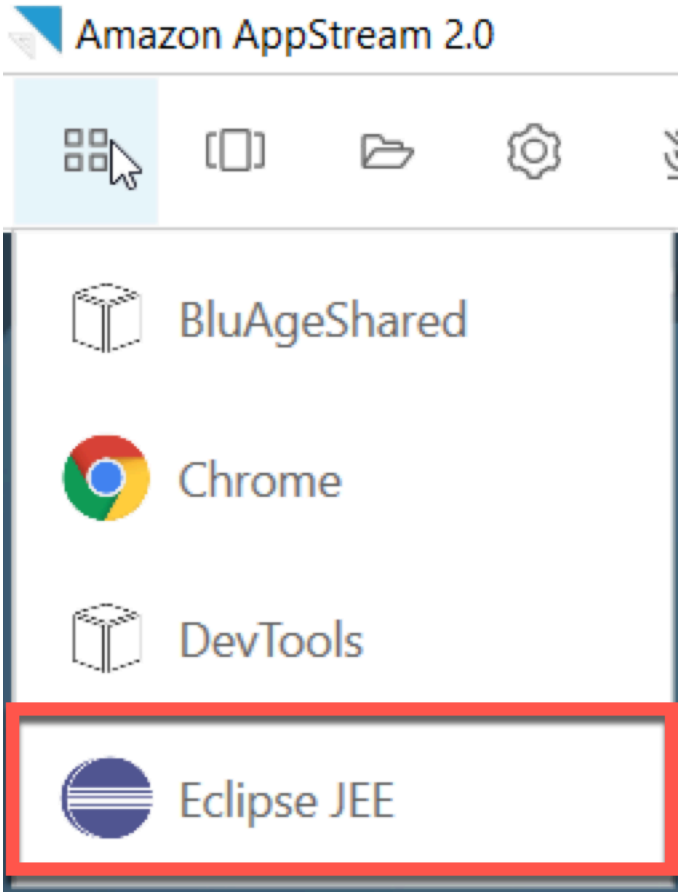
1. 从菜单栏中选择应用程序，然后选择终端。
2. 输入以下命令：

```
~/_install-velocity-runtime.sh
```

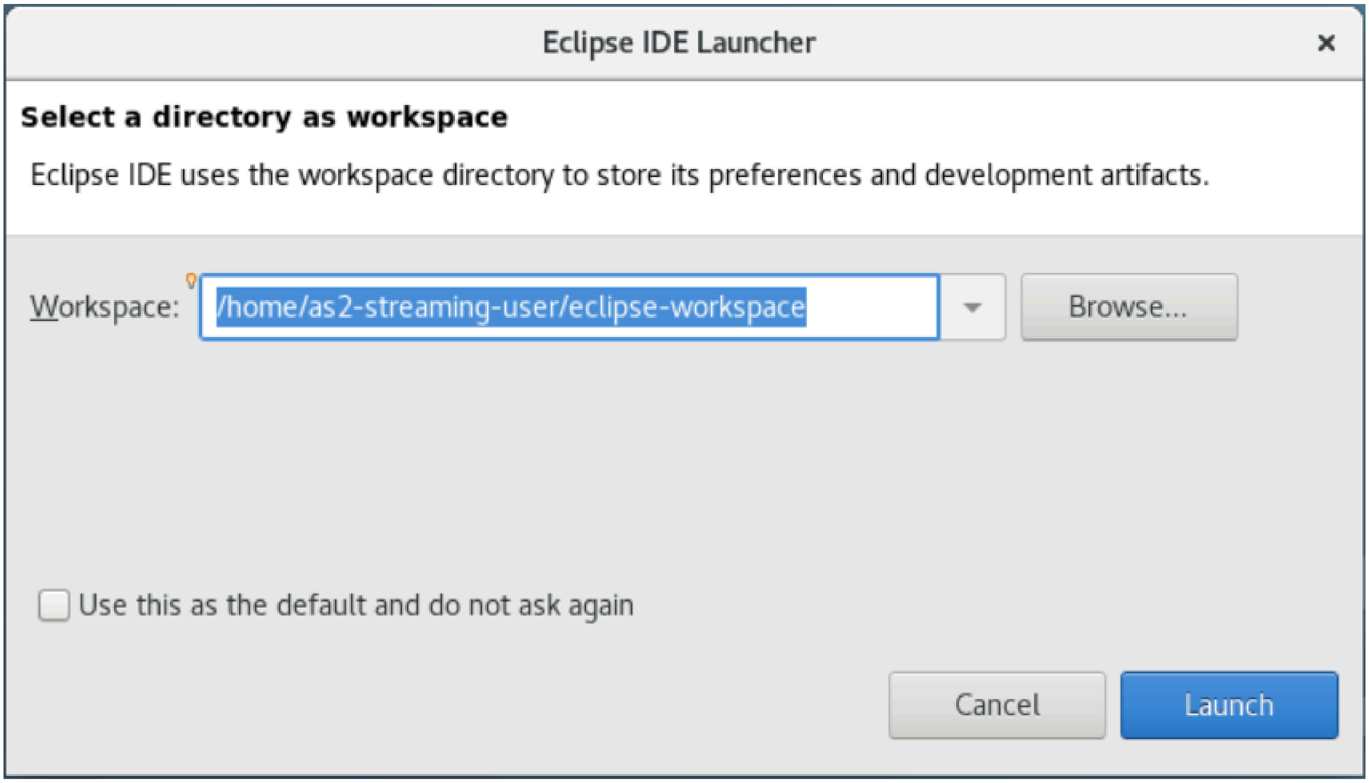
步骤 4：启动 Eclipse IDE

在此步骤中，启动 Eclipse IDE 并选择要创建工作区的位置。

1. 在 " WorkSpaces 应用程序" 中，选择工具栏上的 "启动应用程序" 图标，然后选择 Eclipse JEE。



2. 启动器打开后，输入要创建工作区的位置，然后选择启动。



您也可以选择通过命令行启动 Eclipse，如下所示：

```
~/eclipse &
```

步骤 5：设置 Maven 项目

在此步骤中，为 Planets 演示应用程序导入 Maven 项目。

1. PlanetsDemo将-pom.zip 上传到你的主文件夹。您可以使用本机客户端“我的文件”特征来执行此操作。
2. 使用 `unzip` 命令行工具提取文件。
3. 在解压缩后的文件夹中导航，然后在文本编辑器中打开项目的根 `pom.xml`。
4. 编辑 `gapwalk.version` 属性，使其与已安装的 AWS Blu Age 运行时相匹配。

   如果不确定安装的版本，请在终端中发出以下命令：

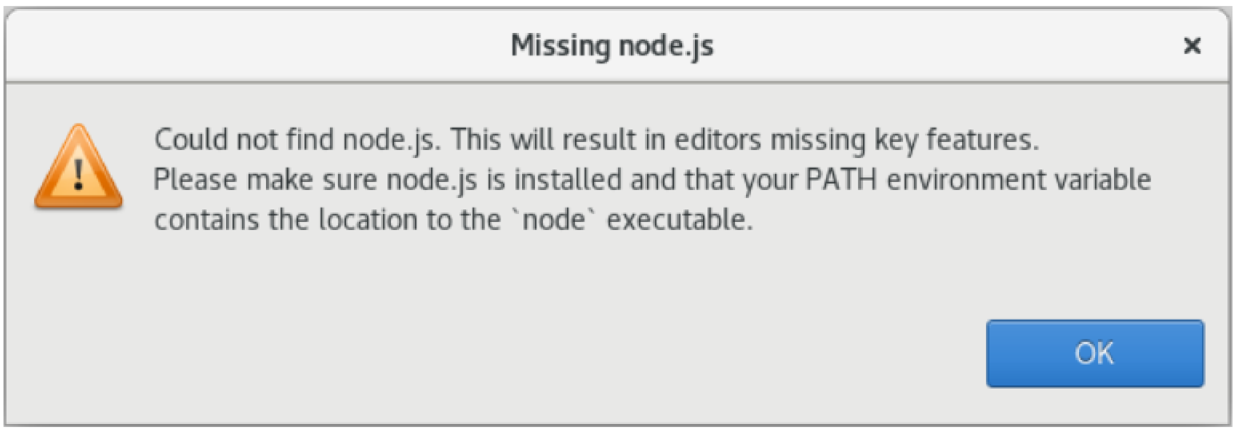
   ```
   cat ~/runtime-version.txt
   ```

   此命令会输出当前可用的运行时版本，例如 `3.1.0-b3257-dev`。

> **Note**
>
> 请勿在 gapwalk.version 中包含 -dev 后缀。例如，有效的值可以是 <gapwalk.version>3.1.0-b3257</gapwalk.version>。

5. 在 Eclipse 中，选择文件，然后选择导入。在导入对话框窗口中，展开 Maven，然后选择现有的 Maven 项目。选择下一步。
6. 在导入 Maven 项目中，提供提取的文件的位置，然后选择完成。

   您可以放心地忽略以下弹出窗口。Maven 下载 node.js 本地副本来构建项目的 Angular (*-web) 部分：

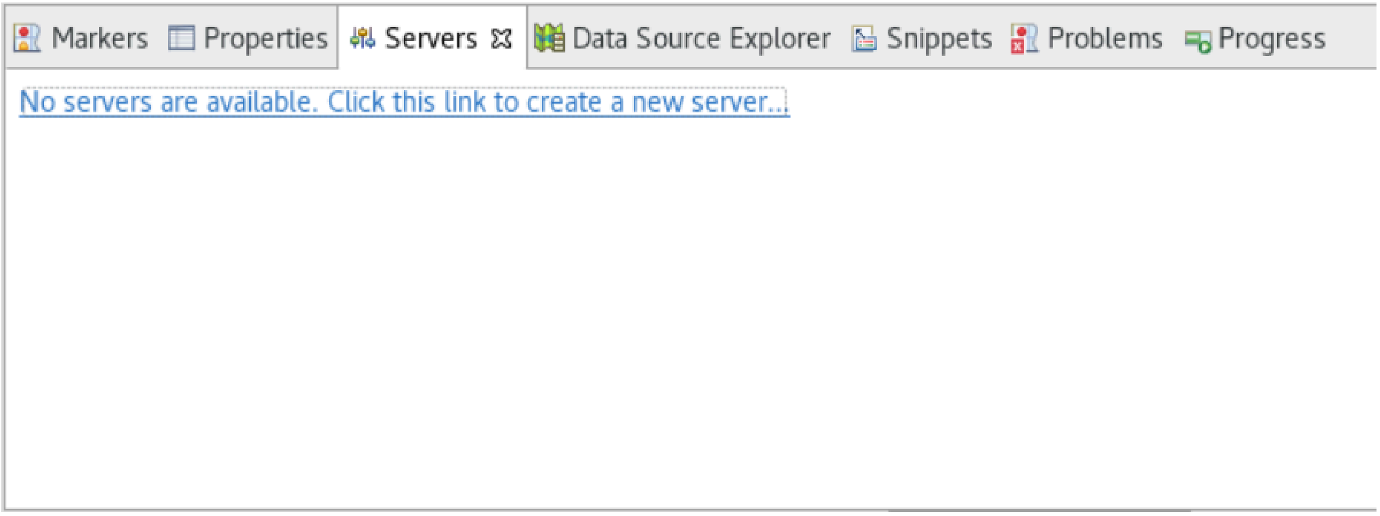
   

   等到构建结束。您可以在进度视图中查看构建进度。
7. 在 Eclipse 中，选择项目并选择运行方式。然后选择 安装。Maven 安装成功后，会在 PlanetsDemoPom/PlanetsDemo-web/target/PlanetsDemo-web-1.0.0.war 下创建 war 文件。

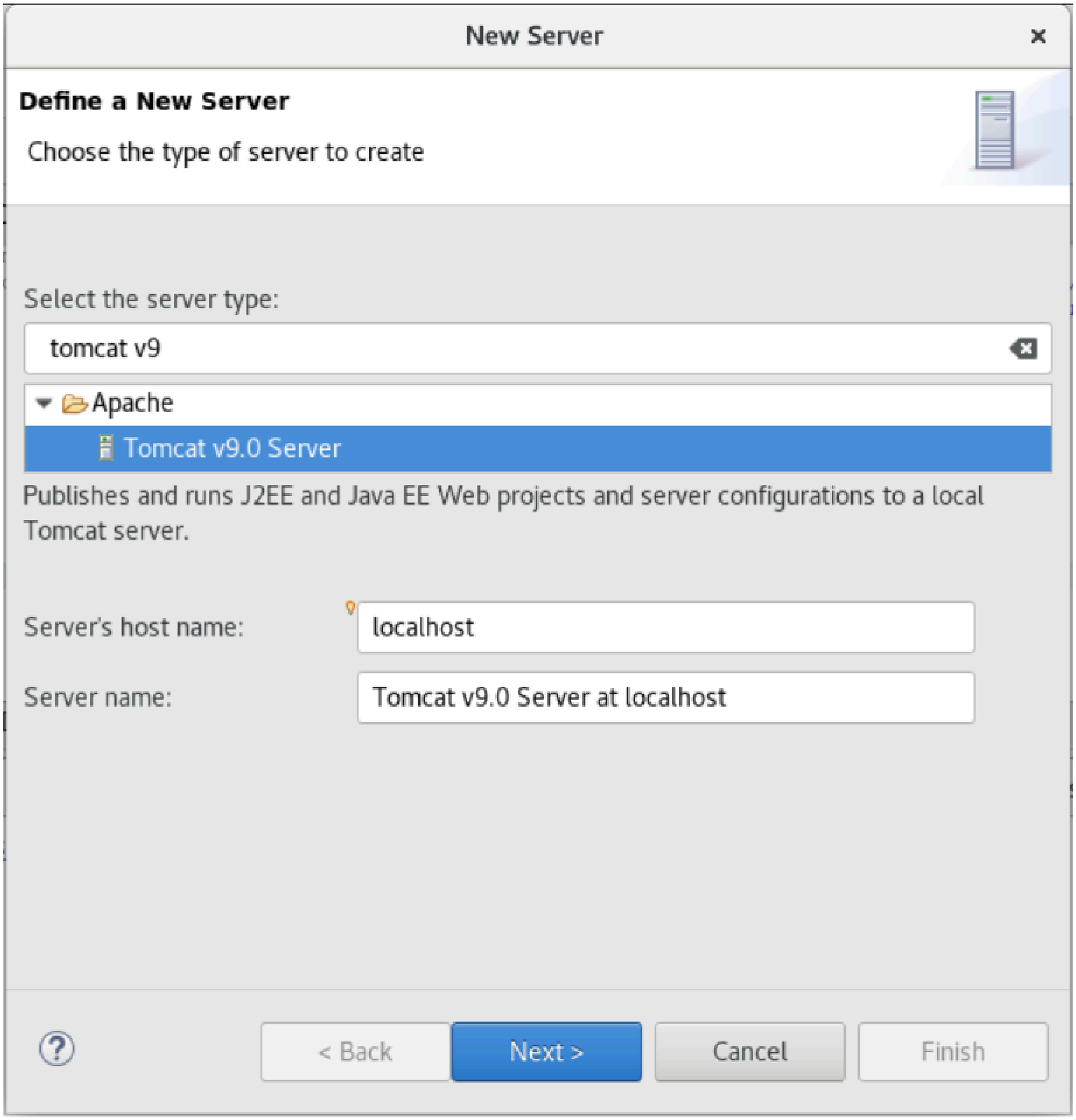
步骤 6：配置 Tomcat 服务器

在此步骤中，配置一个 Tomcat 服务器，用于部署和启动已编译的应用程序。

1. 在 Eclipse 中，选择窗口 > 显示视图 > 服务器以显示服务器视图：

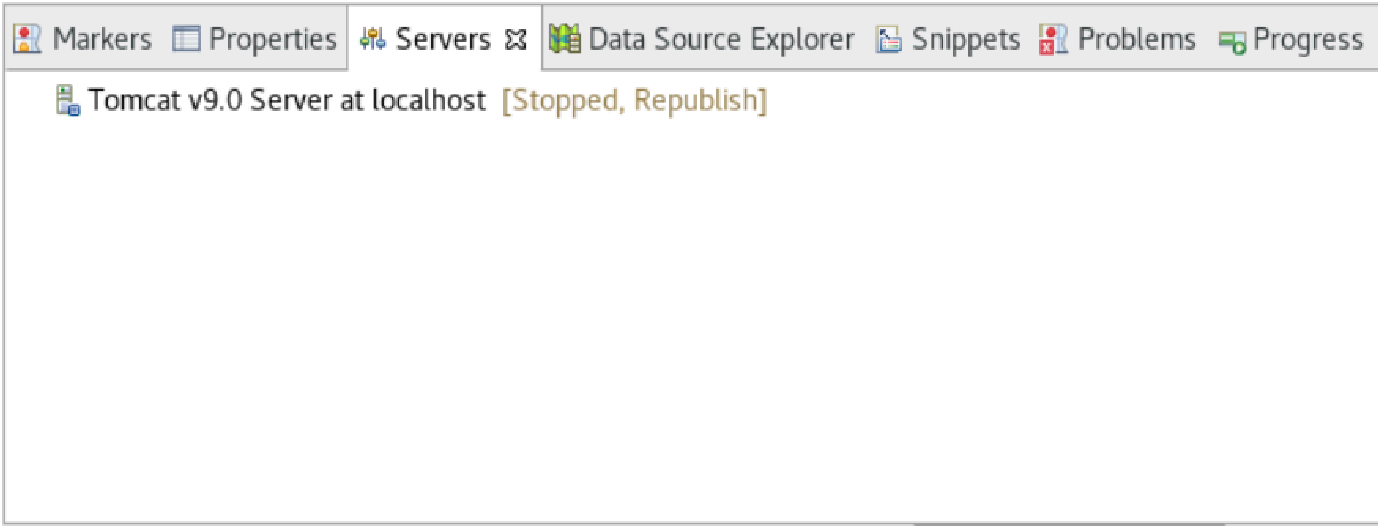


2. 选择没有可用的服务器。请单击此链接创建新服务器...。随后将出现新建服务器向导。在向导的选择服务器类型字段中，输入 tomcat v9，并选择 Tomcat v9.0 服务器。然后选择下一步。



3. 选择浏览，然后选择主文件夹根目录下的 tomcat 文件夹。保持 JRE 的默认值，然后选择完成。

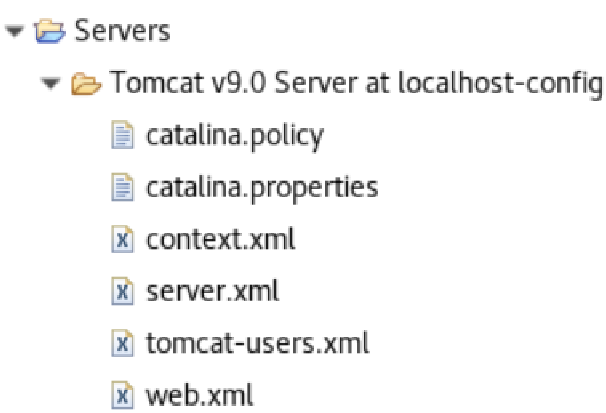
   服务器项目已在工作区中创建，Tomcat v9.0 服务器现在在服务器视图中可用。已编译应用程序将在此处部署和启动：



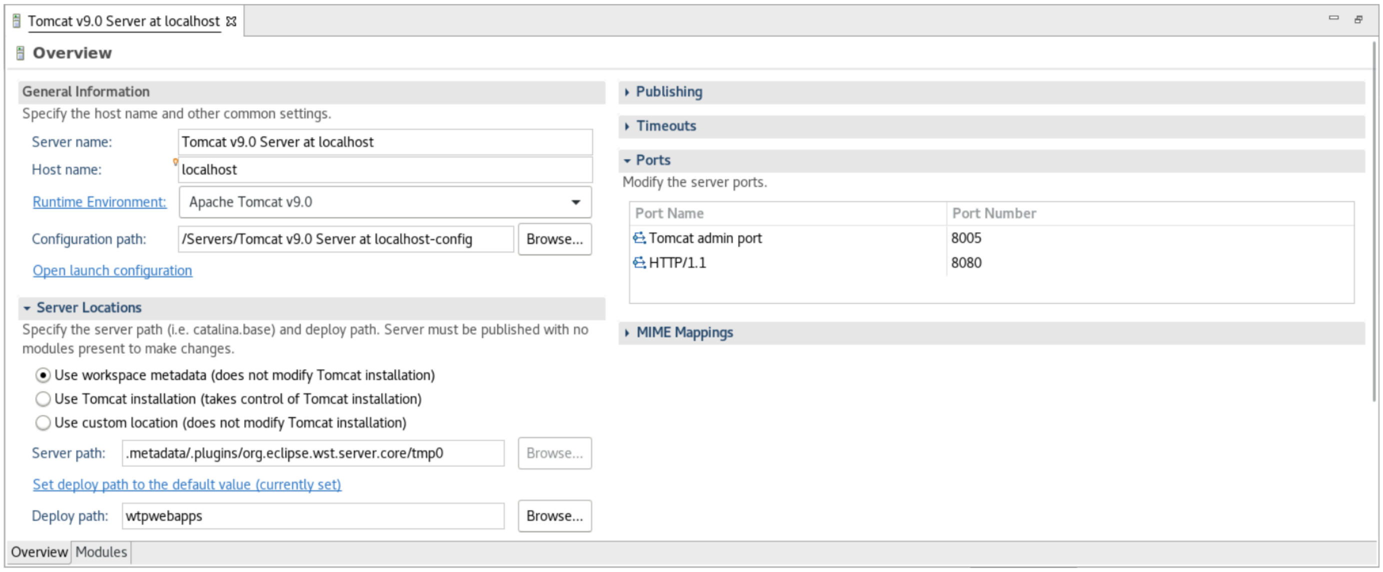
步骤 7：部署到 Tomcat

在此步骤中，将 Planets 演示应用程序部署到 Tomcat服务器，以便运行该应用程序。

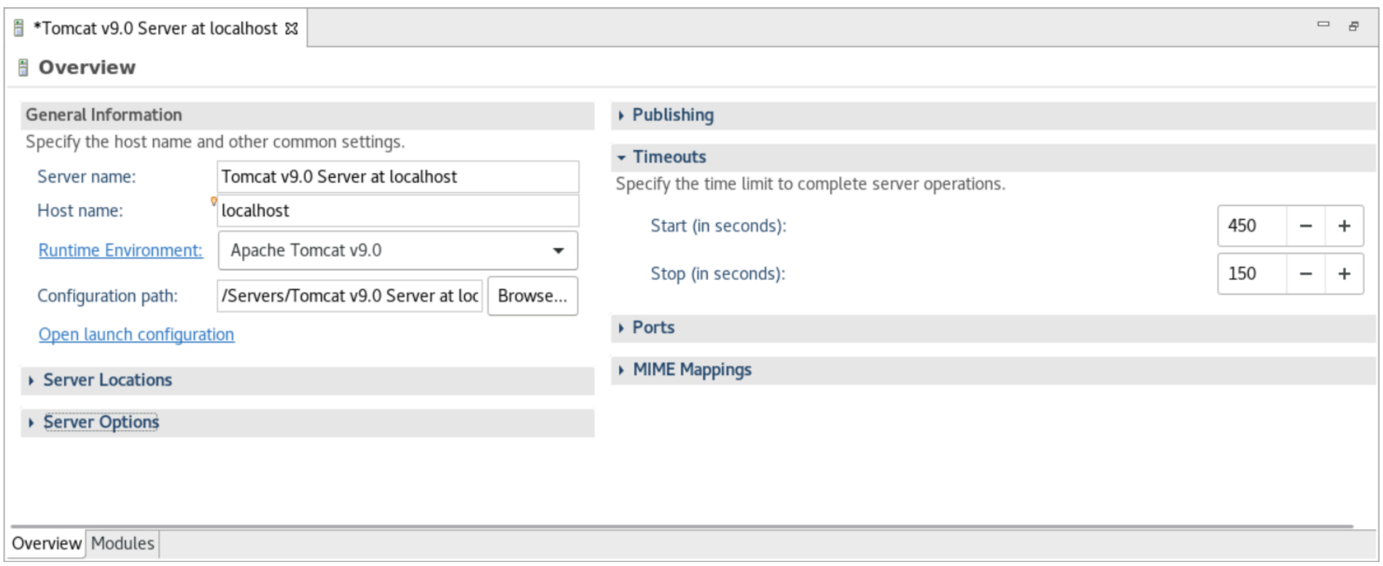
1. 选择 `PlanetsDemo-web` 文件并选择运行方式 > Maven 安装。再次选择 `PlanetsDemo-web` 并选择刷新，确保将经 npm 编译的前端正确编译为 .war 并被 Eclipse 注意到。
2. 将 [PlanetsDemo-runtime.zip]() 上传到实例，然后将文件解压缩到可访问的位置。这样可以确保演示应用程序可以访问所需的配置文件夹和文件。
3. 将 `PlanetsDemo-runtime/tomcat-config` 的内容复制到您为 Tomcat 服务器创建的 `Servers/Tomcat v9.0...` 子文件夹中：



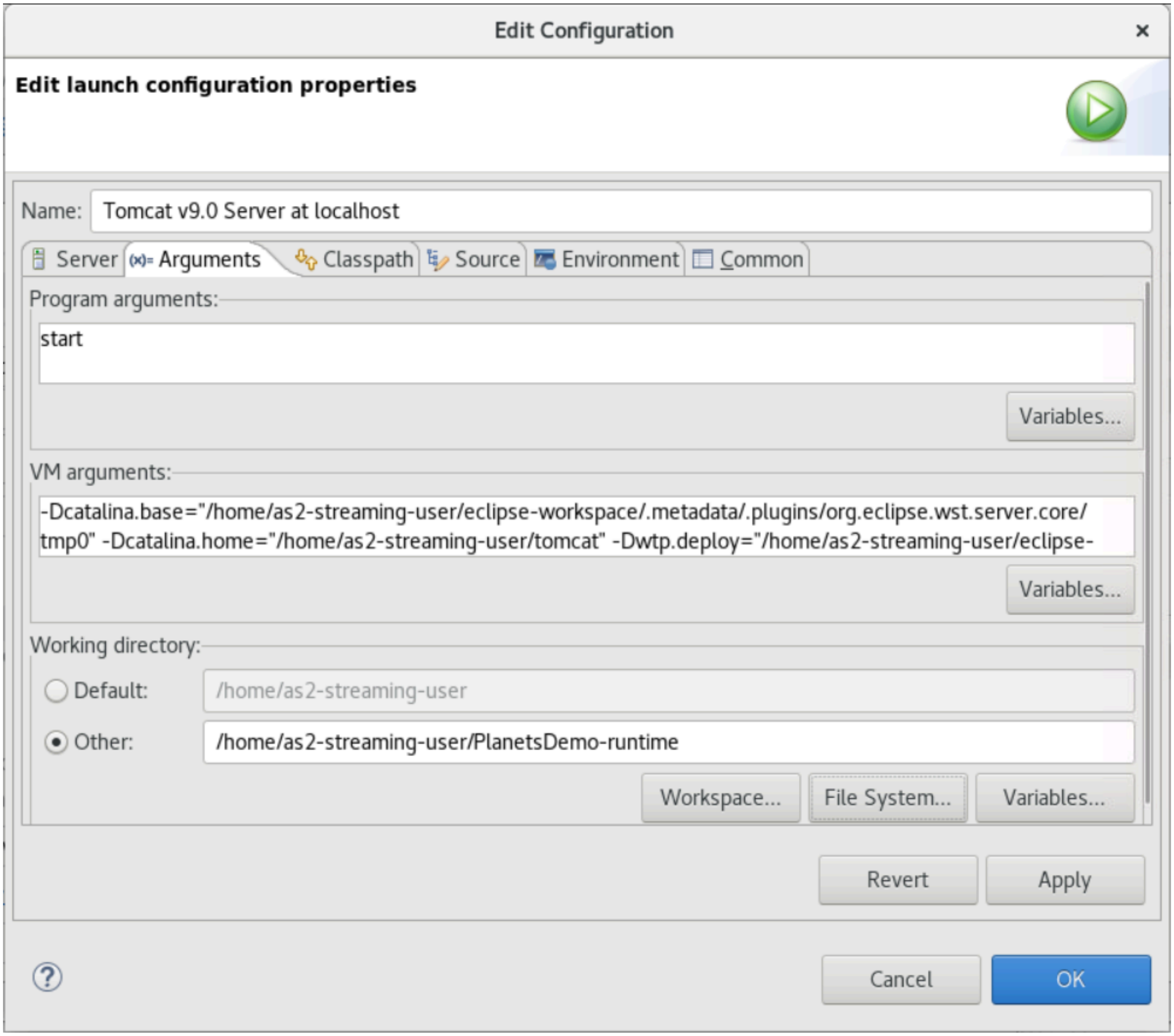
4. 在“服务器”视图中打开 `tomcat v9.0` 服务器条目。此时会显示服务器属性编辑器：



5. 在概览选项卡中，将“启动”的超时值增加到 450 秒，将“停止”的“超时”值增加到 150 秒，如下所示：



6. 选择打开启动配置。此时将显示向导。在向导中，导航到参数文件夹，然后在工作目录中选择其他。选择文件系统，然后导航到之前解压缩的 `PlanetsDemo-runtime` 文件夹。此文件夹应包含一个名为 config 的直接子文件夹。

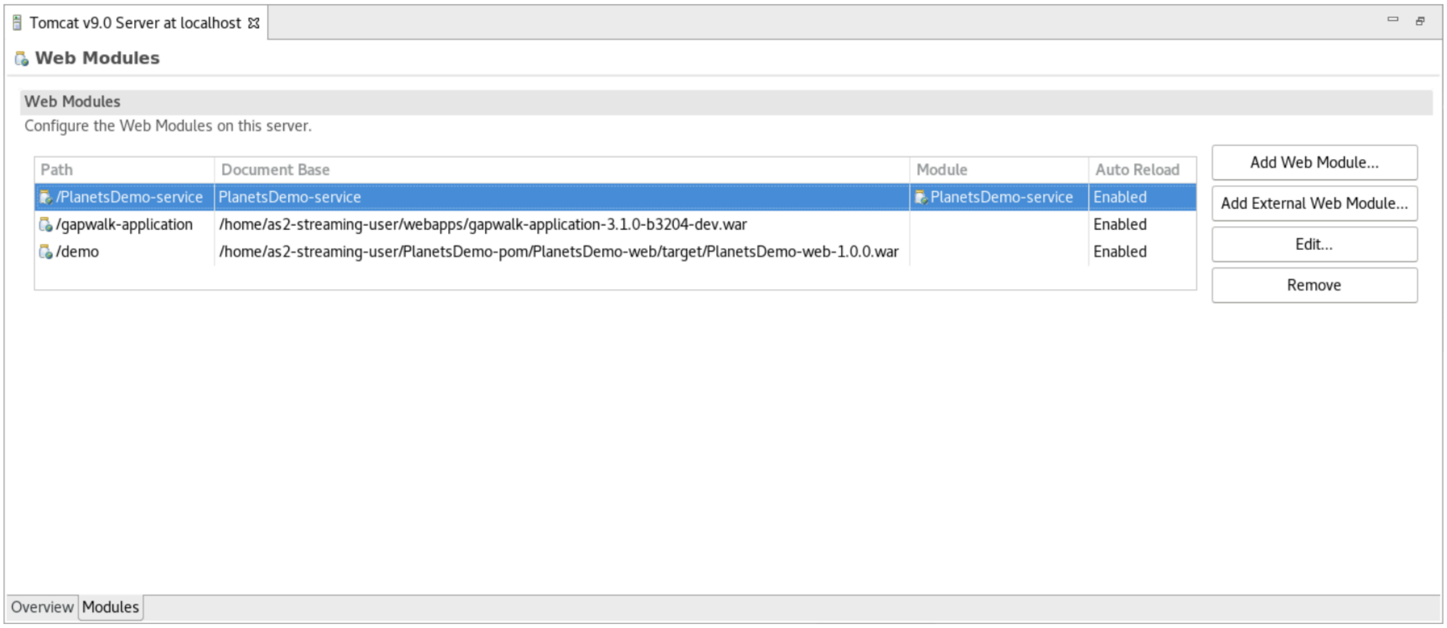


7. 选择服务器属性编辑器的模块选项卡并进行以下更改：

- 选择添加 Web 模块并添加 `PlanetsDemo-service`。
- 选择添加外部 Web 模块。随后显示添加 Web 模块对话框窗口。进行以下更改：
  - 在文档库中，选择浏览并导航至 `~/webapps/gapwalk-application...war`
  - 在路径中，输入 `/gapwalk-application`。
- 选择“确定”。
- 再次选择添加外部 Web 模块并进行以下更改：
  - 对于文档库，输入前端 .war 的路径（在 `PlanetsDemo-web/target` 中）

  - 在路径，输入 /demo
- 选择“确定”。
- 保存编辑器修改 (Ctrl + S)。

编辑器内容应类似于以下示例所示。



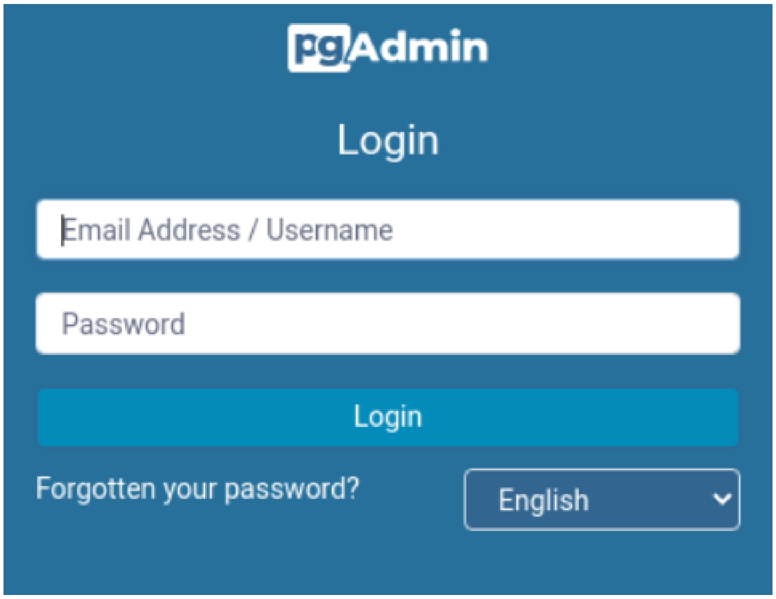
步骤 8：创建 JICS 数据库

在此步骤中，连接到您在 步骤 1：创建数据库 中创建的数据库。

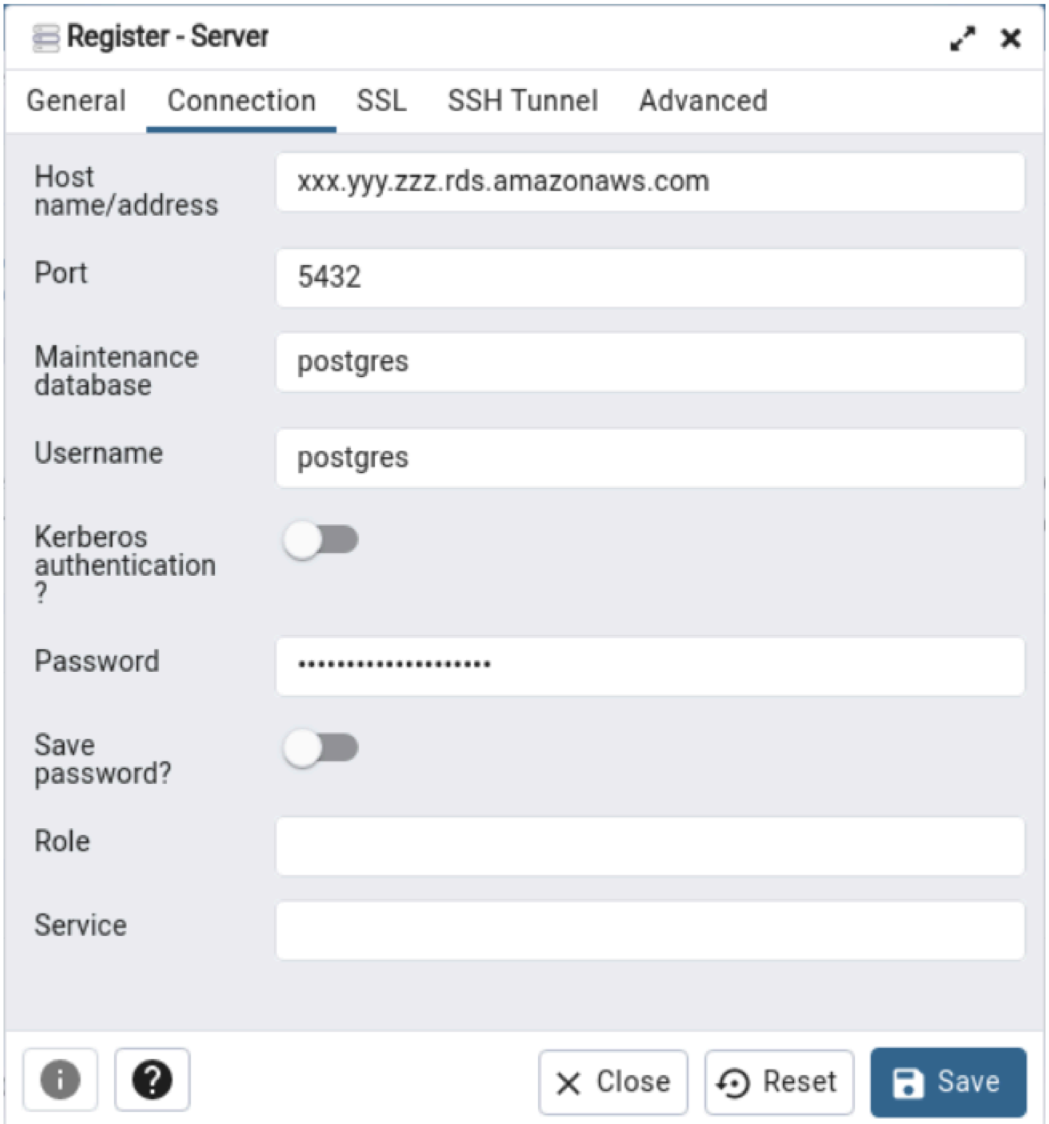
1. 在 WorkSpaces 应用程序实例中，在终端中发出以下命令进行启动pgAdmin：

```
./pgadmin-start.sh
```

2. 选择电子邮件地址和密码作为登录标识符。记录提供的 URL（通常是 http://127.0.0.1:5050）。在实例中启动 Google Chrome，将此 URL 复制并粘贴到浏览器中，然后使用您的标识符登录。



3. 登录后，选择添加新服务器，然后按如下方式输入先前创建的数据库的连接信息。



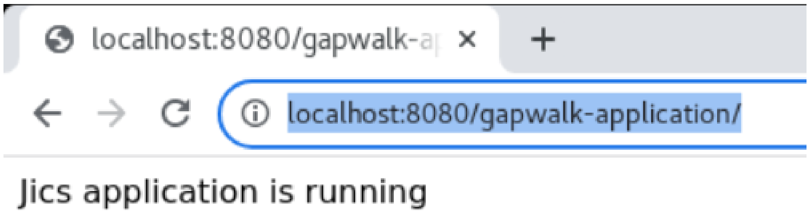
4. 连接到数据库服务器时，使用对象 > 创建 > 数据库并创建一个名为 jics 的新数据库。
5. 编辑演示应用程序使用的数据库连接信息。此信息在 `PlanetsDemo-runtime/config/application-main.yml` 中定义。搜索 `jicsDs` 条目。要检索 `username` 和 `password` 的值，请在 Amazon RDS 控制台中导航到数据库。在配置选项卡的主凭证 ARN 下，选择在 Secrets Manager 中管理。然后，在 Secrets Manager 控制台中，在密钥中选择检索密钥值。

步骤 9：启动和测试应用程序

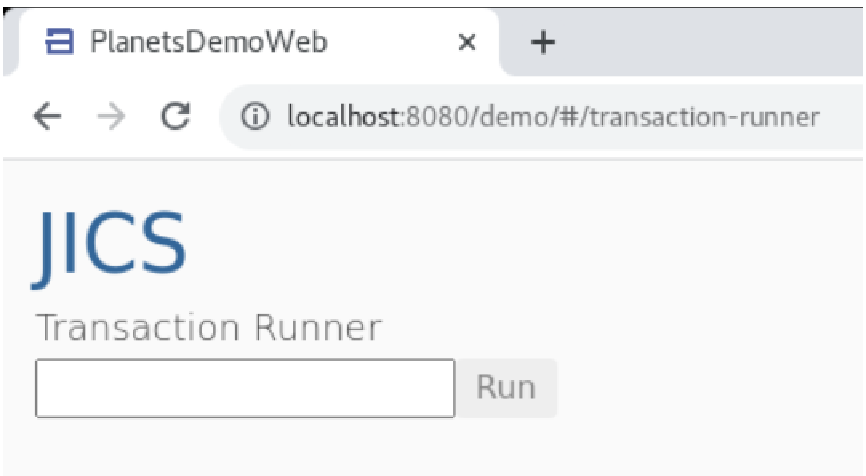
在此步骤中，启动 Tomcat 服务器和演示应用程序，以便对其进行测试。

1. 要启动 Tomcat 服务器和先前部署的应用程序，请在“服务器”视图中选择服务器条目，然后选择启动。随后将显示控制台，控制台上显示启动日志。

2. 在“服务器”视图中查看服务器状态，或者在控制台中等待服务器在 [xxx] 毫秒后启动消息。服务器启动后，检查 gapwalk-application 是否已正确部署。为此，请在 Google Chrome 浏览器中访问 http://localhost:8080/gapwalk-application URL。您应该看到以下内容。



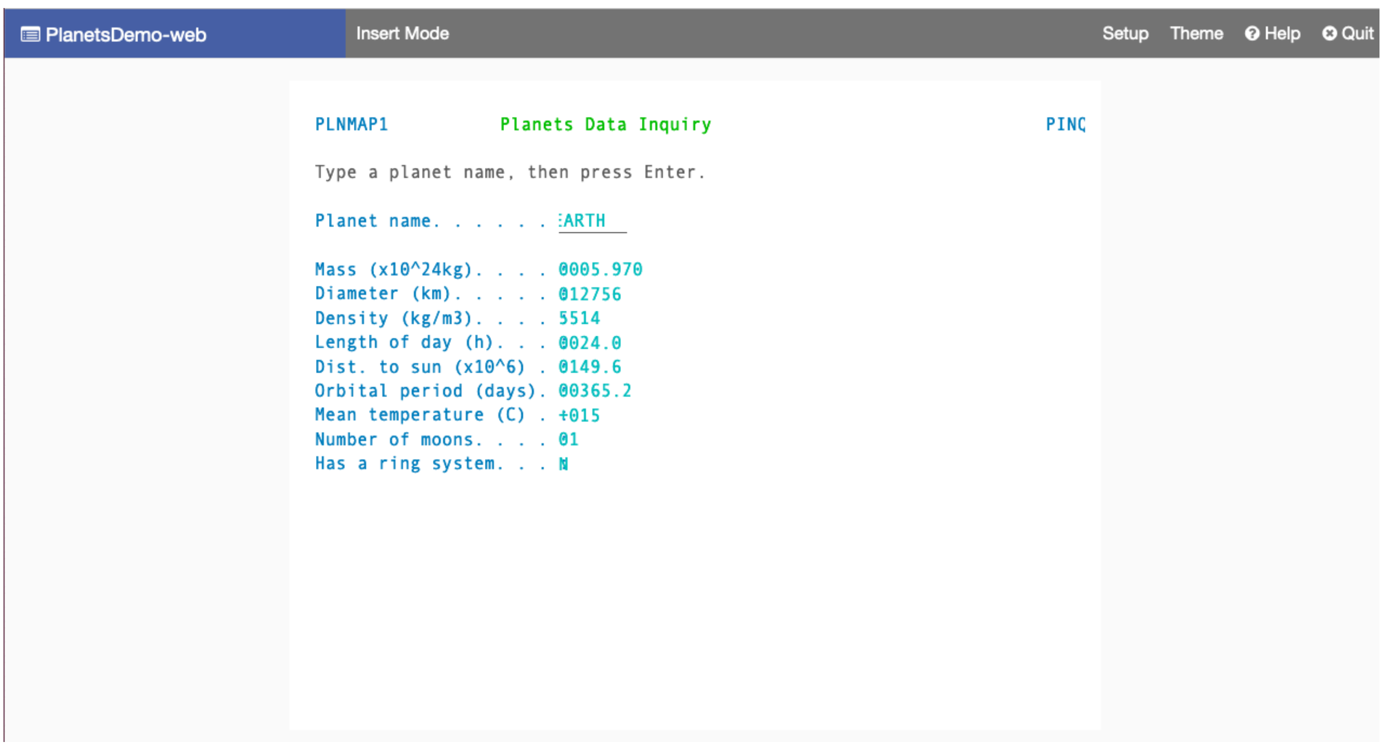
3. 从 Google Chrome 访问已部署的应用程序前端，网址为 http://localhost:8080/demo。随后会出现以下事务启动器页面。



4. 要启动应用程序事务，请在输入字段中输入 `PINQ`，然后选择运行（或按 Enter）。

   随后会显示演示应用程序屏幕。

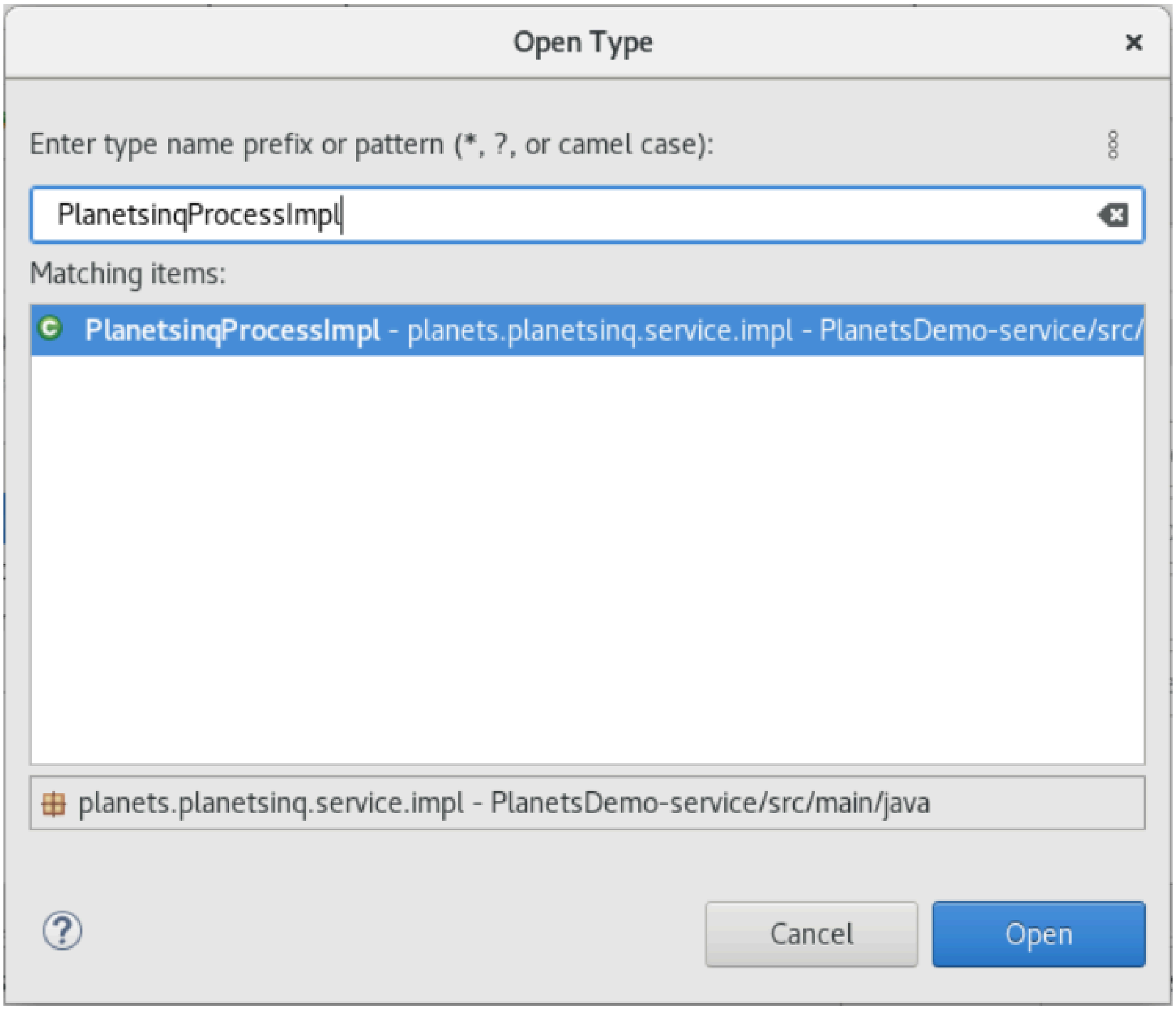
5. 在相应的字段中键入行星名称，然后按 Enter。



步骤 10：部署应用程序

在此步骤中，使用标准 Eclipse 调试特征进行测试。当您使用经过现代化造的应用程序时，这些特征可用。

1. 要打开主服务类，请按 Ctrl + Shift + T。然后输入 `PlanetsinqProcessImpl`。



2. 导航到 `searchPlanet` 方法，并在此处放置一个断点。
3. 选择服务器名称并选择在调试中重新启动。
4. 重复前述步骤，即访问应用程序，输入行星名称，然后按 Enter。

   Eclipse 将在 `searchPlanet` 方法中停止该应用程序。现在，您可以进行检查。

清理资源

如果您不再需要为本教程创建的资源，请将其删除，以免产生额外费用。完成以下步骤：

- 如果 Planets 应用程序仍在运行，请将其停止。
- 删除您在 步骤 1：创建数据库 中创建的数据库。有关更多信息，请参阅删除数据库实例。

AWS 蓝光时代常见问题解答

General

1. AWS Blu Age 重构功能的主要目的是什么？

   重构功能采用自动重构模式，使用现代语言和框架使用现代分布式应用程序将传统的整体代码重构为 Java。这种模式包括自动分析遗留代码，了解其功能，并将其转换为等效的现代代码，同时保留业务逻辑。该过程不仅包括使用自动化工具和流程对代码进行现代化改造，还包括对整个应用程序堆栈、依赖关系和基础架构进行现代化改造。该解决方案旨在加快现代化进程，同时保持功能等同性和性能。这包括转换应用程序代码以及相关的数据库和数据存储，同时实施云最佳实践和设计模式。

2. AWS Blu Age 支持哪些大型机应用程序？

   AWS Blu Age 目前支持使用 COBOL、PL/I、JCL（作业控制语言）编写的 IBM z/OS 大型机应用程序的现代化，并依赖于 CICS（客户信息控制系统）事务管理器、BMS（基本映射支持）屏幕、IMS MFS 屏幕、数据库、IMS DB2 数据库、平面文件、GDG（生成数据组）和 VSAM（虚拟存储访问方法）数据文件。

3. AWS Blu Age 可以对哪些大型机语言进行现代化改造？

   AWS Blu Age 将 COBOL 和 PL/I 代码转换为 Java、 JCLs Groovy、屏幕（BMS 或 MFS）转换为 HTML（使用 Sass）和 JavaScript （Angular 应用程序 — 目前不支持 React），从而实现了传统大型机应用程序向云原生架构的现代化。之所以选择这些技术，是因为其广泛采用、强大的生态系统和云原生功能。Angular 提供了一个现代的、响应式的用户界面层，它取代了传统的绿屏界面。它允许创建动态、用户友好的 Web 应用程序，这些应用程序可以在不同的设备和平台上进行访问。其基于组件的架构支持可维护和可扩展的前端开发。这种转型会产生遵循现代架构模式和最佳实践的分布式应用程序。

4. AWS Blu Age 如何在传统限制与云端优势之间取得平衡？

   AWS Blu Age 通过在引入云原生功能的同时保留关键业务逻辑和功能来实现平衡。它可确保现代化应用程序保持必要的传统业务逻辑，同时利用云可扩展性、敏捷性和现代运营实践。这种方法可以帮助组织保持业务连续性，同时获得云基础架构的好处。

5. 面向服务的架构在现代化应用程序中扮演什么角色？

面向服务的架构在将单片应用程序分解为更易于管理的模块化组件方面起着至关重要的作用。 AWS Blu Age 创建了面向服务和面向对象的应用程序，以提高可维护性和可扩展性。这种架构方法使组织能够实现更高的业务效率，并为未来可能采用的微服务做好准备。

6. 重构过程中包含应用程序堆栈的哪些方面？

   重构过程包括完整的软件堆栈：应用程序代码、依赖关系、数据库和基础架构（例如缓存选项、消息传递支持等）。它涵盖了传统编程语言、数据库系统、数据文件和相关基础设施组件的转换。这种全面的方法可确保应用程序的各个方面都实现连贯的现代化，从而实现完全变革的现代应用程序堆栈。

7. AWS Blu Age 现代化过程是否无需对现代化的 Java 应用程序进行任何测试或质量保证检查？

   不， AWS Blu Age 现代化过程并不能消除对现代化的 Java 应用程序进行任何测试或质量保证检查的需求。

8. AWS Blu Age JAGE JAC 代表什么？

   JAC 代表 JICS 管理控制台

9. 如何访问 AWS Blu Age 工具？

   AWS Blu Age 工具可通过 AWS 控制台通过 AWS 大型机现代化 (M2) 重构进行访问，功能访问权限取决于您的认证级别。从转换中心开始，评估源代码的 Java 自动重构。有关详细指导，请参阅 AWS Blu Insights 文档。现代化后，您可以使用运行时部署应用程序。有关更多信息，请参阅AWS 大型机现代化文档。

10. 如何调整项目规模（工作量和时间表）？

    有关这方面的更多信息，请参阅 AWS Blu Insights 估算值或咨询您的客户经理。

11. 维护 Java AWS Blu Age 迁移的解决方案是否有特定的要求？

    不，在维护 Java AWS Blu Age 迁移的解决方案方面没有具体要求。

12. AWS Blu Age 生成的代码的技术规格和兼容性是什么？

    AWS Blu Age 生成的代码具有特定的技术特征和广泛的兼容性。虽然它不支持 JPA，但它对外部化查询使用直接 SQL 执行。该代码依赖运行时特定的库来实现功能等效、Web 服务生成和 MQ 实现。生成的代码可以导入到任何 Java IDE 中进行开发、测试、构建和部署，但必须相应地导入所需的库。虽然默认情况下，Maven 与用于构建过程 AWS 的大型机现代化服务集成，但可以通过在转换后修改打包格式来使用像 Gradle 这样的替代工具。该平台在开发工具和源代码控制方面提供了灵活性，并为管理代码的开发团队提供了培训。有关更多信息，请参阅 AWS Blu Age Runtime 高级架构。

AWS 蓝光时代运行时间

1. 在哪里可以找到有关 AWS Blu Age Runtime 的信息？

   请参阅设置 AWS Blu Age Runtime 文档运行时，其中详细介绍了设置流程、入门、检索工件、部署等。

2. AWS Blu Age JAR 依赖项是否已上传到客户端的 Maven 存储库进行本地开发？

   可以 EC2 使用 AMI 导入库，该AMI可用于配置开发、测试和生产环境。将向团队提供有关生成的应用程序代码 maintain/enhance 的培训和支持。有关更多信息，请参阅 AWS Blu Age Runtime 高级架构。

3. 分布式 AWS Blu Age Runtime jars 中 “Gapwalk” 一词指的是什么？

   有关 Gapwalk 的信息，请参阅 AWS Blu Age 运行时工件。

4. 如何申请访问 AWS Blu Age Runtime？

   可通过 “Blu Age工具箱” 访问运行时Blu Insights。

5. AWS Blu Age 重构后的应用程序支持的运行时有哪些？

   AWS Blu Age提供单个 Runtime，以满足现代化旅程的不同阶段和运营需求，请参阅 AWSBlu AgeRuntime。

6. 什么时候使用 AWS Blu Age Runtime？

   AWS Blu Age 运行时是支持执行经过重构的 AWS Blu Age 应用程序所必需的。在基于 AWS Blu Age 的重构项目中，需要运行时来测试重构后的应用程序。重构项目结束后，还需要运行时来在生产环境中维护、测试和运行经过重构的 AWS Blu Age 应用程序。

7. 如何 AWS 分发 AWS Blu Age Runtime 的新版本？

   AWS Blu Age运行时版本可通过Blu Age工具箱访问。请参阅 AWSBlu Age 发布说明。

8. AWS Blu Age 运行时的新主要版本和次要版本多久发布一次？

   有关更多信息，请参阅AWS 大型机现代化组件生命周期。

9. 如何为 AWS Blu Age Runtime AWS 提供支持？

   Support 是通过提供的 AWS 支持，问题通过提票来解决，并且适用标准 SLA。有关更多信息，请参阅AWS 大型机现代化组件生命周期。

10. AWS 大型机现代化 AWS Blu Age 运行时需要什么？

AWS Blu Age Runtime 包含工具箱库，用于加快现代化、促进云集成以及提高代码质量和可维护性。它还通过促进传统架构和云架构之间的过渡来实现更高的现代化自动化。运行时支持使用 java 习惯用法处理传统动词和数据结构内存表示形式。它允许基于面向对象的编程技术构建现代化应用程序，并能够重现传统的控制流。它使用诸如 Amazon Aurora 之类的关系数据库对传统 VSAM 数据集或 IMS 分层数据库支持进行现代化改造。它为传统系统实用程序（IDCAMS、IEBGENER、DFSORT等）和传统交易管理系统（CICS、IMS）提供了Java替代品。它通过亚马逊的缓存 ElastiCache 和对 AWS 消息解决方案（SQS、Kinesis）的支持来促进云集成。

11. AWS Blu Age Runtime 是否支持非 x86 计算机架构？

    目前， AWS Blu Age Runtime 仅支持基于 x86 的计算机架构和计算。 AWS Blu Age Runtime 不支持基于 ARM 和基于 Graviton 的计算。

12. 客户如何随时了解 AWS Blu Age Runtime 版本，包括新版本的通知以及版本历史记录和发行说明的访问权限？

    AWS Blu Age Runtime 的新版本已上传到我们的官方发布页面。我们建议您定期查看此页面，最好是每 3 个月查看一次，以了解最新版本和更新。关于版本历史记录和发行说明的访问权限，可用性取决于每个主要版本的 end-of-life (EOL) 日期。有关 EOL 日期、版本升级计划和历史信息访问权限的详细信息，请参阅 AWS Blu Age 生命周期。

13. AWS Blu Age Runtime 高级架构的主要组成部分是什么？

    AWS Blu Age 运行时架构包括两种主要组件类型。首先是存储在共享文件夹（应用程序服务器类加载器可访问）中的 Java 库（jar 文件），这些库提供传统结构和语句支持。其次是包含基于 Spring 的应用程序的 Web 应用程序（战争文件），这些应用程序为现代化程序提供框架和服务。运行时还包括：一个程序注册表，用于收集所有用于调用和跨程序调用的程序，以及一个收集所有现代化作业脚本的脚本注册表。这些组件协同工作，为现代化应用程序提供统一的基于 REST 的入口点和执行框架。运行时和现代化应用程序一起部署在应用程序服务器（例如 Tomcat）中。

14. 如何配置存放 AWS Blu Age Runtime 工件的共享文件夹？

    AWS Blu Age Runtime 工件 (jar) 必须收集在应用程序服务器类加载器可以访问的共享文件夹中。对于 tomcat 服务器，配置是通过修改名为 catalina.properties 的常规配置文件进行的。例如，如果您在 tomcat 文件夹中将共享文件夹创建为名为 “shared” 的文件夹，则需要修改 catalina.properties 中的 common.loader 条目，以使 tomcat 类加载器可以访问该共享文件夹，如下所示：

```
common.loader="${catalina.base}/lib","${catalina.base}/lib/*.jar","${catalina.home}/
lib","${catalina.home}/lib/*.jar","${catalina.home}/shared","${catalina.home}/shared/
*.jar"
```

15.AWS Blu Age Runtime 如何处理无状态和会话管理？

AWS Blu Age Runtime 通过多种机制实现无状态和会话管理。对于 HTTP 会话，它使用基于 Cookie 的标识以及外部缓存存储空间来存储用户上下文。会话可以存储在各种数据存储中，包括 Amazon ElastiCache、Redis 集群或内存地图。无国籍状态设计可确保大多数非临时状态存储在外部通用的 “单一事实来源” 中，从而实现高可用性和横向扩展。这种方法与负载平衡和共享会话相结合，允许在多个节点之间分发面向用户的对话。

16.Web 应用程序在 Bl AWS u Age 运行时环境中扮演什么角色？

AWS Blu Age Runtime 中的 Web 应用程序提供多种关键功能。它们提供可重现传统环境和事务监视器（例如 JCL 批处理、CICS、IMS）的执行框架。它们通过提供基于 REST 的入口点，`gapwalk-application.war`用于触发和控制交易、程序和批次。此外，它们还提供操作系统提供的程序和专用 “驱动程序” 程序的仿真，传统应用程序依赖这些程序通过MFS访问IMS DB或用户对话框等服务。

17.如何在 AWS Blu Age Runtime 中注册和管理程序？

AWS Blu Age Runtime 中的程序是通过服务器启动期间填充的ProgramRegistry 系统注册的。每个程序都实现程序接口，并标记为 Spring 组件。程序使用其标识符进行注册，如果一个程序有多个标识符，则可能有多个条目。注册过程是自动的，并记录在 Tomcat 日志中。ProgramRegistry使其他程序和脚本能够定位和调用注册的程序，从而保持现代化系统的模块化和互连性。

18.如何在 AWS Blu Age 运行时应用程序中管理配置？

AWS Blu Age Runtime 中的配置是使用 Spring Boot 框架功能通过 YAML 文件管理的。使用了两个主要的配置文件：用于框架配置的 application-main.yml 和用于客户端特定的选项。`application-profile.yml`系统遵循Spring的优先级逻辑，允许通过各种方式覆盖配置。可以通过 JNDI 为数据库和命令行参数提供其他配置，从而提供配置管理的灵活性。记录器配置是使用 logback xml 配置文件完成的。

19.秘密管理员在 Bl AWS u Age 运行时配置中扮演什么角色？

AWS Blu Age Runtime 中的机密管理器可以保护敏感的配置数据，例如数据库凭据和 Redis 缓存密码。它们允许将关键数据存储在 AWS 机密中，并在 YAML 配置文件中引用它们。该系统支持不同类型的密钥，包括自动填充所有相关字段的数据库密钥和受密码保护的资源的单一密码密钥。这种方法通过将敏感数据与应用程序配置分开来增强安全性。

20开发者如何编写与 AWS Blu Age Runtime 兼容的自己的程序？

开发人员可以通过实现程序接口并遵循特定模式来创建与 AWS Blu Age Runtime 兼容的程序。该程序必须声明为 Spring 组件，实现所需的方法，并在中正确注册 ProgramRegistry。开发人员需要创建配套上下文和配置类，处理程序标识符，并确保与 Spring 框架正确集成。实现应遵循程序结构和执行的 AWS Blu Age 运行时惯例。

21AWS Blu Age Runtime 如何处理程序执行错误？

AWS Blu Age Runtime 通过多种机制处理程序执行错误。对于批处理作业，它会在任务执行详细信息中捕获执行状态、退出代码和详细的错误信息。错误处理包括特定的退出代码（-1 表示技术错误，-2 表示服务程序故障）和 Tomcat 日志中的详细日志记录。可以将系统配置为在运行时异常时回滚事务，并提供错误通知和恢复选项。错误详细信息可通过 REST 端点访问，以便进行监控和故障排除。

22.有哪些 AWS Blu Age 运行时监控功能可用于批处理作业？

AWS Blu Age Runtime 通过各种端点为批处理作业提供监控功能。它跟踪任务执行状态、 start/end 时间、执行模式和详细结果。系统提供端点，用于列出触发的脚本、检索任务执行详细信息和监控当前正在运行的作业。Metrics 的端点提供 JVM 统计信息、会话计数和详细的批处理执行指标。该平台还支持对监控数据进行分页和基于时间的过滤。

23.如何跟踪和管理 AWS Blu Age 运行时任务执行状态？

任务执行状态通过全面的状态系统进行跟踪，该系统包括完成、已触发、正在运行、KILLED 和 FAILED 等状态。每个任务执行都会收到一个用于跟踪的唯一标识符，并维护详细的执行信息，包括开始时间、结束时间、调用方信息和执行结果。系统提供 REST 端点，用于查询作业状态、管理正在运行的作业和检索执行历史记录。状态信息保留在服务器内存中，可以根据时间进行清除，以便进行资源管理。

24AWS Blu Age Runtime 如何处理外部系统交互？

运行时通过各种机制处理外部系统交互，包括用于服务集成的 REST 端点、对消息队列（SQS、RabbitMQ、IBM MQ）的支持以及数据库连接选项。它通过专用组件提供对旧系统交互的仿真，支持 SSL/TLS 安全通信，并包括用于处理外部文件系统的功能。该系统还支持与外部身份验证提供商集成，并且可以配置为与各种第三方服务进行交互。

25.在 AWS Blu Age Runtime 中如何处理身份验证？

AWS Blu Age Runtime 支持多种身份验证方法， OAuth2 其中是主要机制。它可以与 Amazon Cognito 或 Keycloak 等身份提供商集成。身份验证配置通过名为 application-main.yml 的主配置文件进行管理，可以在其中定义安全设置、身份提供者和身份验证方法。该系统支持 XSS 保

护、CORS、CSRF 等功能，并且可以针对全局安全和特定端点安全进行配置。为了进行开发，还提供了具有默认超级管理员凭据的本地身份验证系统。

26. AWS Blu Age Runtime 如何确保高可用性？

AWS Blu Age Runtime 通过多种机制确保高可用性。它通过将非临时状态存储在外部共享存储中来实现无状态，从而使多个应用程序实例能够协同工作。该系统支持负载平衡和共享会话，允许将请求分发到多个节点。对于数据存储，它可以利用高可用性的数据库和缓存系统。该架构支持自动故障转移，可以跨多个可用区进行部署，以提高可靠性。

27. 哪个组件用于通过 AWS Blu Age 应用程序重现 CICS 分布式事务？

AWS Blu Age Runtime 提供了一个专用端点，允许将现有 JICS 事务作为全局事务的一部分调用（XA 支持）。底层的两个阶段提交支持依赖于 Atomikos 软件组件。

28. 用于定义特定程序行为的类的 AWS Blu Age 名称是什么？

每个程序都绑定到一个专用的配置类，该类允许指定程序的特定行为。有关命名和位置约定的更多信息，请参阅现代化应用程序的AWS Blu Age 结构

29. 哪种编码的字符序列顺序如下：空格、小写字符、大写字符、数字？

属于 EBCDIC 变体家族的字符集（例如 CP1 047 等 CP297）。

30. AWS Blu Age Runtime 的定价维度是多少？

AWS 大型机 Modernization-core-hours（参见AWS 大型机现代化定价）。

31. 用于通过 HTTP 将原始数据传递到程序端点的机制是什么？

以 Base64 编码的字符串。

32. 用户如何启动批处理作业？

使用对其中一个专用批处理端点的 HTTP 调用（参见批处理端点文档页面）。

33. 哪个 AWS Blu Age 运行时端点是主要 Web 前端应用程序的主要入口点？

```
/transaction
```

34. AWS Blu Age JICS 代表什么？

AWS Blu Age JICS 是用于支持 CICS 资源现代化的运行时组件。资源定义存储在专用的数据存储中。要对其进行管理，请使用 REST API 或 JICS 应用程序控制台。有关信息，请参阅在 AWS Blu Age 中管理 JICS 应用程序控制台。

35. 有哪些 AWS Blu Age 运行时缓存机制可用？

AWS Blu Age Runtime 支持多种缓存机制，包括 Redis 和 EhCache. 建议在生产环境中使用 Redis，提供跨多个节点的共享永久缓存。 EhCache 适用于带有嵌入式易失性本地缓存的独立部署。系统支持缓存各种组件，包括Blusam数据、会话信息、JICS 资源和临时存储队列。缓存配置可以根据不同的用例和性能要求进行自定义。

36.我们如何估算 AWS 大型机现代化 AWS Blu Age 运行时部署的价格？

AWS 根据客户的需求和目标架构向其提供估算值。

37.AWS 大型机现代化 AWS Blu Age 运行时的价格是多少？

AWS 大型机现代化为 AWS Blu Age 提供了两种定价模式：一种是托管运行时、计算资源、内部存储和自动化，另一种是仅涵盖 AWS 运行时本身的非托管Blu Age运行时选项。对于 AWS 部署，两者都使用定 pay-as-you-go价结构。有关最详细 up-to-date的定价信息，建议查阅 AWS 大型机现代化官方定价页面。

38.如果我们需要在支持的运行时中未列出的基础架构上部署 AWS Blu Age 重构后的应用程序，该怎么办？

如果您需要在支持的运行时中未列出的基础架构上部署 AWS Blu Age 重构后的应用程序，则有多种选项可供选择。首先，检查您的基础设施是否与 Amazon EKS Anywhere 或其他容器编排平台等现有部署选项兼容。如果是这样，你可以使用 AWS Blu Age Runtime。对于不兼容的基础架构，我们建议您咨询 AWS 大型机专家，以探索定制解决方案或潜在的适应方案。您也可以提交产品功能申请 (PFR)，以获得扩展的基础架构支持。请联系您的 AWS 代表，讨论您的具体需求和适合您环境的最佳方法。

39.AWS Blu Age Runtime 是如何获得许可的？ 它是开源吗？

AWS Blu Age Runtime 不是开源的。它是作为云原生服务分发 AWS 的 IP。有两个部署选项：

a. AWS Blu Age Manag ed，运行时部署到专用的 AWS 托管服务中，利用完全预先配置且随时可以部署的环境，无需设置或管理。
b. AWS Blu Age Non Manag ed，它可以部署到你自己的定制 AWS 架构中，基于 Amazon EC2 或 Amazon ECS/AWS Fargate，你必须自己配置和设置。这两个选项都会产生运行费，这些费用包含在提供给您的项目估算中。由于这是一项具有 支持 访问权限的托管服务，因此您不需要源代码。有关定价的更多详细信息，请参阅AWS 大型机现代化定价页面。

40.如何管理 AWS Blu Age 框架和库的变更和升级？

AWS Blu Age 框架和库通过定期的代码生成和部署过程进行更新。这些更新作为 AWS 大型机现代化生命周期的一部分进行管理，其中包括版本升级和 AWS Blu Age 团队或认证合作伙伴的支持。有关版本控制、升级过程和支持时间表的详细信息，请参阅AWS 大型机现代化生命周期文档。

41. Blu Age Runtime 使用的工具（Tomcat、Postgres、MQ 等）和依赖项（Spring、Angular 等）的支持版本是什么？ AWS

    详情请参阅发行说明。

42. 在 BAC 和 JAC 的背景下，“独立” 是什么意思？

    独立版是指 BAC（管理控制台）和 JAC（JICS Blusam 管理控制台）的特殊打包和部署模式，允许这些 Web 应用程序在自己的 Tomcat 服务器中独立运行，与您的现代化应用程序分开。BAC 和 JAC 独立版本可用。`aws-bluage-webapps-x.y.z.zip`BAC 和 JAC 非独立版本可在文件夹下找到`gapwalk-x.y.z.zip`。`webapps-consoles`参见 AWS Blu Age 运行时工件。

数据

1. 关于传统数据库的现代化，现代化应用程序有哪些数据库选项可用？

   现代化的应用程序可以使用多种现代数据库选项，包括：PostgreSQL、Amazon Aurora、适用于 PostgreSQL 的 RDS、Oracle 数据库、MS-SQL 和 IBM Db2。这些选项使您可以灵活地根据特定要求选择最合适的数据库系统，同时利用现代数据库管理系统和云原生功能的优势。

2. IBM Db2 向 Postgres DDL 的z/OS转型覆盖范围是多少？

   完全转换（包括数据库约束）。

3. AWS Blu Age 是否支持群组数据生成 (GDG)？

   是的，在相对和绝对生成以及自动清理策略的支持下，支持批量使用 GDG。

4. AWS Blu Age 是否支持串联的数据集？

   是的，支持批量使用串联的数据集。使用串联后，可以将多个数据集作为单个数据集读取。请注意，Blusam数据集不能是串联的一部分。

5. 在 SQL 查询上应用的过程是什么？

   在代码转换期间进行了调整，具体取决于目标数据库。

6. 如果一个应用程序有多个数据库，哪些选项适用？

   为每个查询配置目标数据库，并定义应用程序和 Apache Tomcat 中的所有数据库。

7. Blusam可以禁用吗？

   是的，在主配置文件中，不需要数据库（有关更多信息，请参阅Blusam配置文档页面）。

8. 哪个 AWS Blu Age API 用于替换 IMS DB 等数据库？

JHDB（Java 分层结构 DataBase）API。

9. 哪 AWS 款 Blu Age 产品可用于将传统数据和数据库迁移到现代关系数据库管理系统 (RDBMS)？

   AWS Blu Age 数据库现代化工具（数据迁移器）。

10. 什么是 AWS Blu Age Data Simplifier，它在现代化中解决了什么问题？

    Data Simplifier 是 AWS Blu Age 中的一个核心库，它解决了在 Java 中处理传统内存访问模式的挑战。它提供了支持低级内存访问、传统数据类型（如分区、打包、字母数字）和混合 structured/raw 内存访问的结构，这些访问在大型机应用程序中很常见，但在 Java 中本机不可用。该库通过熟悉的 Java 模式（如 getters/setters 和基于类）公开了这些功能 APIs，使得 Java 开发人员可以在保留传统功能的同时访问这些功能。

11. AWS Blu Age 如何处理传统的内存布局和数据结构？

    AWS Blu Age 通过 R ec ord 接口处理传统内存布局，该接口提供了固定大小的字节数组的抽象。对于像 COBOL '01 数据项'这样的结构化数据，它使用在现代化过程中自动生成的RecordEntity子类。这些类保持了遗留数据的层次结构，每个元素都具有父子关系。该系统支持原始存储器访问和结构化访问模式，既保留了传统系统的灵活性，又提供了现代编程接口。

12. AWS Blu Age 是如何处理 VSAM 数据集现代化的？

    Blusam 组件通过专用 API、端点和管理 Web 应用程序（BAC：Blusam 管理控制台）为 VSAM 数据集的现代化提供支持。Blusam 依赖关系数据库作为后端（PostgreSQL，使用 RDS 或 Aurora）。

转换

1. 我在哪里可以找到有关转型过程的详细信息？

   请参阅 AWS Blu Insights 文档

2. AWS Blu Age 生成的模块叫什么名字？

   服务、实体、Web 和工具。

3. 为什么被 Java/Spring 选为 AWS Blu Age 的目标技术之一？

   Java/Spring之所以被选为目标技术，是因为它的广泛采用、庞大的人才库和强大的企业能力。Java 生态系统提供了大量支持现代应用程序开发的库、框架和工具。Spring framework 提供企业级功能、云原生功能，并遵循行业最佳实践，使其成为现代化应用程序的理想之选。

4. 包含 AWS Blu Age 生成的模块的父项目的名称是什么？

   父项目的名称以 “-pom” 为后缀，可以在转换中心使用名为 project 的 Transform 属性进行定义。

5. AWS Blu Age 如何管理旧版调度程序的现代化（如果提供）？

   AWS Blu Age 并未对传统调度程序资产进行现代化改造。在评估阶段会将它们考虑在内，以帮助识别可能丢失的工件。

6. 使用 AWS Blu Age 调试生成的代码有什么要求？

   任何支持 Java 的集成开发环境 (IDE)，例如 Eclipse JetBrain、或。 VisualCode

部署

1. 哪些环境可用于部署带有 Bl AWS u Age 的现代化应用程序？

   Windows 服务器、Linux 服务器和 Docker Linux 容器。

2. AWS Blu Age 重构后的应用程序能否在任何基础架构上运行？

   虽然 AWS Blu Age 重构后的应用程序不是为在任何基础架构上运行而设计的，但它们在部署选项方面提供了极大的灵活性。这些应用程序可以部署在各种计算平台上，包括云托管服务、无服务器计算和本地基础架构。

3. AWS Blu Age 支持哪种 MQ 配置？

   SQS，IBM M WebSphere Q。

4. 用户可以使用 AWS 大型机现代化运行时将 Java 业务应用程序逻辑部署到哪些应用程序服务器中？

   Apache Tomcat，版本大于或等于 10.1。

5. 重构后的应用程序如何与 Amazon AWS 服务 Aurora 等其他应用程序集成？

   现代化的应用程序 AWS 服务 通过支持向云原生数据库解决方案的转换，例如Amazon Aurora和适用于PostgreSQL的RDS，与之集成。 AWS Blu Age 可确保现代化应用程序之间的集成 AWS 服务，使组织能够使用云功能。这种集成扩展到 AWS 生态系统内的数据存储和应用程序服务。除了数据库存储之外， AWS Blu Age Runtime 还与各种存储集成， AWS 服务 包括用 ElastiCache 于 Redis 缓存的 Amazon、 AWS Secrets Manager 用于配置管理和用于 AWS 部署的大型机现代化。它支持亚马逊 EC2、亚马逊 EKS、由 Fargate 管理的 ECS 进行容器部署。该系统 AWS Identity and Access Management 可用于身份验证，使用 Amazon 简单存储服务进行存储，并支持 AWS 服务 通过配置和服务连接器与其他系统集成。

6. 重构后的应用程序如何确保满足可扩展性要求？

   该解决方案通过将应用程序转换为可以使用 AWS 弹性基础设施的云原生架构来确保可扩展性。它实现了支持水平和垂直缩放的现代设计模式和最佳实践。以服务为导向的方法允许独立扩展组件。现代化的应用程序可以利用云服务固有的可扩展性功能。

7. 源代码重构完成后会发生什么？

   源代码重构后，需要执行两个主要步骤。首先，构建重构后的应用程序。其次，部署应用程序。部署是在您的 AWS 账户（AWS 大型机现代化 AWSBlu Age运行时）中完成的，客户可以在其中管理自己的基础设施并在各种平台上进行部署，包括 Amazon EC2、Fargate 上 EC2 或上的 ECS、EKS。 EC2

8. 如何在自定义 Amazon Linux AMI 上部署和运行使用 AWS Blu Age 进行现代化改造的应用程序？

   这可以通过在亚马逊上使用 AWS Blu Age Runtime 部署应用程序来实现 EC2。该过程包括创建一个依赖于 AWS Blu Age Runtime 库的 Java/Spring 应用程序，然后将其部署到自定义 Amazon Linux AMI 上。有关此方法的详细说明，请参阅在亚马逊上设置 AWS Blu Age 运行时 EC2。

9. 是否有亚马逊系统映像 (AMI) 可用？ 是否有 Docker 镜像可用？

   - AMI：不是，由于客户需要根据自己的喜好自定义和设置环境，因此没有 AMI 可用。客户可以检索 AWS Blu Age 神器并根据自己的要求设置实例。
   - Docker Image：不，没有开箱即用的 docker 镜像，但是 “在容器上 AWS 设置 Blu Age 运行时” 页面解释了如何根据 AWS 蓝光时代运行时二进制文件构建自己的 docker 镜像并将其部署到合适的容器管理系统。

10. 客户能否将 AWS Blu Age 应用程序打包并作为 Docker 容器运行？

    是的，请参阅在容器上设置 AWS Blu Age 运行时。

11. 作业调度如何与批量配合使用？

    它与 Control-M /Stone 分支或任何其他分布式调度器集成。

安全性

1. 应用程序如何防御 SQL 注入攻击？

   该应用程序自始至终实施 SQL 最佳实践。所有数据库查询都使用预准备好的语句，通过将 SQL 代码与用户提供的数据分开，从而有效防止 SQL 注入攻击。这样可以确保用户输入始终被视为数据而不是可执行代码。

2. 有哪些措施可以防止操作系统命令注入漏洞？

AWS 安全团队定期对 AWS Blu Age Runtime 进行安全审查。如果团队检测到任何异常（例如可能允许操作系统命令注入的代码），请 AWS 立即着手解决。这种持续监控可确保运行时保持安全，抵御命令注入威胁。

3. 应用程序是否容易受到目录遍历攻击？

   不是。 此漏洞不适用于前端 Angular 应用程序。对于后端，应用程序仅使用一个端点，其接口合约有限，从不包含 “路径” 或 “目录” 信息。这种设计消除了目录遍历攻击的风险。

4. 该应用程序如何防御 XSS 攻击？

   该应用程序遵循Angular官方安全文档中描述的Angular安全最佳实践。该应用程序的性质在用户界面方面存在固有的局限性，因为它与传统的大型机应用程序相对应。该应用程序不处理由后端生成的 URLs、动态脚本或 HTML。这种有限的攻击面显著降低了 XSS 风险。

5. 针对CSRF攻击实施了哪些保护？

   该应用程序利用了 Spring 的原生 CSRF 支持，该支持开箱即用，可针对跨站请求伪造攻击提供强大的保护。

6. 用户输入会导致 HTTP 标头注入吗？

   不是。 用户输入的长度受到严格限制，以匹配传统的大型机应用程序格式。当后端收到请求时，它会立即根据预定义的格式格式化输入，这将截断任何注入的内容。此外，应用程序永远不会根据应用程序使用的单个端点中的用户输入来构造 HTTP 标头，这使得无法注入 HTTP 标头。

7. 如何保护应用程序免受点击劫持攻击？

   该应用程序使用击键而不是鼠标点击来执行大多数操作，因此它本质上可以抵御点击劫持攻击。这种设计选择大大减少了此类漏洞的攻击面。

8. 应用程序中会发生缓冲区溢出攻击吗？

   不是。 用户输入的长度受到严格限制，以符合传统的大型机应用程序格式。当后端收到请求时，它会立即根据相应的格式格式化输入，从而防止出现缓冲区溢出情况。

9. 如何管理访问控制和授权？

   访问控制可在全球范围内决定用户是否有权访问唯一的后端端点。经过身份验证后，应用程序代码和数据将处理授权控制。这样可以确保适当区分关注点并进行安全的访问管理。

10. 哪些措施可以防止会话劫持？

   结合使用Spring和Angular原生功能进行会话管理和XSS防护，并遵守框架最佳实践，可以防止会话劫持。该应用程序可以通过以下方式进一步增强保护：

- 使用 HTTPS：确保所有通信都经过加密。
- 实现基于令牌的身份验证：应用程序可以在需要时将其激活。
- 监控用户会话：应用程序可以激活会话监控和异常检测以识别可疑活动。

这些分层安全措施可提供全面保护，防止会话劫持企图。

使用 Rocket Software（以前是 Micro Focus）

本指南介绍了使用大型机现代化解决方案对大型机应用程序进行 AWS 平台改造的 end-to-end过程。AWS它描述了所有任务，并包括有关在 Amazon 上配置和操作 AWS 大型机现代化运行时的信息，EC2 从初始设置和分析到构建、测试和部署现代化应用程序。 AWS此外，还涵盖了一些高级主题，例如处理遗留数据结构、使用模板和预定义的项目，以及为流会话设置自动化。

主题

在亚马逊上设置 Rocket Software（前身为 Micro Focus EC2）

AWS Mainframe Modernization 提供了多个亚马逊机器映像 (AMIs)，其中包括 Rocket Software（前身为 Micro Focus）许可产品。它们 AMIs 允许您快速配置亚马逊弹性计算云 (Amazon EC2) 实例，以支持您控制和管理的 Rocket Software 环境。本主题提供了访问和启动这些内容所需的步骤 AMIs。使用它们完全 AMIs 是可选的，不需要他们来完成本用户指南中的教程。

主题

设置 Rocket 软件（前身为 Micro Focus）（在亚马逊上 EC2）的先决条件

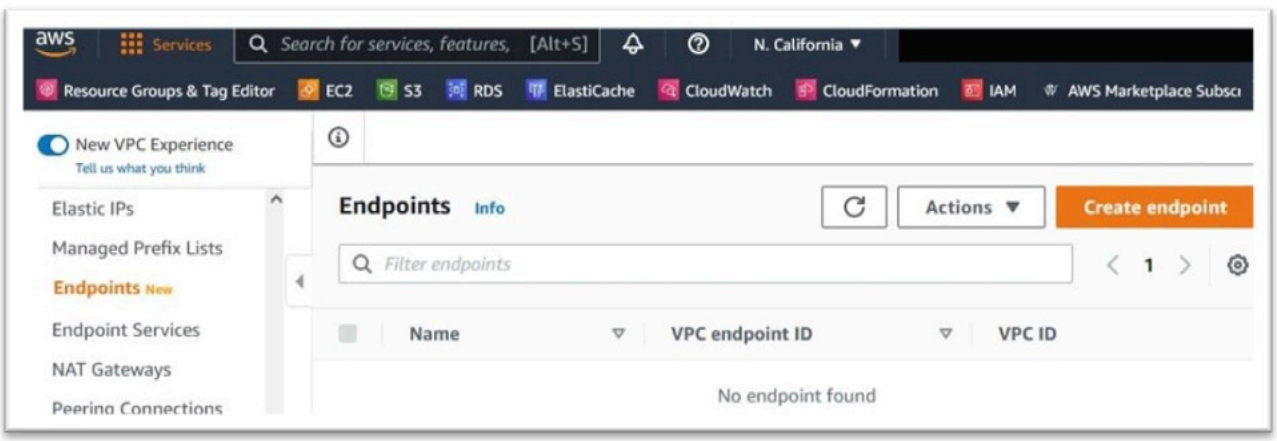
在设置 Rocket Software（在亚马逊上 EC2）时，请确保满足以下先决条件。

- 管理员访问将在其中创建 Amazon EC2 实例的账户。
- 确定将在 AWS 区域 何处创建 Amazon EC2 实例并验证 AWS Mainframe Modernization 服务是否可用。请参阅按区域划分的AWS 服务。请务必选择可提供该服务的区域。
- 确定将在其中创建亚马逊 EC2 实例的亚马逊虚拟私有云（亚马逊 VPC）。

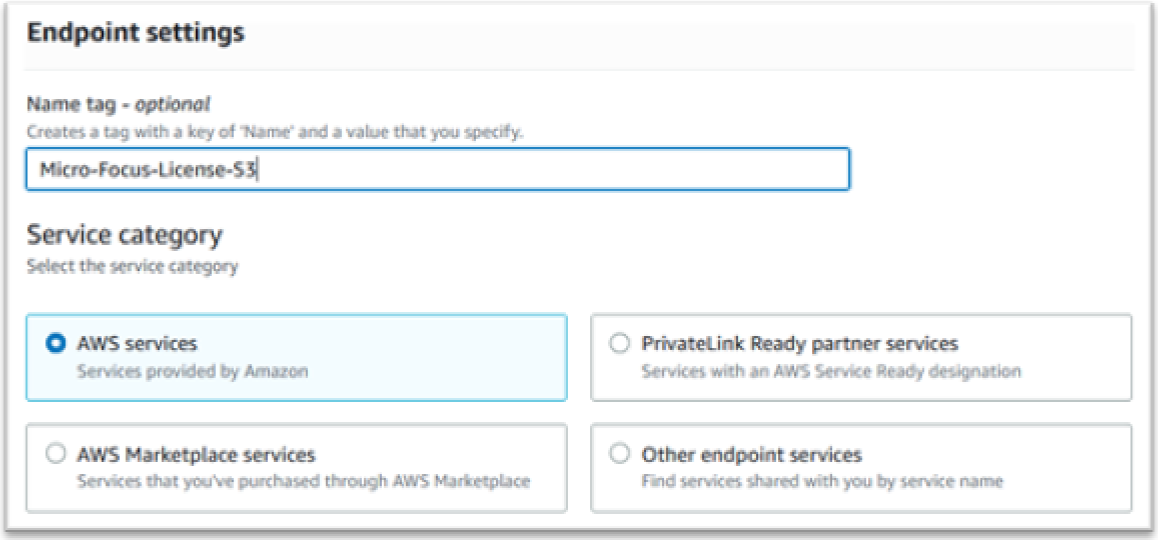
为 Amazon S3 创建 Amazon VPC 端点：

在本节中，创建一个供 Amazon S3 使用的 Amazon VPC 端点。设置此端点将对稍后设置 VPC 的互联网访问有帮助。

1. 在 AWS 管理控制台中导航到 Amazon VPC。
2. 在导航窗格中，选择端点。
3. 选择 创建端点。



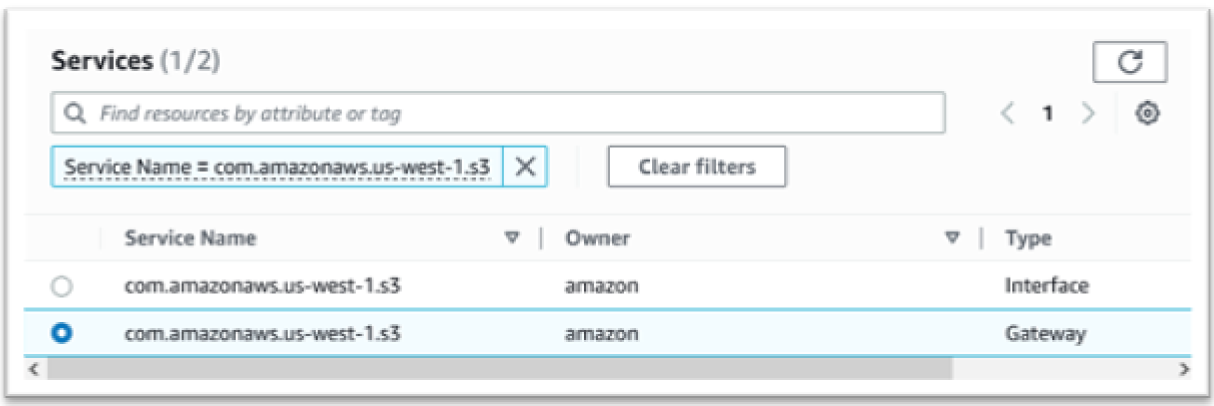
4. 输入一个有意义的名称标签，例如：“Micro-Focus-License-S3”。
5. 对于“服务类别”，选择 AWS 服务。



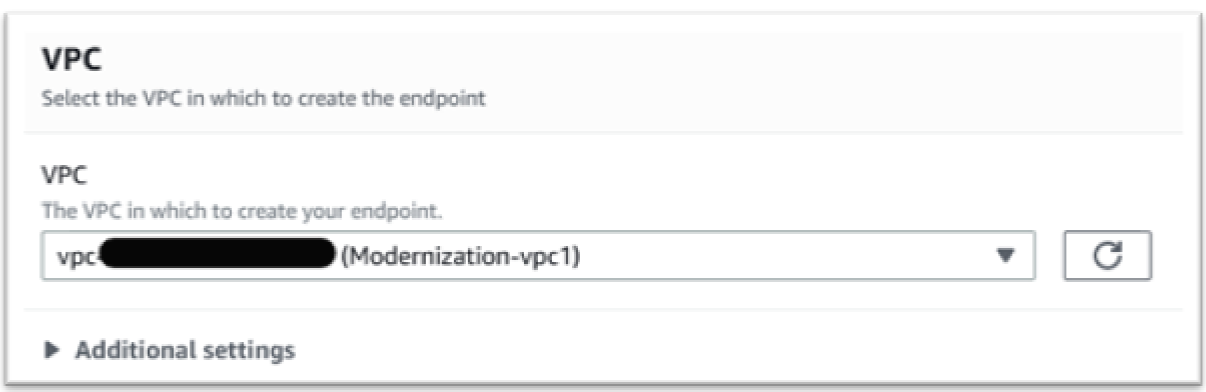
6. 在服务下搜索 Amazon S3 网关服务：com.amazonaws.[region].s3。

   对于 `us-west-1`，请使用：`com.amazonaws.us-west-1.s3`。

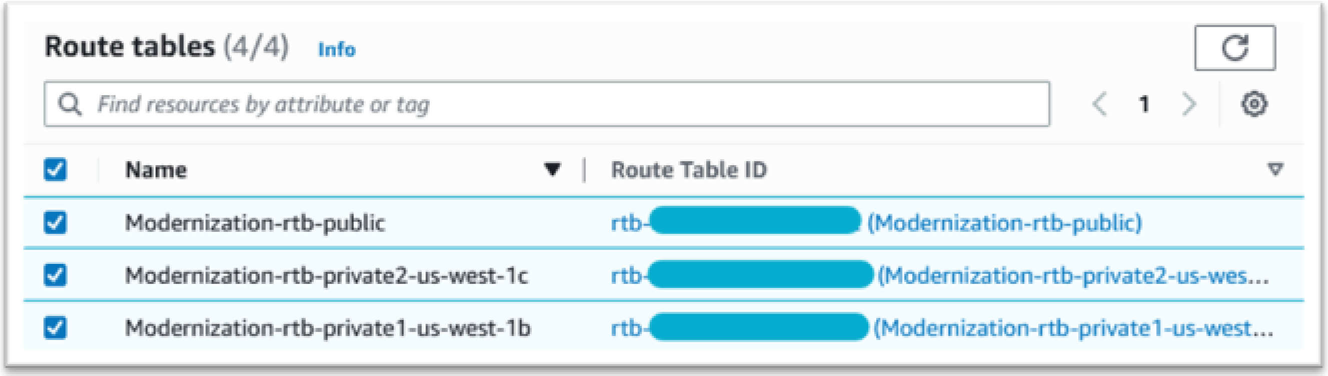
7. 选择网关服务。



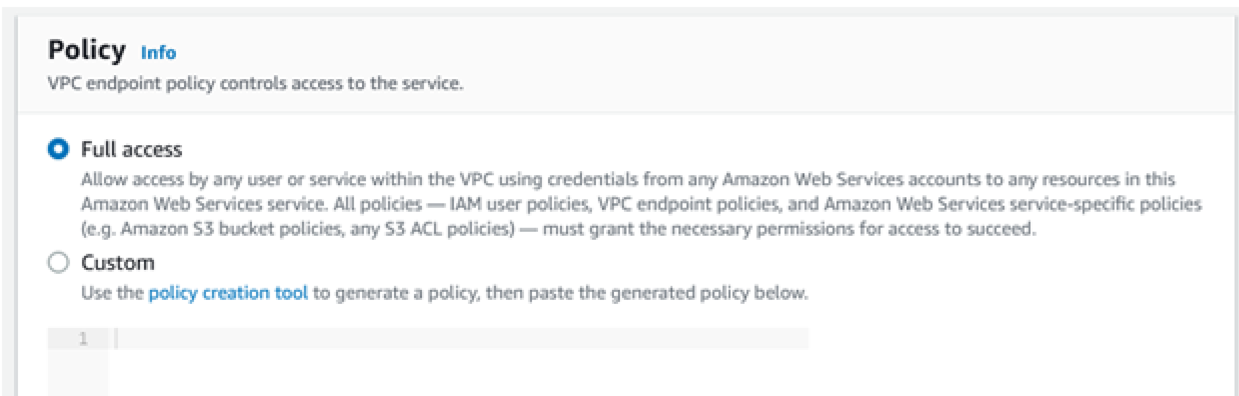
8. 对于 VPC，请选择您要使用的 VPC。



9. 选择 VPC 的所有路由表。



10. 在策略下，选择完全访问权限。



> **Note**
>
> 如果您决定创建自定义策略，请确保该策略可以访问 Amazon S3 存储桶 `s3://aws-supernova-marketplace-<region>-prod`。

11. 选择创建端点。

请求针对账户更新支持列表

请与您的 AWS 代表合作，将您的账户列入许可名 AWS Mainframe Modernization AMIs单。并提供以下信息：

- AWS 账户 身份证。
- 创建 Amazon VPC 终端节点 AWS 区域 的位置。
- 在为 Amazon S3 创建 Amazon VPC 端点：中创建的 Amazon VPC Amazon S3 端点 ID。此处是为 com.amazonaws.[region].s3 Gateway 端点创建的 `vpce-xxxxxxxxxxxxxxxxx` ID。
- 所有 Rocket Software 企业套件 AMI Amazon EC2 实例所需的许可证数量。

每个 CPU 内核需要一个许可证（CPUs 对于大多数 Amazon EC2 实例，每 2 v）。

有关更多信息，请参阅优化 CPU 选项。

将来，可以通过以下方式调整所请求的号码 AWS。

> **Note**
>
> 联系您的 AWS 代表，或者联系 AWS 支持 谁将代表您打开允许名单请求的支持请求。您不能直接申请，请求可能需要几天才能完成。

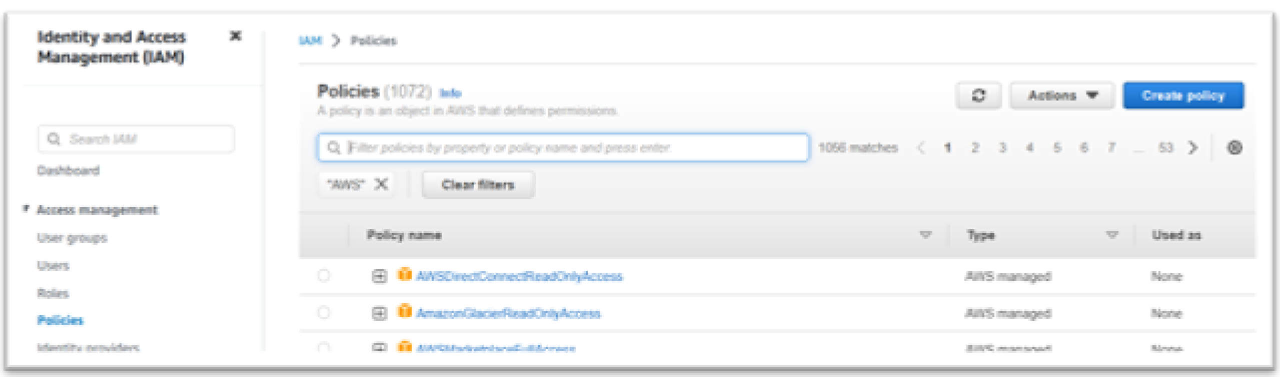
创建 AWS Identity and Access Management 角色

创建要由 AWS Mainframe Modernization Amazon EC2 实例使用的 AWS Identity and Access Management 策略和角色。通过 IAM 控制台创建角色会创建具有相同名称的关联实例配置文件。将此实例配置文件分配给 Amazon EC2 实例允许分配 Rocket 软件许可。有关实例配置文件的更多信息，请参阅使用 IAM 角色向在 Amazon EC2 实例上运行的应用程序授予权限。

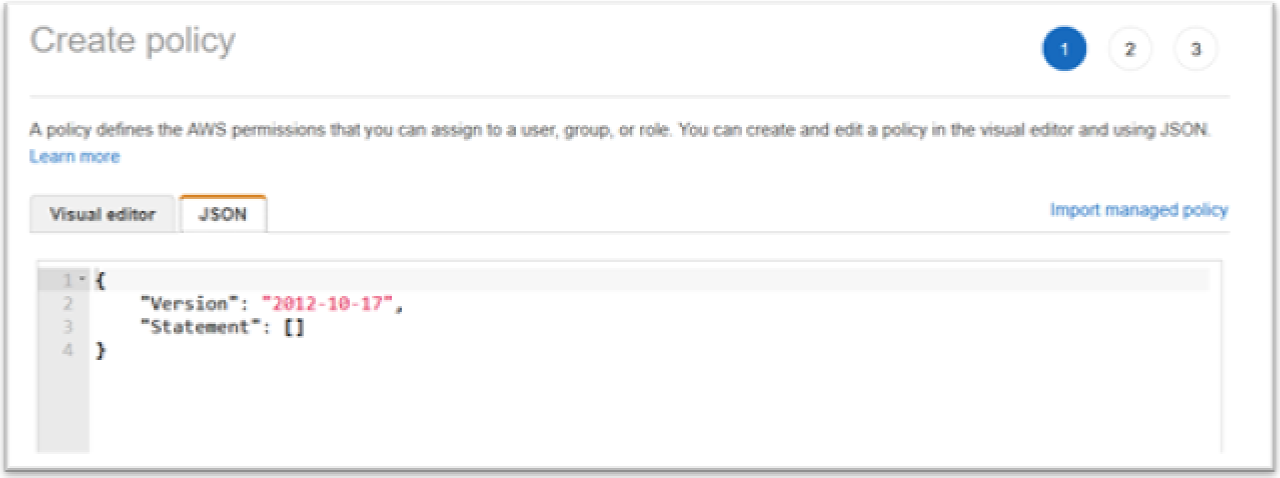
创建 IAM 策略

首先创建一个 IAM 策略并将其附加到角色。

1. 导航 AWS Identity and Access Management 到 AWS 管理控制台。
2. 选择策略，然后选择创建策略。



3. 选择 JSON 选项卡。



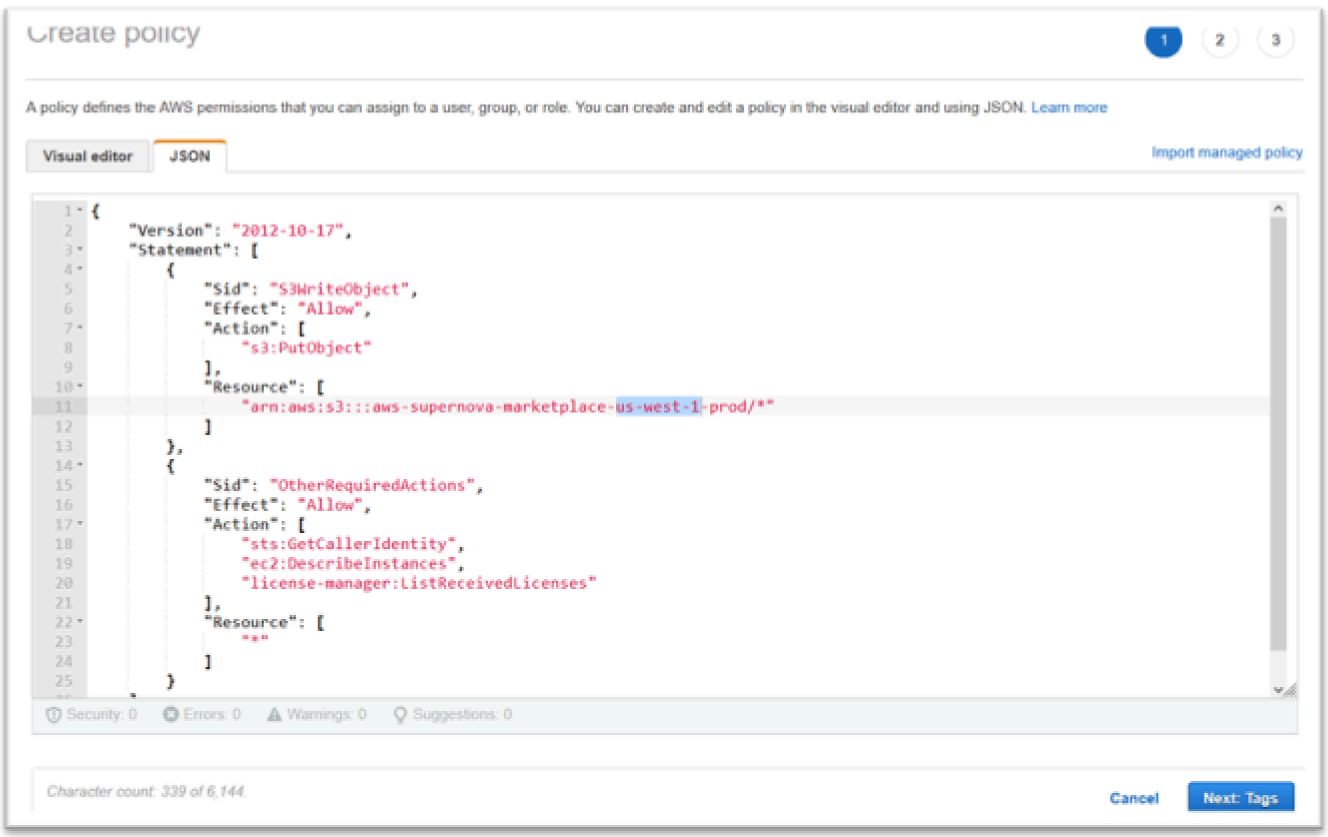
4. 将以下 JSON `us-west-1` 中的替换为定义 Amazon S3 终端节点 AWS 区域 的位置，然后将 JSON 复制并粘贴到策略编辑器中。

JSON

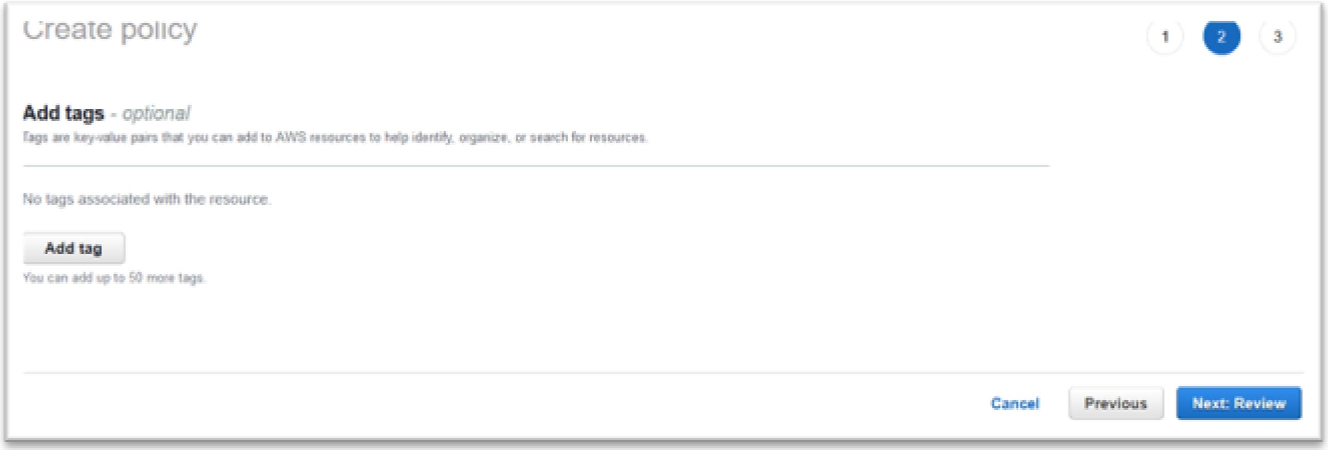
```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "S3WriteObject",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::aws-supernova-marketplace-us-west-1-prod/*"
            ]
        },
        {
            "Sid": "OtherRequiredActions",
            "Effect": "Allow",
            "Action": [
                "sts:GetCallerIdentity",
                "ec2:DescribeInstances",
                "license-manager:ListReceivedLicenses"
            ],
            "Resource": [
                "*"
```

```
                ]
            }
        ]
    }
```

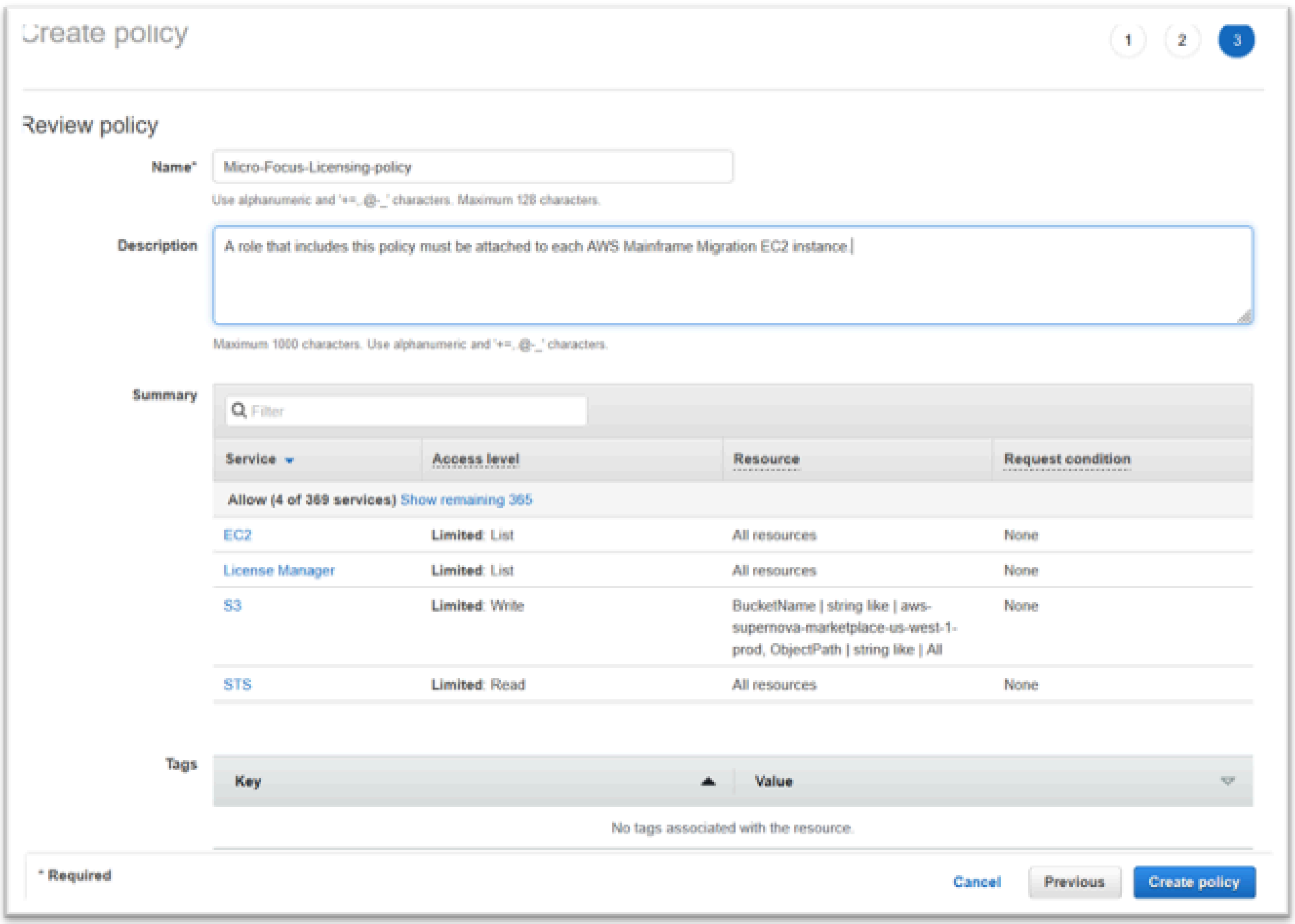
> **Note**
>
> Sid `OtherRequiredActions` 下的操作不支持资源级别的权限，必须在资源元素中指定 *。



5. 选择下一步：标签。



6. （可选）输入任何标签，然后选择下一步：审核。
7. 输入策略的名称，例如“Micro-Focus-Licensing-policy”。（可选）输入描述，例如 “包含此策略的角色必须附加到每个 AWS Mainframe Modernization Amazon EC2 实例”。

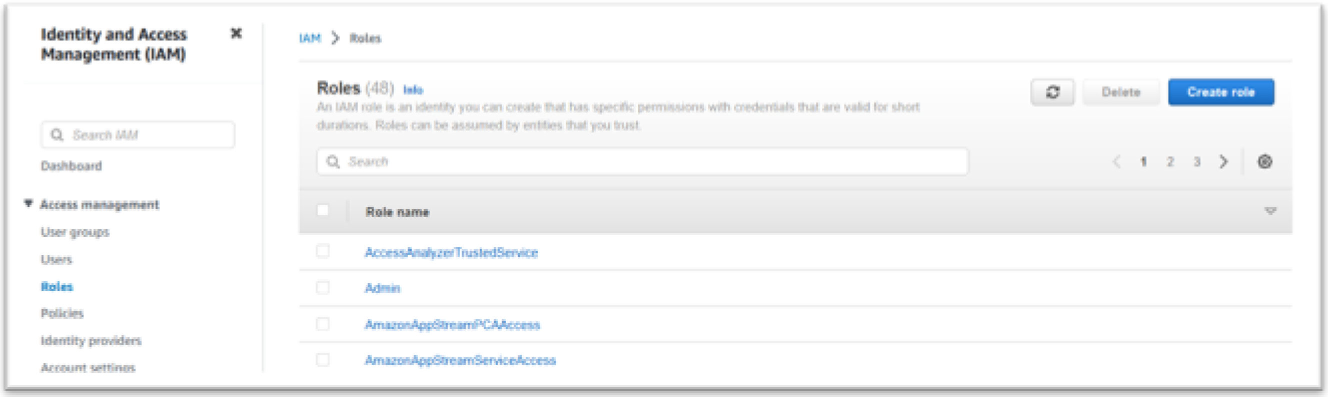


8. 选择创建策略。

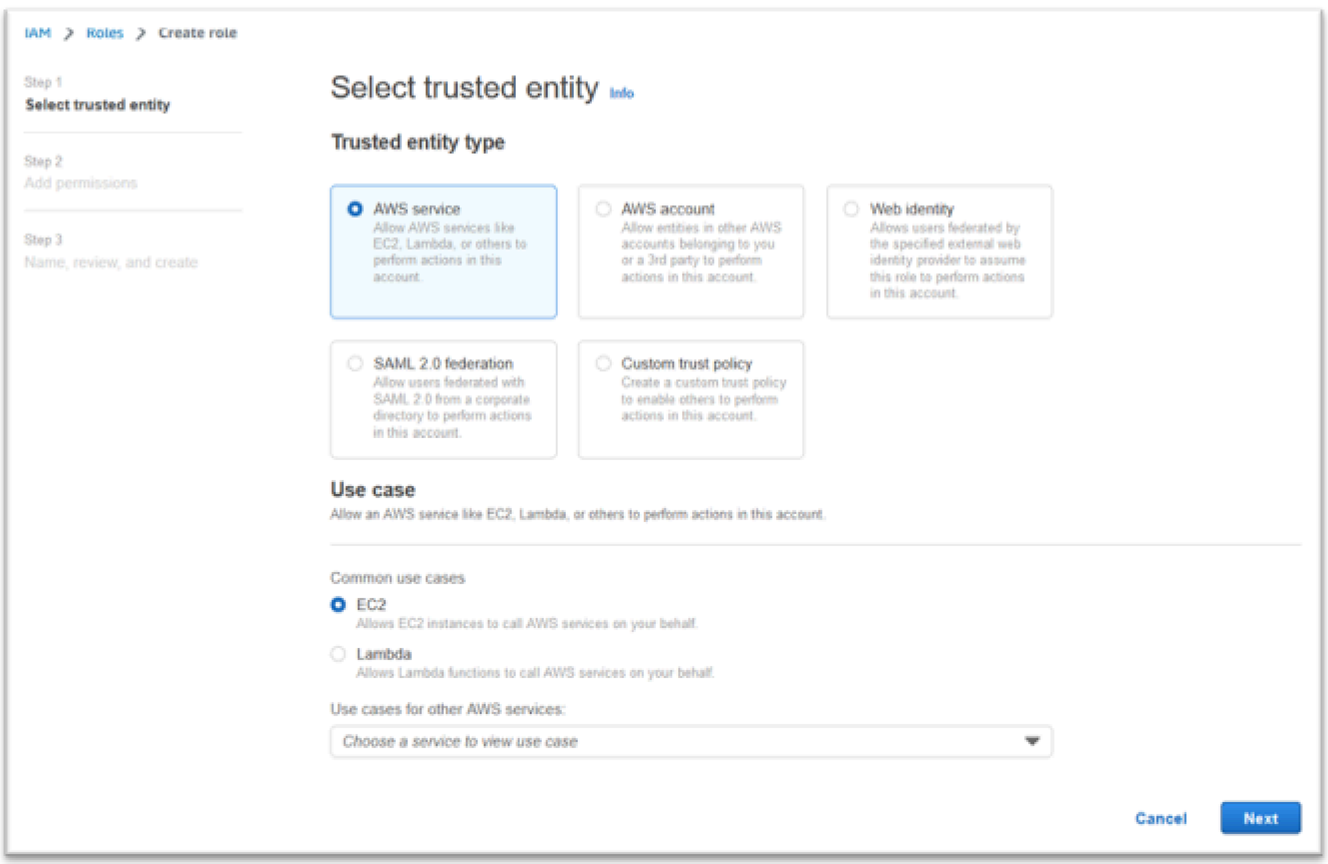
创建 IAM 角色

创建 IAM 策略后，您可以创建一个 IAM 角色并将其附加到该策略。

1. 在 AWS 管理控制台中，导航到 IAM。
2. 选择角色，然后选择创建角色。

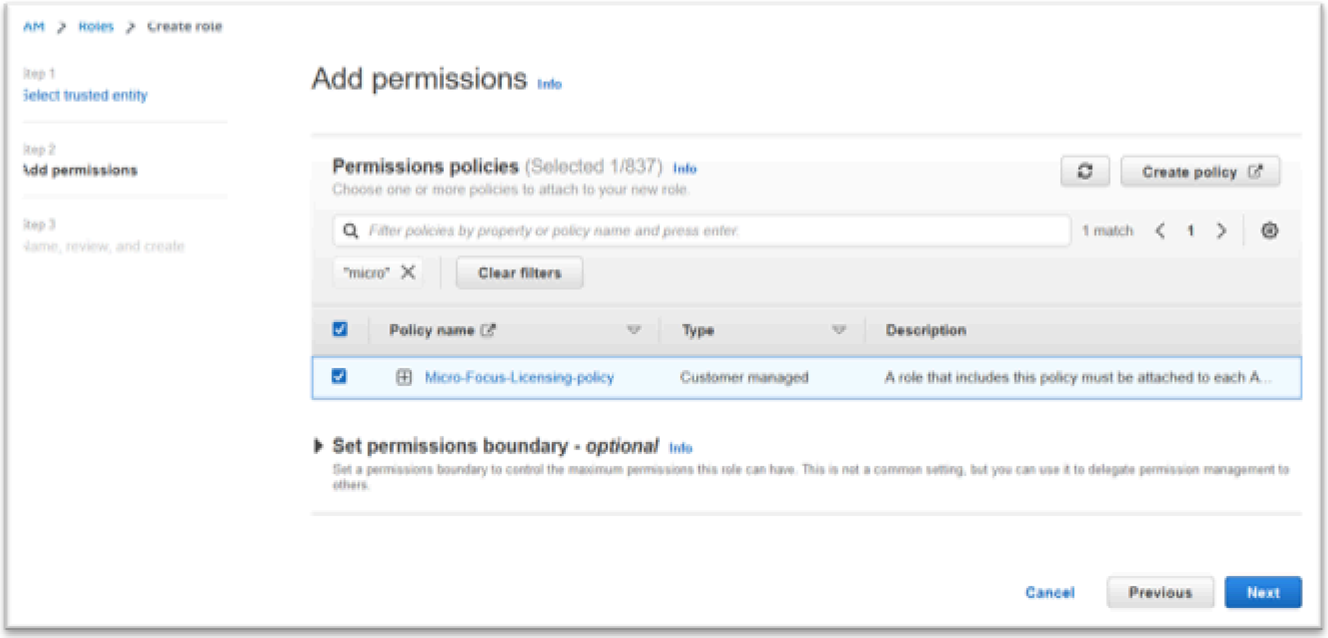


3. 将可信实体类型保留为AWS 服务，然后选择EC2常见用例。

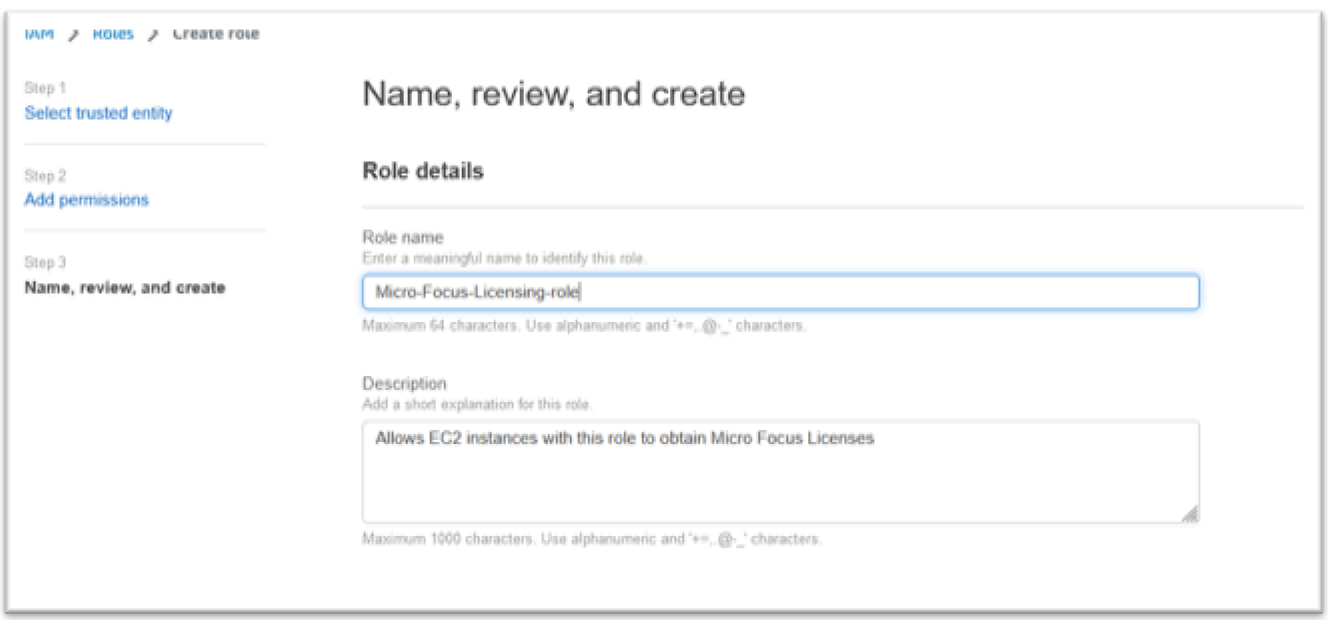


4. 选择下一步。

5. 在过滤条件中输入“Micro”，然后按 Enter 来应用该过滤条件。
6. 选择刚刚创建的策略，例如“Micro-Focus-Licensing-policy”。
7. 选择下一步。



8. 输入角色名称，例如“Micro-Focus-Licensing-role”。
9. 将描述替换为您自己的描述，例如 “允许具有此角色的 Amazon EC2 实例获取 Micro Focus 许可证”。



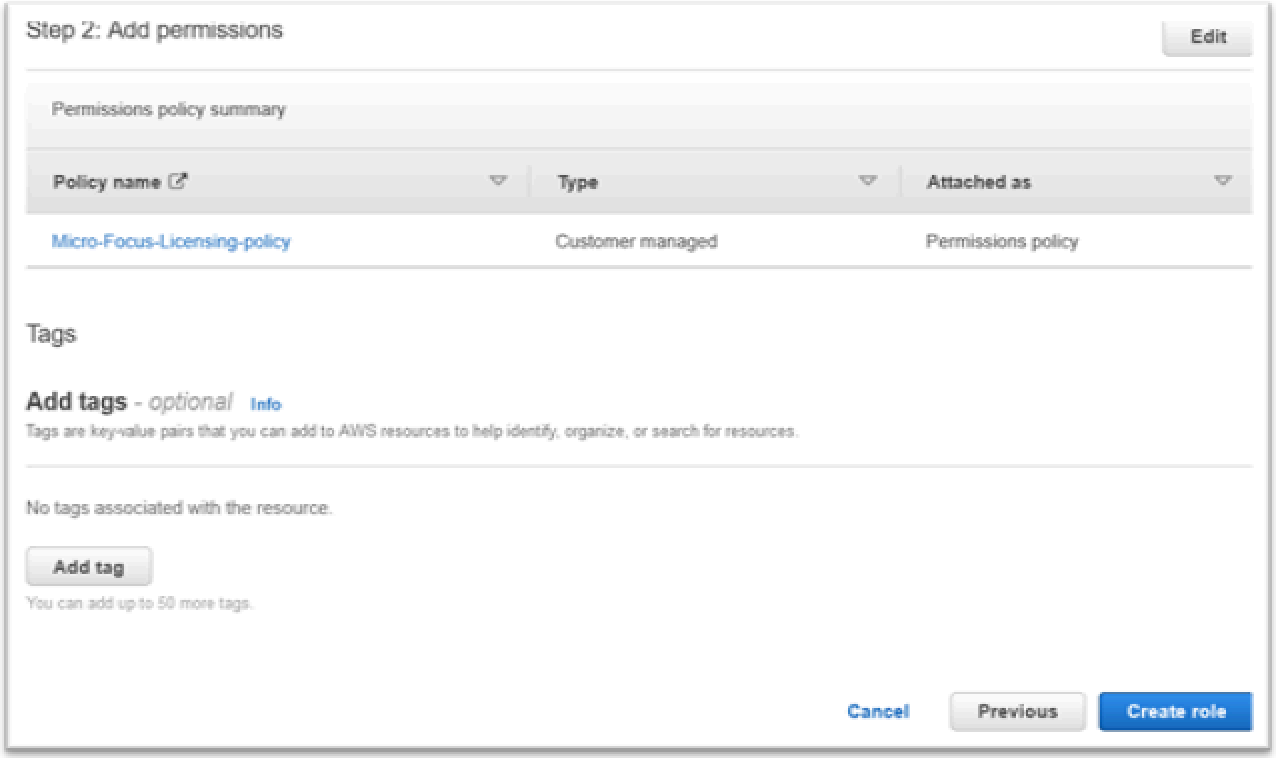
10. 在步骤 1：选择可信实体下，查看 JSON 并确认其具有以下值：

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Principal": {
                "Service": [
                    "ec2.amazonaws.com"
                ]
            }
        }
    ]
}
```

> **Note**
>
> Effect、Action 和 Principal 的顺序不重要。

11. 确认步骤 2：添加权限显示了您的许可策略。



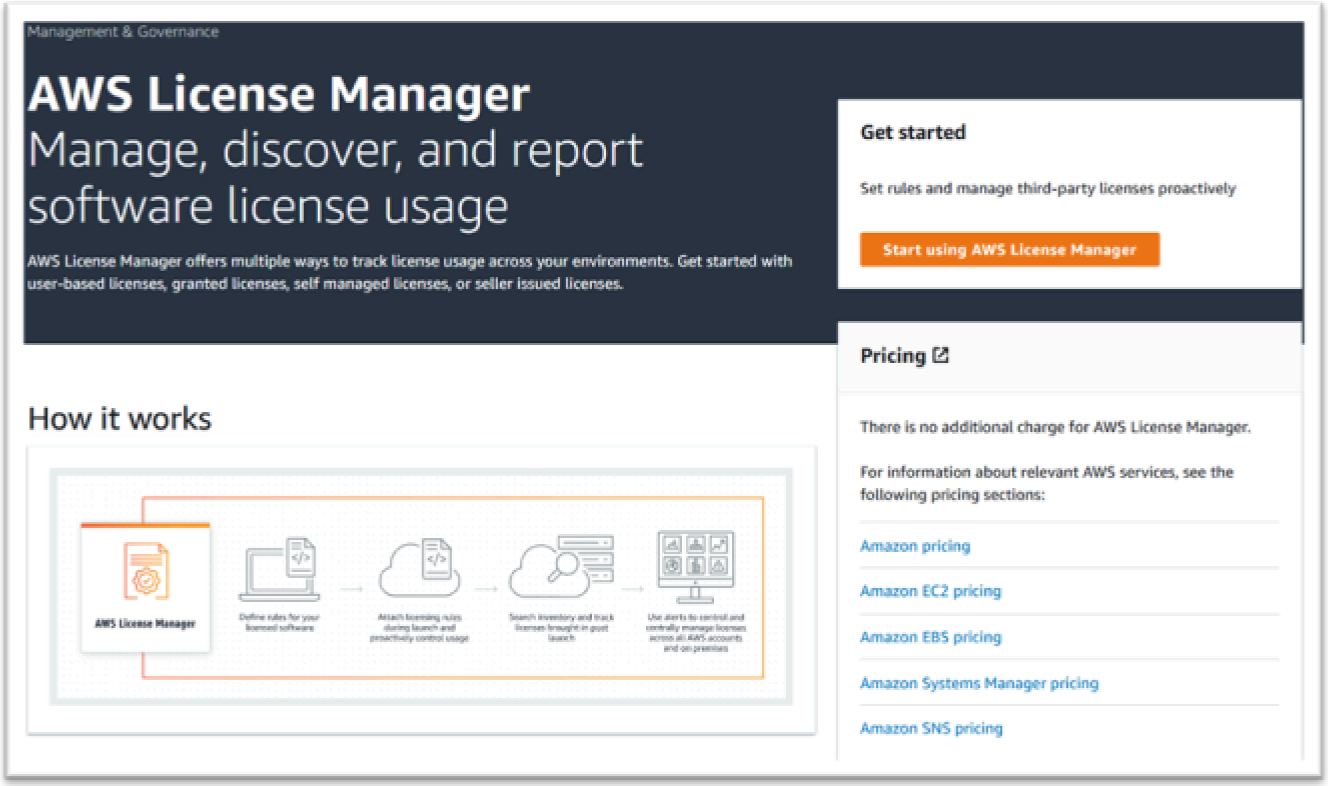
12. 选择创建角色。

在完成允许列表申请之后，继续执行后续步骤。

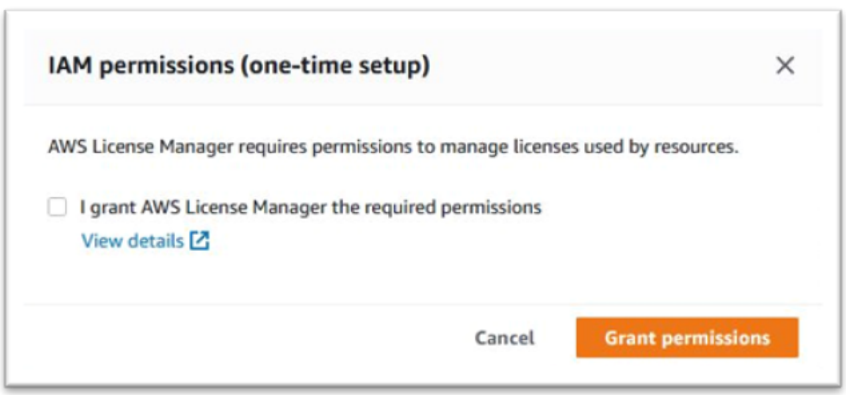
授予 License Manager 所需权限

您需要授予权限 AWS License Manager 才能设置 Rocket Software 运行时引擎（在亚马逊上 EC2）。

1. 导航 AWS License Manager 到 AWS 管理控制台。



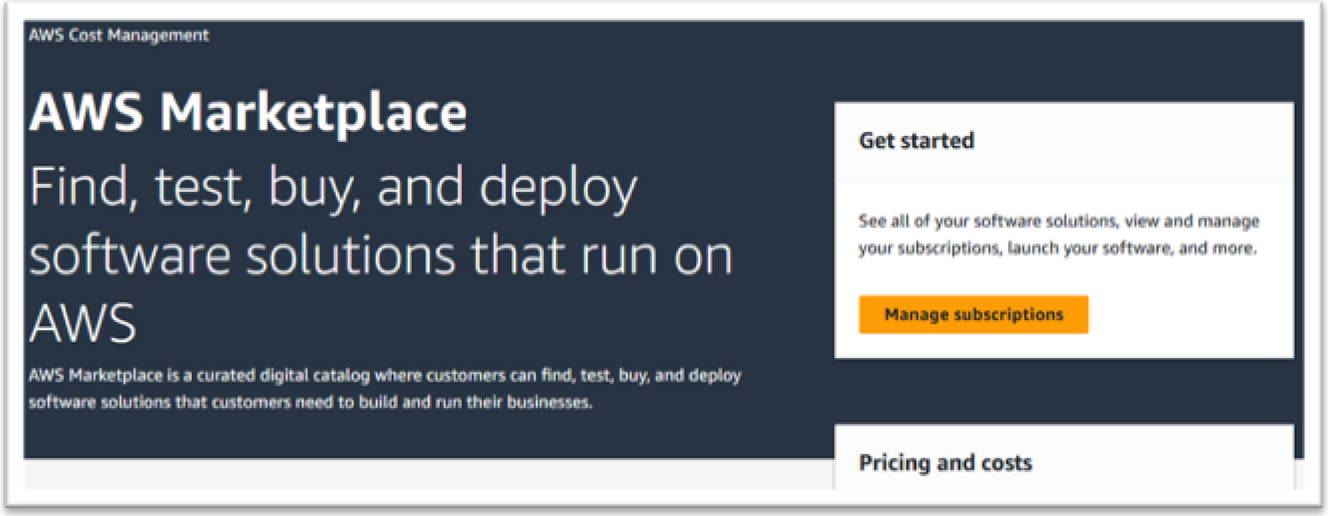
2. 选择开始使用 AWS License Manager。
3. 如果您看到以下弹出窗口，请查看详细信息，然后选中复选框并按授予权限。



订阅亚马逊机器映像

订阅产品后，您可以从该 AWS Marketplace 产品的 AMI 启动实例。您还可以在设置 Rocket Software（前身为 Micro Focus）运行时引擎（在亚马逊上 EC2）时管理您的订阅 AMIs 。

1. 导航到中的 “ AWS Marketplace 订阅” AWS 管理控制台。
2. 选择管理订阅。



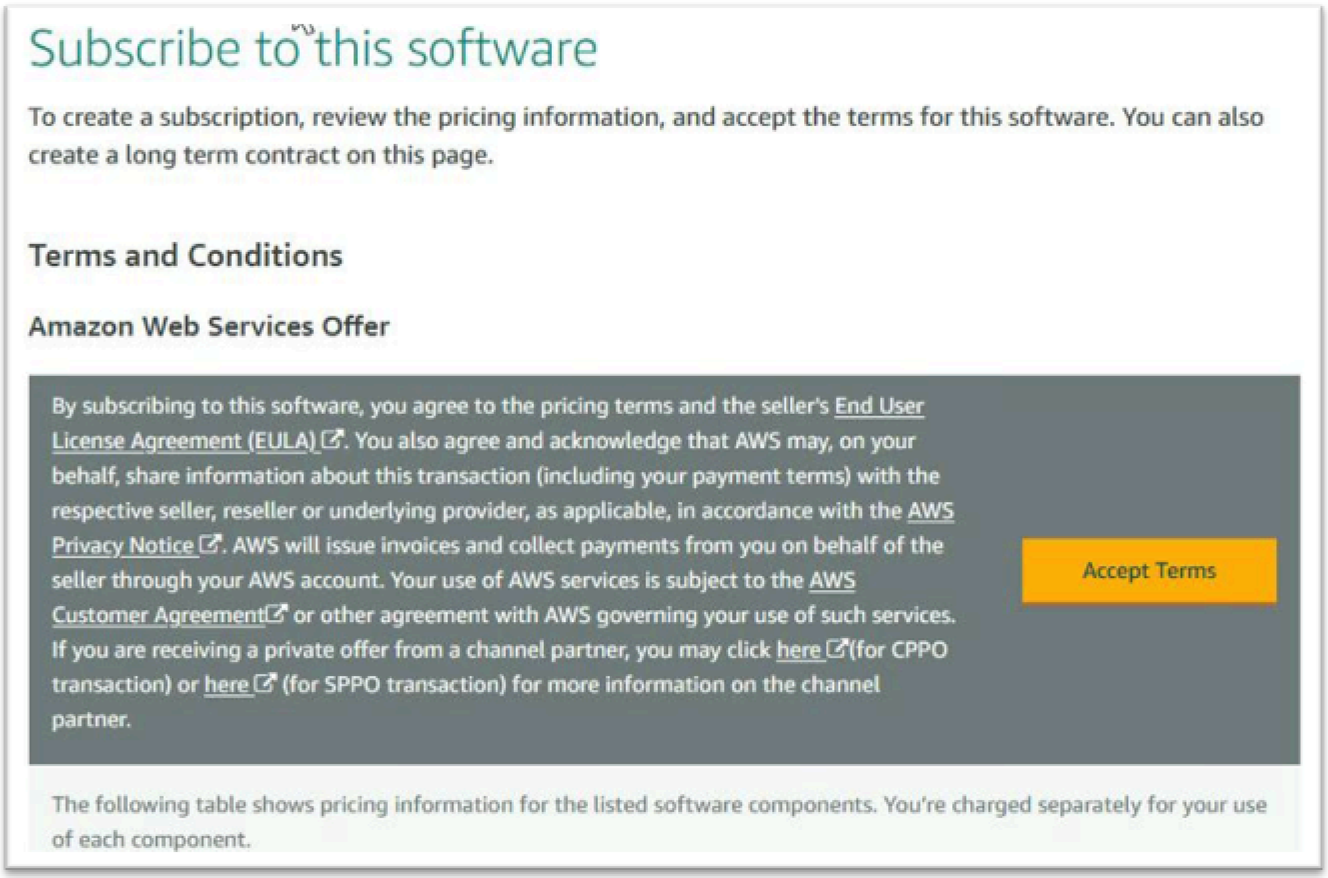
3. 复制以下链接之一并将其粘贴到浏览器地址栏中。

> **Note**
>
> 1. 仅选择您已获授权使用的产品之一的链接。
> 2. 按照请求针对账户更新支持列表页面使用这些链接，确保您的账户被列入许可名单。

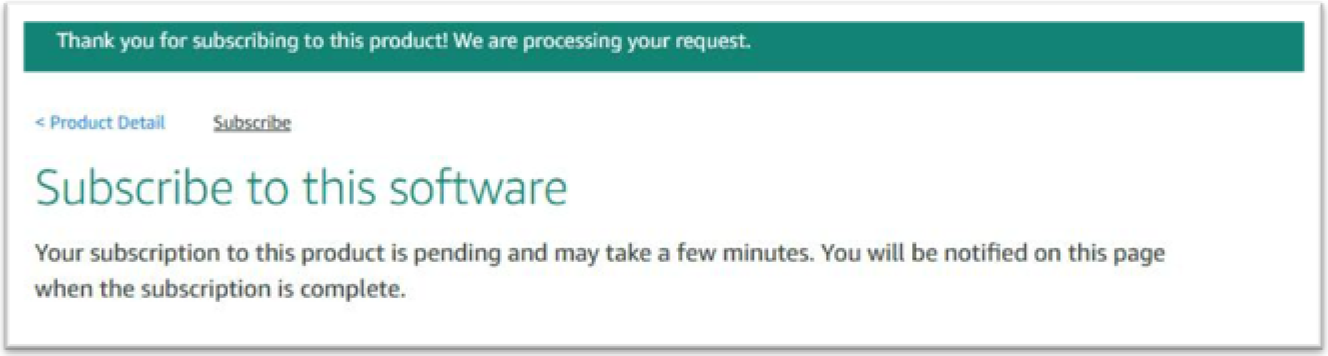
- 企业服务器：https://aws.amazon.com/marketplace/pp /prodview-g5emev63l7blc
- 适用于 Windows 的企业服务器：https://aws.amazon.com/marketplace/pp /prodview-lwybsiyikbhc2
- 企业开发人员：https://aws.amazon.com/marketplace/pp /prodview-77qmpr42yzxwk
- Visual Studio 2022 的企业开发人员：https://aws.amazon.com/marketplace/pp /prodview-m4l3lqiszo6cm
- 企业分析器：https://aws.amazon.com/marketplace/pp /prodview-ttheylcmcihm
- 适用于 Windows 的企业版构建工具：https://aws.amazon.com/marketplace/pp / prodview-2rw35bbt6uozi
- 企业存储过程：https://aws.amazon.com/marketplace/pp /prodview-zoeyqnsdsj6ha
- 使用 SQL Server 的企业存储过程 2019：https://aws.amazon.com/marketplace/pp /prodview-ynfklquwubnz4

4. 选择继续订阅。

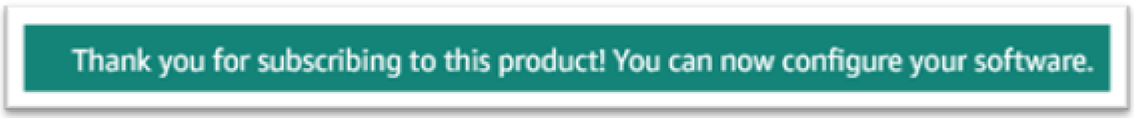
5. 如果可以接受“条款和条件”，请选择接受条款。



6. 处理此订阅可能需要几分钟时间。

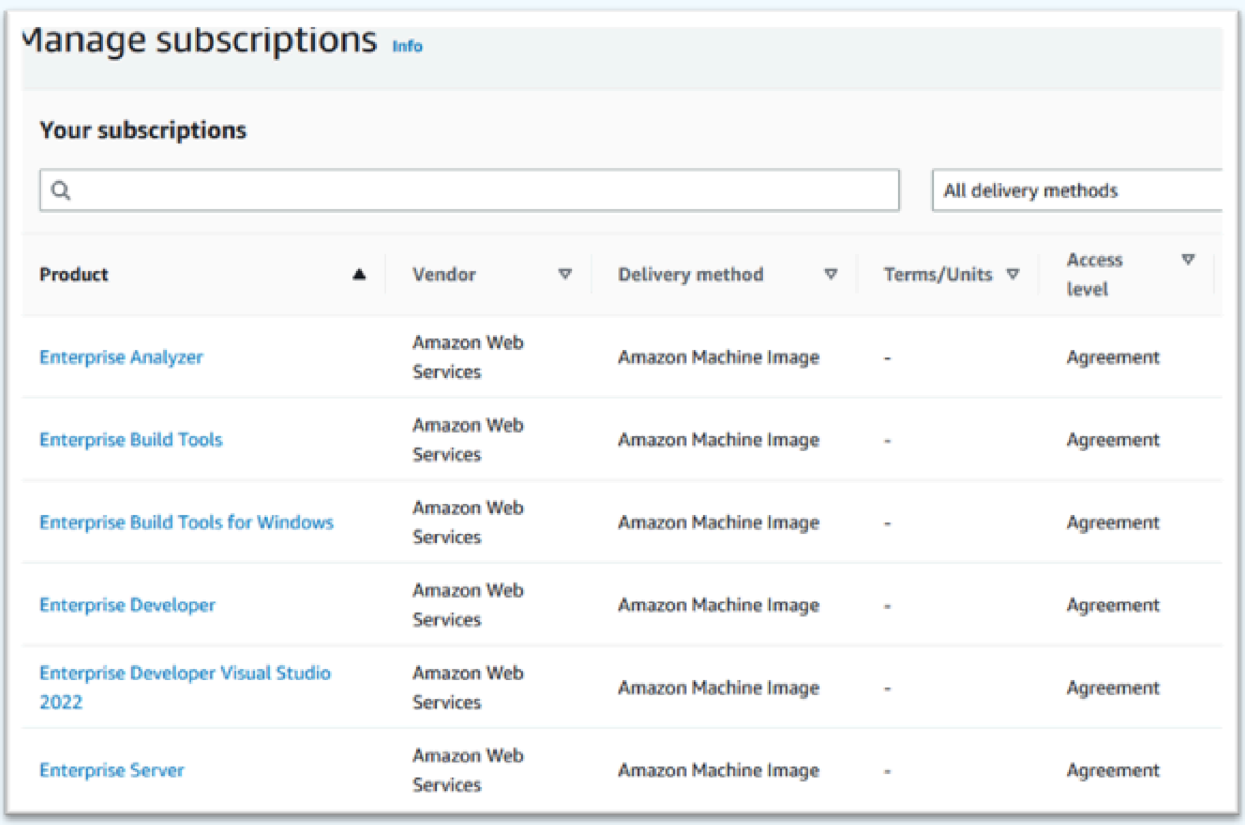


7. 显示感谢消息后，复制并粘贴步骤 3 中的下一个链接以继续添加订阅。



8. 当 “管理订阅” 显示所有已订阅 AMIs内容时停止。

> **Note**
>
> 面板首选项（齿轮图标）设置为以表格形式显示。



Manage subscriptions Info

Your subscriptions

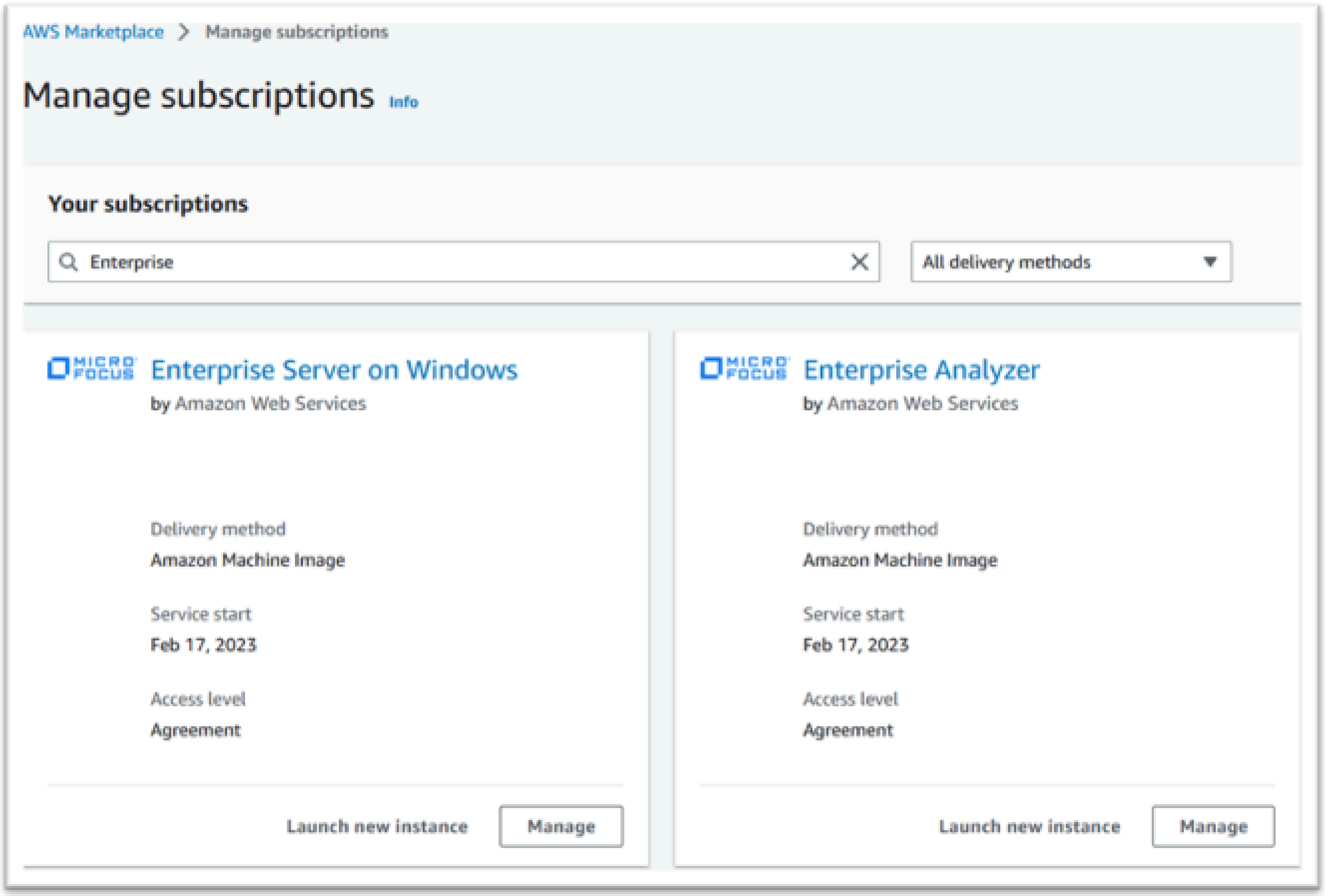
All delivery methods

Product	Vendor	Delivery method	Terms/Units	Access level
Enterprise Analyzer	Amazon Web Services	Amazon Machine Image	-	Agreement
Enterprise Build Tools	Amazon Web Services	Amazon Machine Image	-	Agreement
Enterprise Build Tools for Windows	Amazon Web Services	Amazon Machine Image	-	Agreement
Enterprise Developer	Amazon Web Services	Amazon Machine Image	-	Agreement
Enterprise Developer Visual Studio 2022	Amazon Web Services	Amazon Machine Image	-	Agreement
Enterprise Server	Amazon Web Services	Amazon Machine Image	-	Agreement

启动 AWS Mainframe Modernization 火箭软件（前身为 Micro Focus）实例

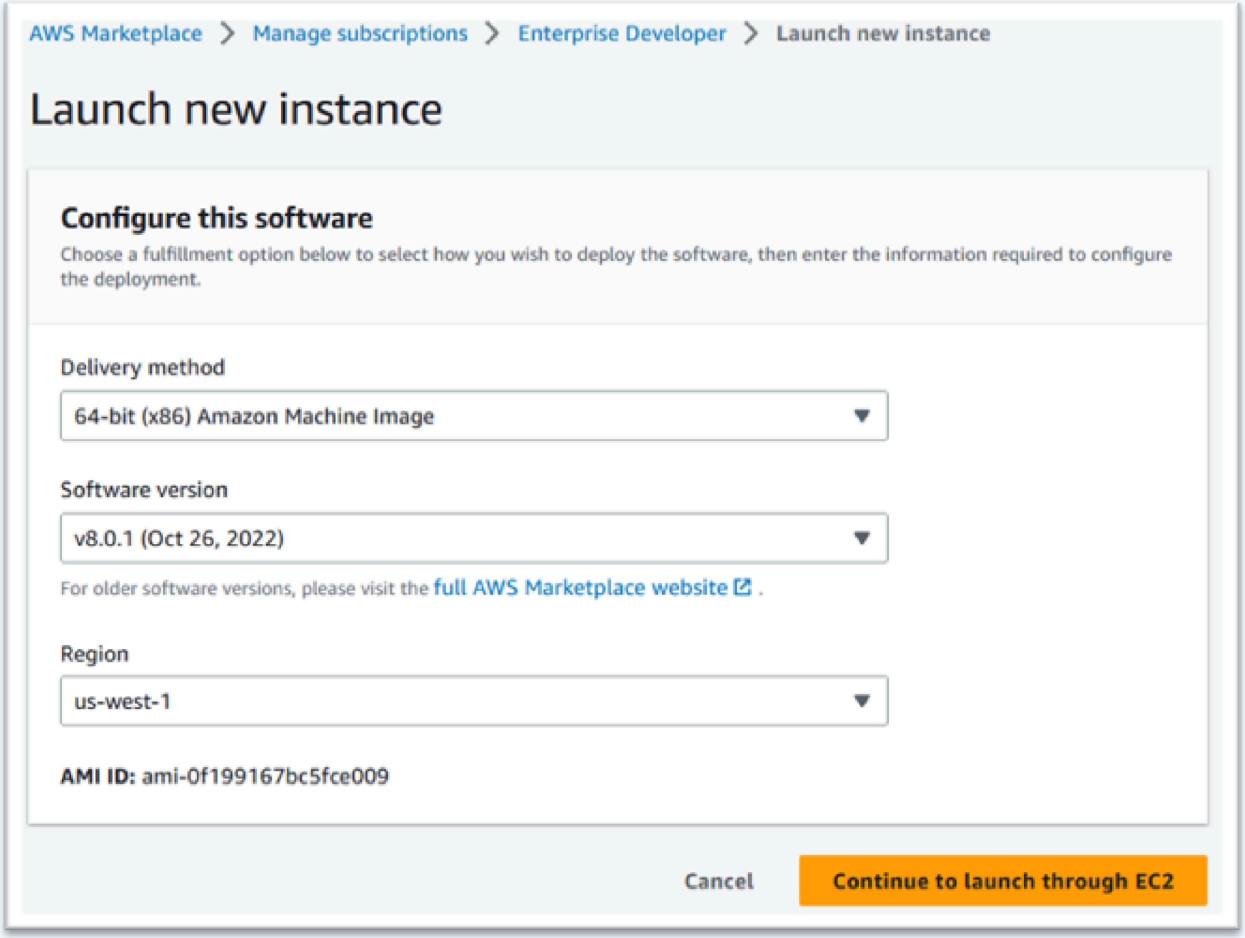
创建终端节点、IAM 策略、IAM 角色并订阅后 AMIs，您就可以在中启动 Rocke AWS Mainframe Modernization t Software（Micro Focus）实例了。 AWS 管理控制台

1. 导航到中的 “ AWS Marketplace 订阅” AWS 管理控制台。
2. 找到要启动的 AMI，然后选择启动新实例。



3. 在“启动新实例”对话框中，确保选择允许列表中的区域。
4. 按 “继续” 即可启动 EC2。

> **Note**
>
> 以下示例显示了企业开发者 AMI 的启动，但所有企业开发者 AMI 的流程都是一样的 AWS Mainframe Modernization AMIs。

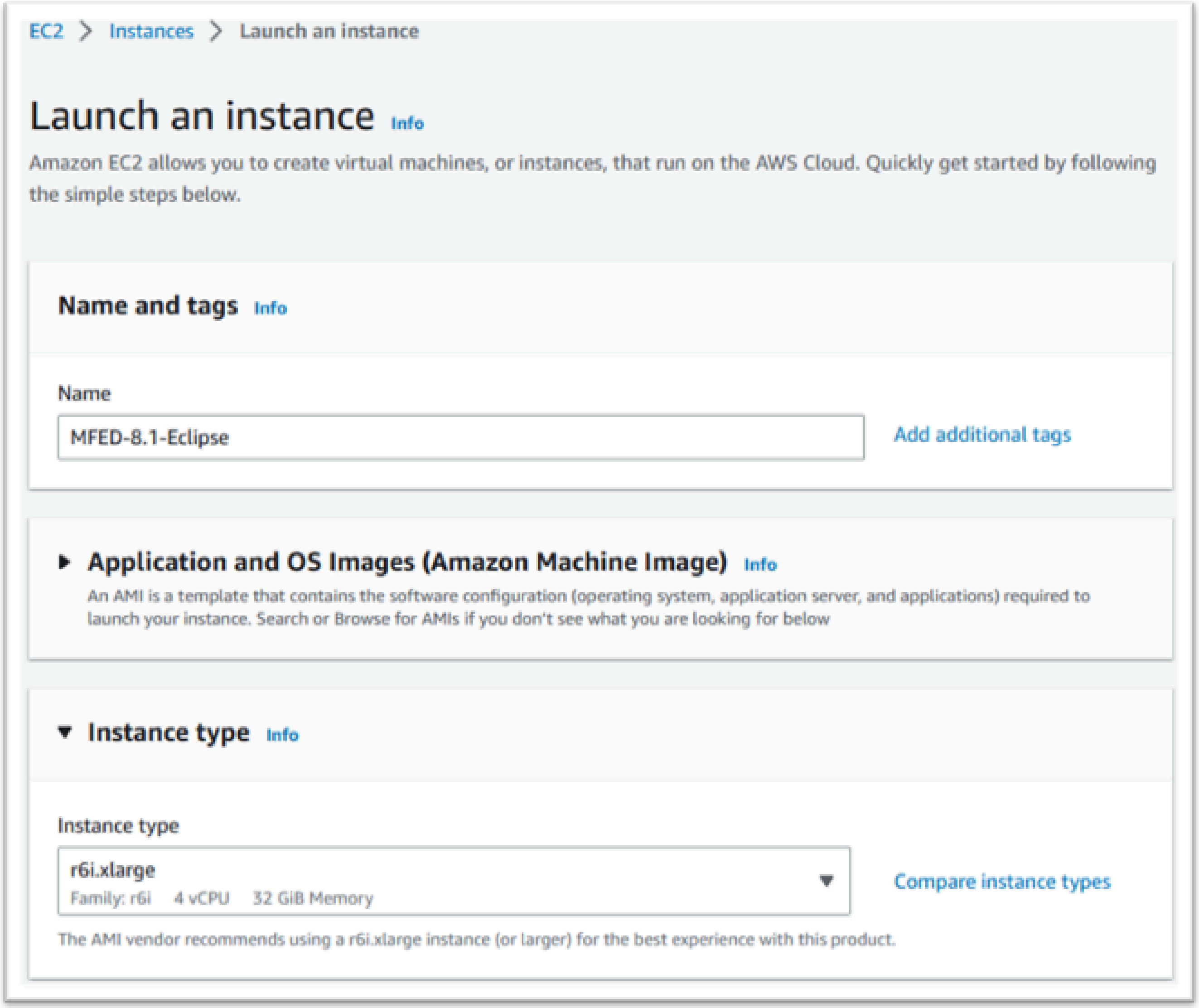


5. 输入服务器的名称。
6. 选择一个实例类型。

   所选的实例类型应由项目性能和成本要求决定。以下是建议的起点：

   - 对于 Enterprise Analyzer，选择 r6i.xlarge
   - 对于 Enterprise Developer，选择 r6i.large
   - 对于 Enterprise Server 的独立实例，选择 r6i.xlarge
   - 对于具有横向扩展功能的 Rocket 软件性能可用性集群 (PAC)，r6i.large

   > Note
   >
   > 屏幕截图中的“应用程序和操作系统映像”部分已折叠。

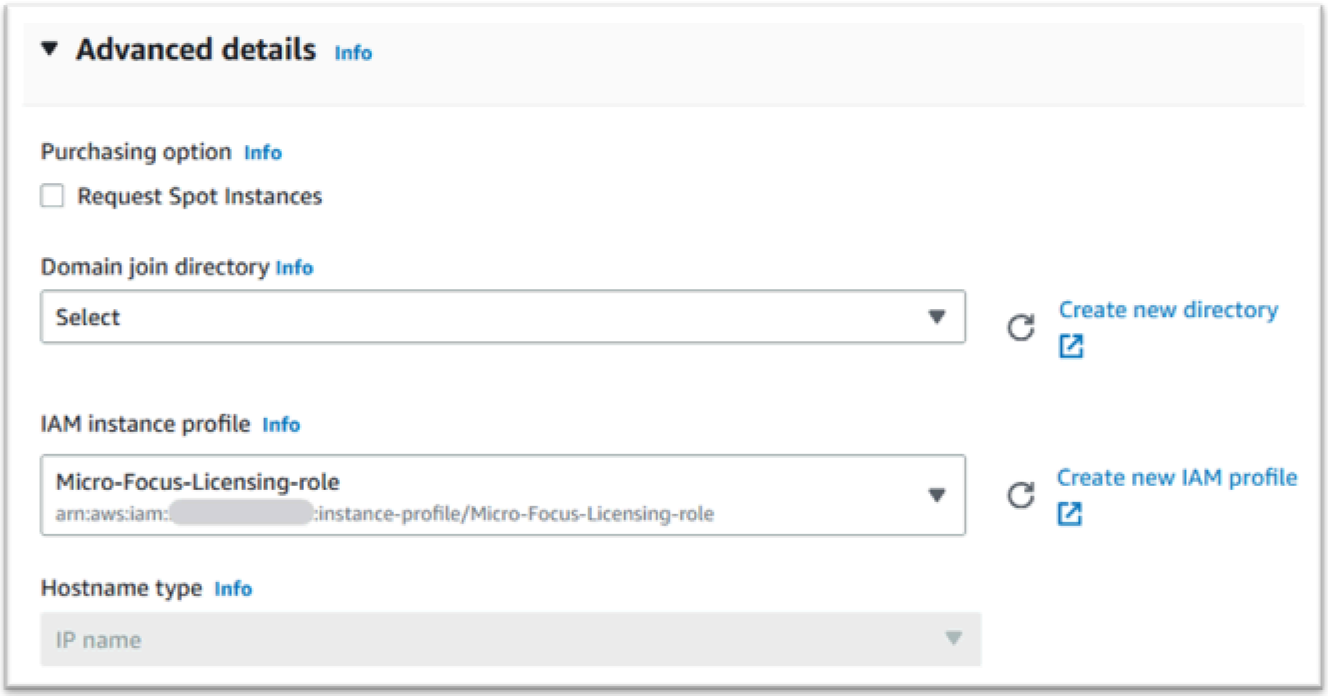


7. 选择或创建（并保存）密钥对（未显示）。

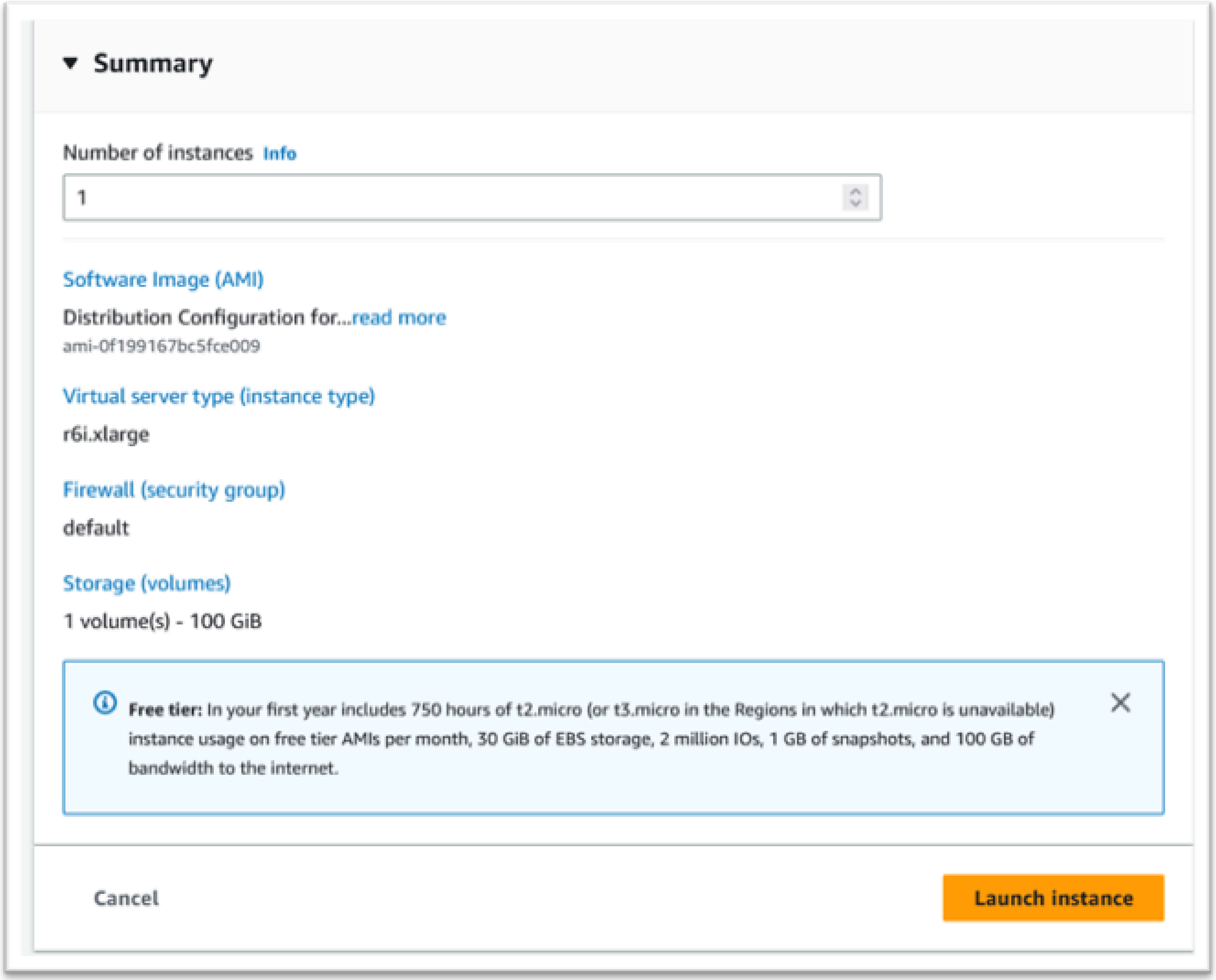
   有关 Linux 实例密钥对的更多信息，请参阅亚马逊 EC2 密钥对和 Linux 实例。

   有关 Windows 实例密钥对的更多信息，请参阅亚马逊 EC2 密钥对和 Windows 实例。
8. 编辑“网络设置”并选择允许列表中的 VPC 和相应的子网。
9. 选择或创建安全组。如果这是企业服务器 EC2 实例，则通常允许流向端口 86 和 10086 的 TCP 流量来管理 Rocket Software 配置。
10. （可选）为 Amazon EC2 实例配置存储空间。
11. 重要提示 – 展开“高级详细信息”，然后在“IAM 实例配置文件”下选择之前创建的许可角色，例如“Micro-Focus-Licensing-role”。

> **Note**
>
> 如果错过了此步骤，则在创建实例后，您可以从 EC2 实例的 “操作” 菜单的 “安全” 选项中修改 IAM 角色。

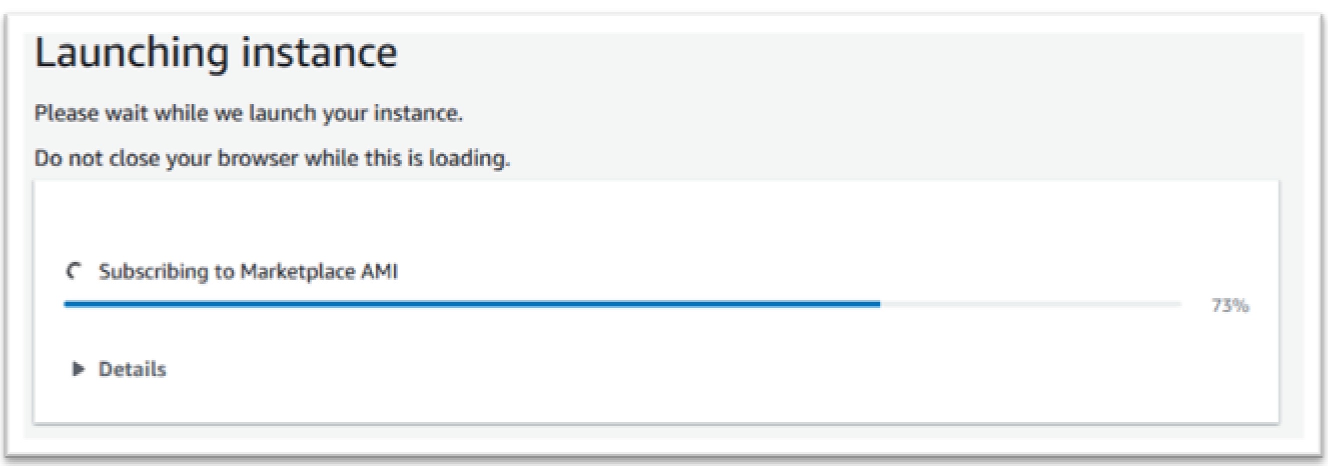


12. 审核“摘要”并按启动实例。

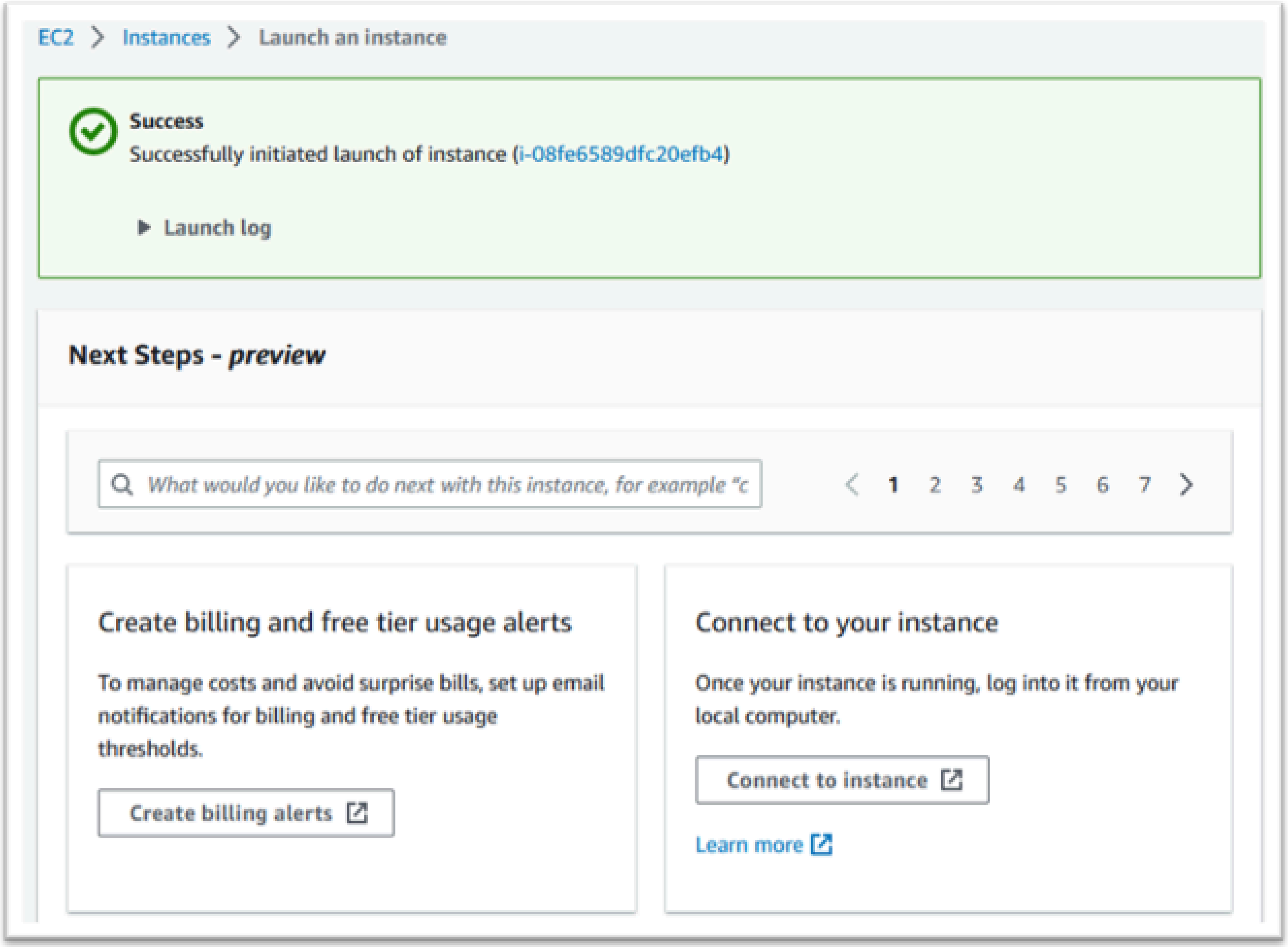


13. 如果选择的虚拟服务器类型无效，则实例启动将失败。

    如果发生这种情况，请选择编辑实例配置并更改实例类型。

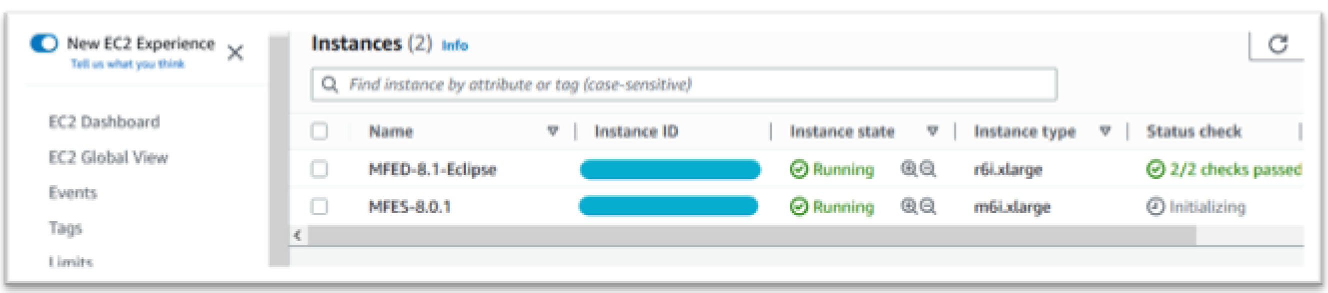


14. 显示“成功”消息后，选择连接到实例以获取连接详细信息。



15. 或者，EC2在中导航至 AWS 管理控制台。

16. 选择实例以查看新实例的状态。



无法访问互联网的子网或 VPC

如果子网或 VPC 没有出站互联网访问权限，请进行以下额外更改。

License Manager 需要访问以下 AWS 服务：

- com.amazonaws。 *region*.s3
- com.amazonaws。 *region*.ec2
- com.amazonaws。 *region*. 许可证管理器
- com.amazonaws。 *region*.sts

之前的步骤定义了 com.amazonaws。 *region*.s3 服务作为网关终端节点。此端点需要一个路由表条目，用于所有无法访问互联网的子网。

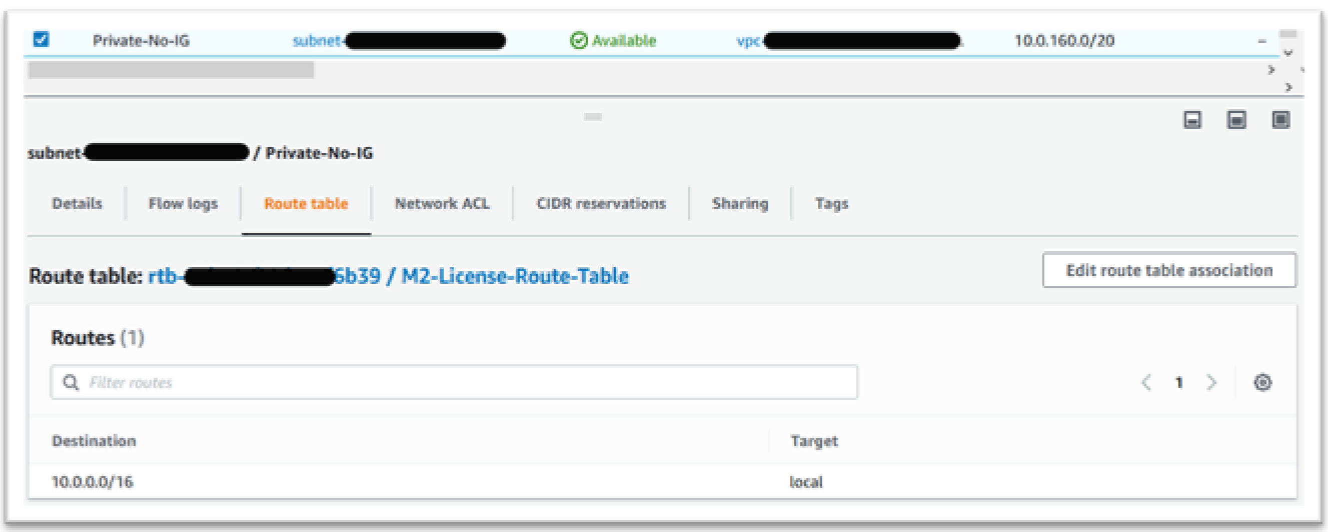
另外三项服务将被定义为接口端点。

主题

- [为 Amazon S3 端点添加路由表条目]
- [定义所需的安全组]
- [创建服务端点]

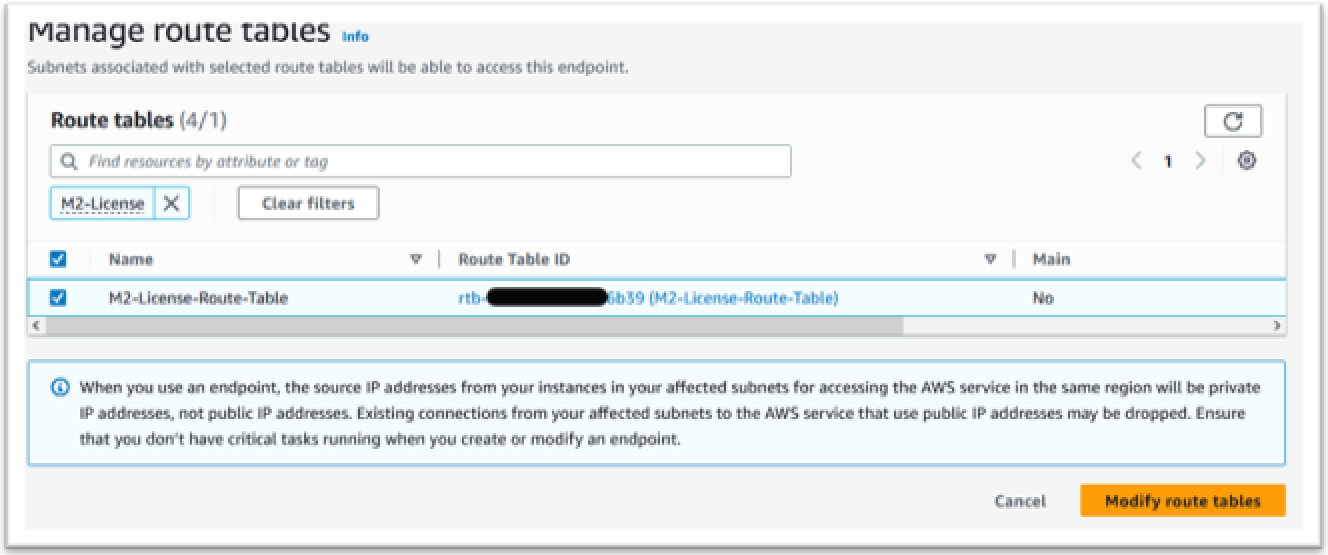
为 Amazon S3 端点添加路由表条目

1. 导航到中的 VPC， AWS 管理控制台 然后选择子网。
2. 选择将在其中创建 Amazon EC2 实例的子网，然后选择路由表选项卡。
3. 记录路由表 ID 的几位尾部数字。例如，下图中的 6b39。



4. 在导航窗格中，选择端点。

5. 选择之前创建的端点，然后从端点的“路由表”选项卡或从“操作”下拉列表中选择管理路由表。
6. 选择标有之前记录数字的路由表，然后按“修改路由表”。

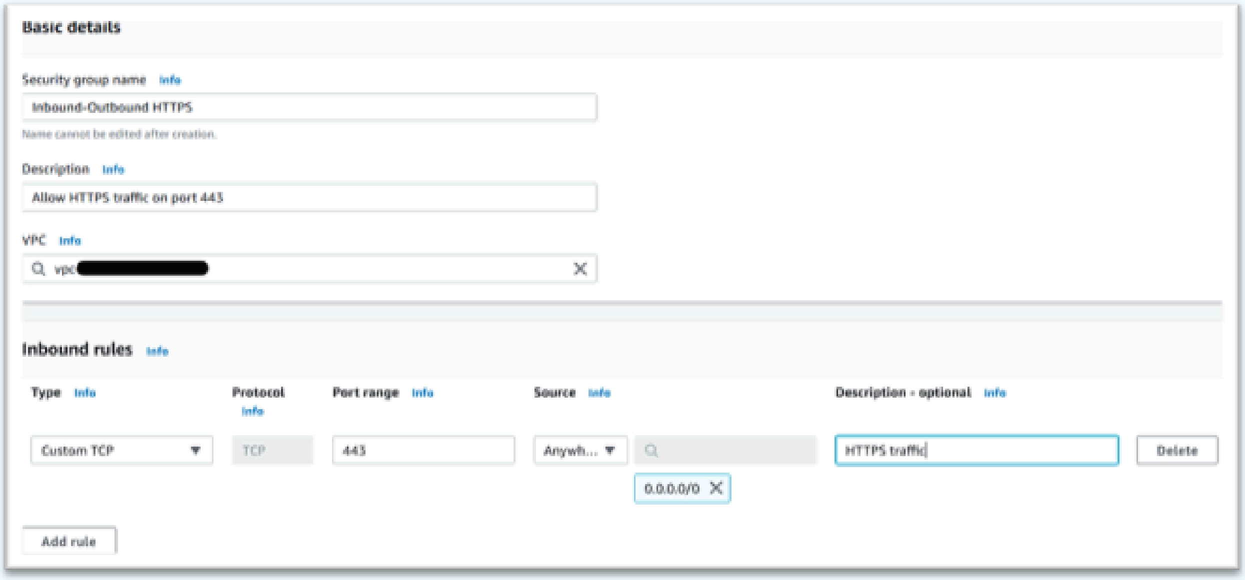


定义所需的安全组

Amazon EC2 和 Lic AWS STS ense Manager 服务通过端口 443 通过 HTTPS 进行通信。这种通信是双向的，需要入站和出站规则来允许实例与服务通信。

1. 在 AWS 管理控制台中，导航到 Amazon VPC。
2. 在导航栏中，找到安全组，然后选择创建安全组。
3. 输入安全组名称和描述，例如“Inbound-Outbound HTTPS”。
4. 在 VPC 选择区域中按“X“来移除默认 VPC，然后选择包含 S3 端点的 VPC。
5. 添加一条入站规则，在端口 443 上允许来自任何源的 TCP 流量。

> **Note**
>
> 通过限制“源”，可以进一步限制入站（和出站规则）。有关更多信息，请参阅 Amazon VPC 用户指南中的[使用安全组控制 AWS 资源流量](#)。

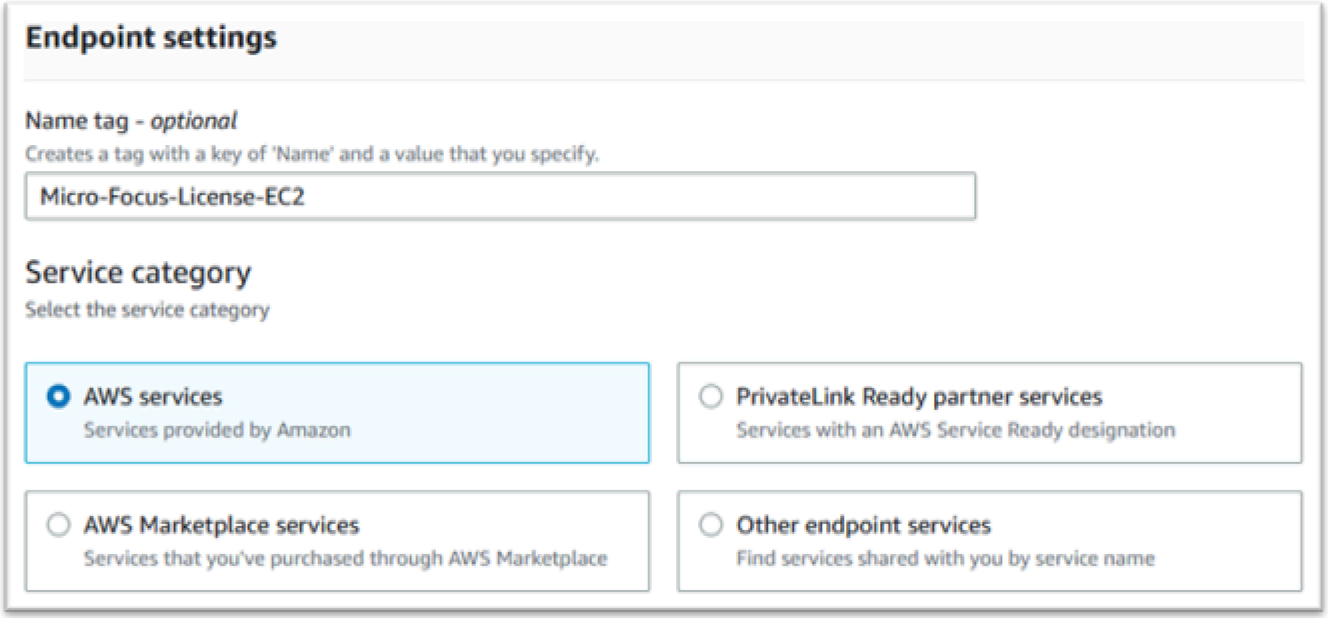


6. 按创建安全组。

创建服务端点

重复以下过程三次，即针对每个服务各执行一次。

1. 导航到中的 Amazon VPC， AWS 管理控制台 然后选择终端节点。
2. 按创建端点。
3. 输入名称，例如 “Micro-Focus-License-License-ST”、“micro-Focus-License-EC2 STS” 或 “Micro-Focus-License-Manager
4. 对于“服务类别”，选择 AWS 服务。



5. 在“服务”下搜索匹配的接口服务，即以下服务之一：

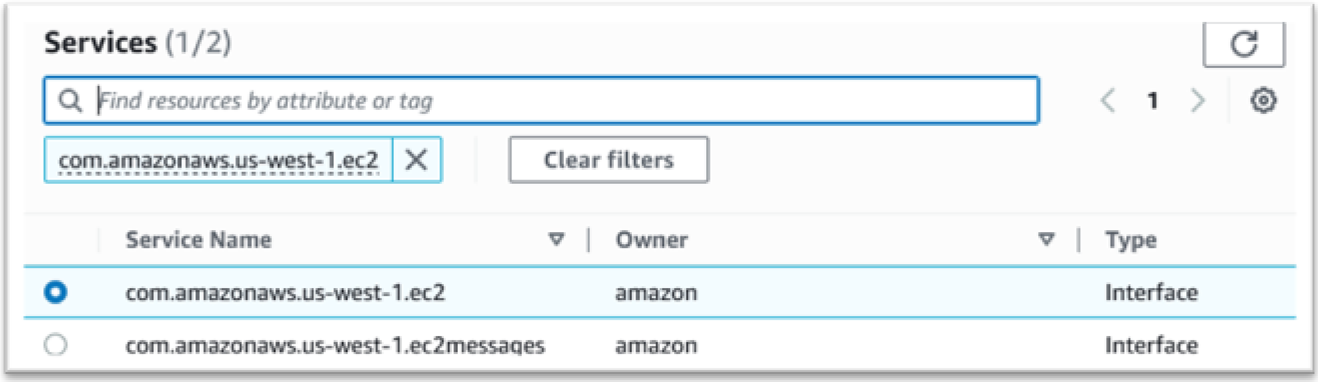
   - “com.amazonaws。 *region*.ec2”
   - “com.amazonaws。 *region*.sts”
   - “com.amazonaws。 *region*.license-Manager”

   例如：

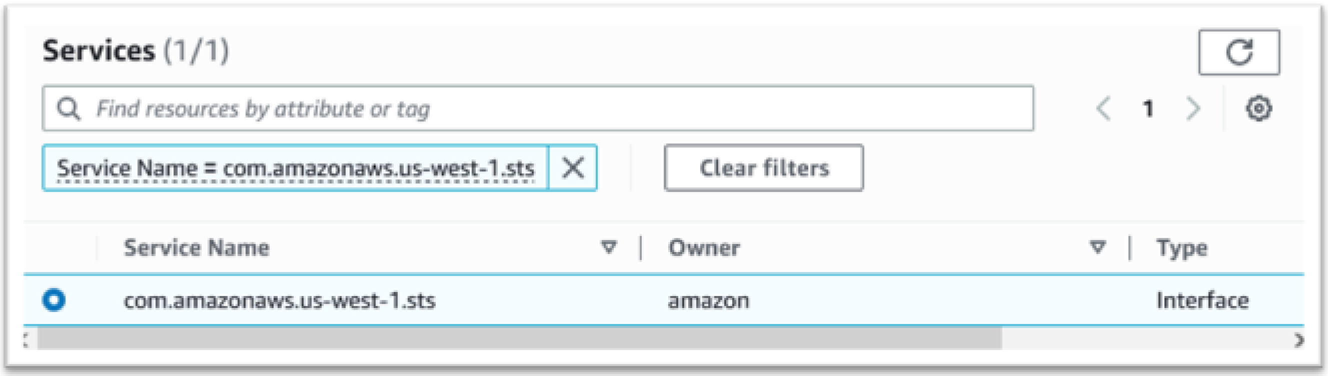
   - “com.amazonaws.us-west-1.ec2”
   - “com.amazonaws.us-west-1.sts”
   - “com.amazonaws.us-west-1.license-manager”

6. 选择匹配的接口服务。

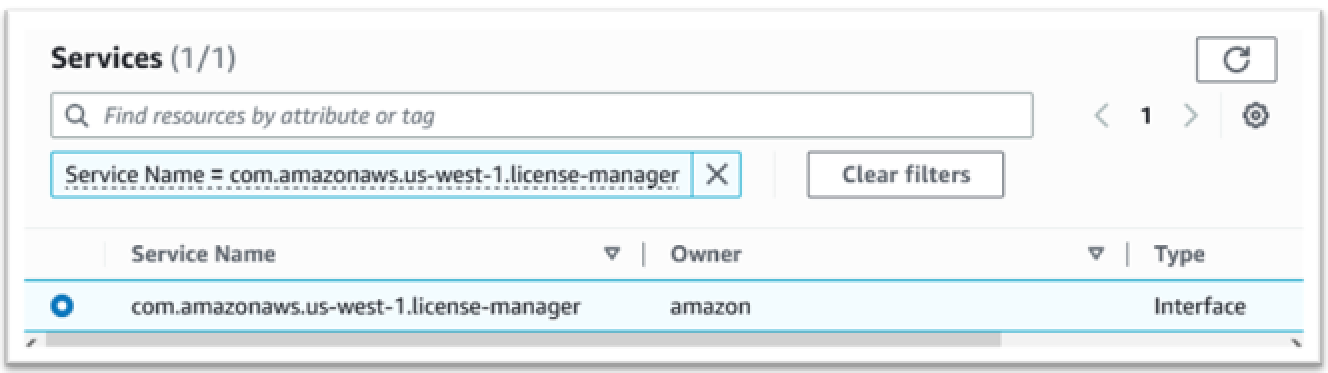
   com.amazonaws。 ***region***.ec2：



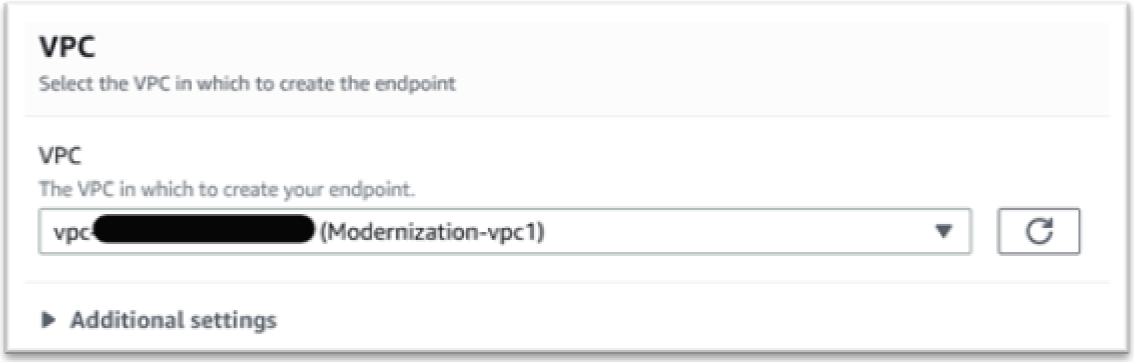
com.amazonaws。 ***region***.sts:



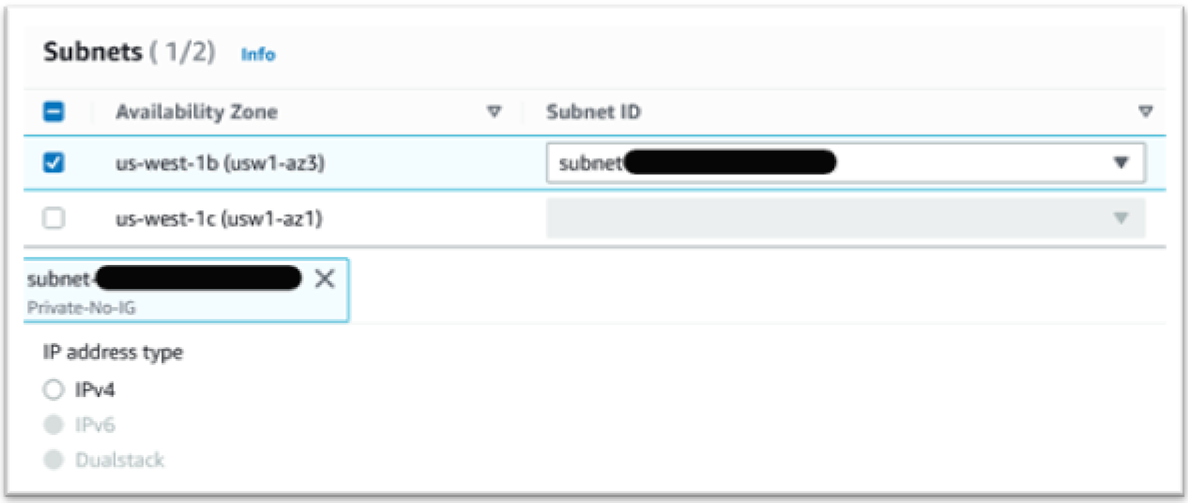
com.amazonaws。 ***region***.license-Manager：



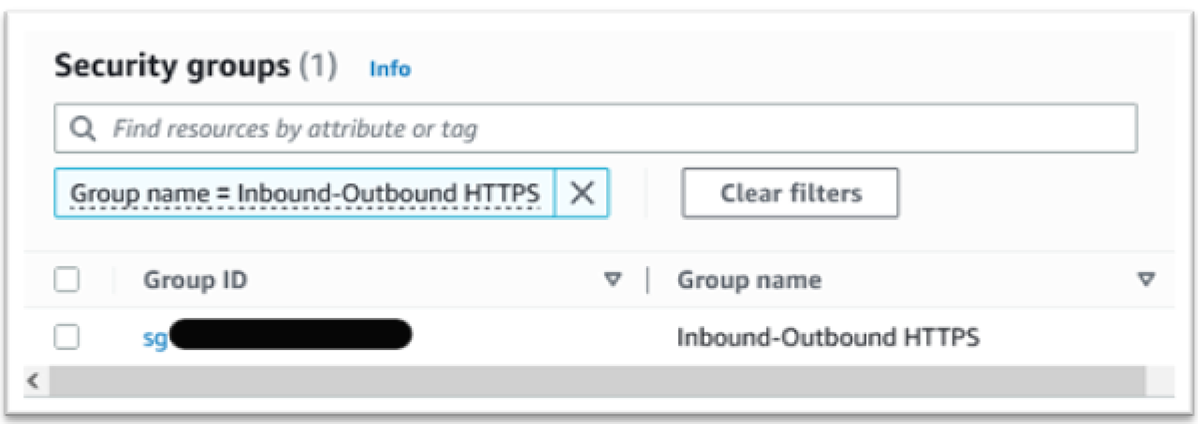
7. 对于 VPC，选择实例的 VPC。



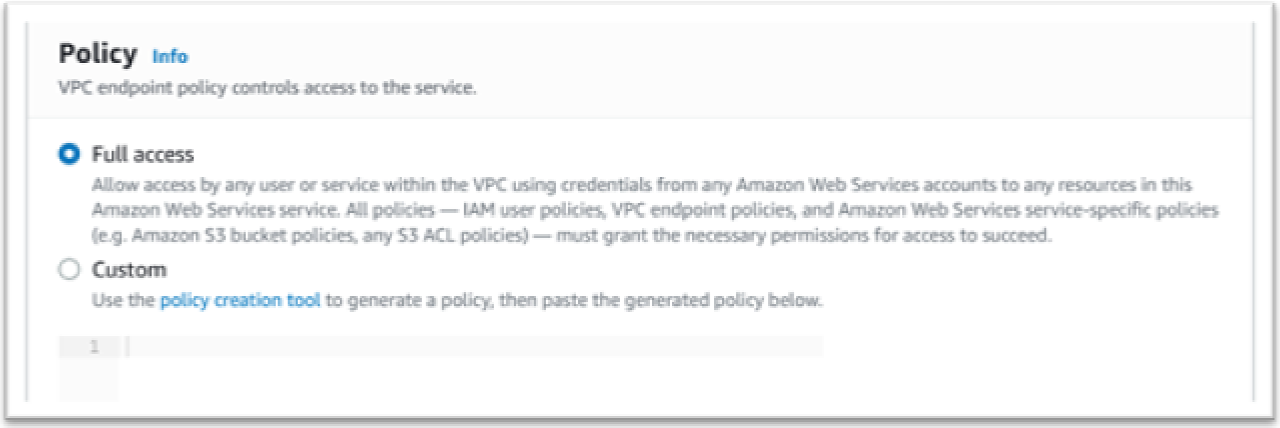
8. 为 VPC 选择可用区和子网。



9. 选择之前创建的安全组。



10. 在“策略”下，选择完全访问权限。



11. 选择创建端点。
12. 为其余接口重复此过程。

为 Rocket Enterprise Analyzer（前身为 Micro Focus）和 Rocket

您可以在会话开始和结束时自动运行脚本，从而实现特定于您的客户环境的自动化。有关此 WorkSpaces 应用程序功能的更多信息，请参阅《Amazon WorkSpaces 应用程序管理指南》中的 “使用会话脚本管理 AppStream 2.0 用户的流媒体体验”。

此特征要求您至少具有以下版本的 Enterprise Analyzer 和 Enterprise Developer 映像：

- `m2-enterprise-analyzer-v8.0.4.R1`
- `m2-enterprise-developer-v8.0.4.R1`

主题

- 设置会话开始时的自动化
- 设置会话结束时的自动化

设置会话开始时的自动化

如果要在用户连接到 WorkSpaces 应用程序时运行自动化脚本，请创建您的脚本并将其命名`m2-user-setup.cmd`。将脚本存储在用户的 “ WorkSpaces 应用程序主目录” 文件夹中。 AWS 大型机现代化提供的 WorkSpaces 应用程序映像在该位置查找具有该名称的脚本，如果存在，则运行该脚本。

> **Note**
>
> 脚本持续时间不能超过 WorkSpaces 应用程序设置的限制，目前为 60 秒。有关更多信息，请参阅《Amazon WorkSpaces 应用程序管理指南》中的 “在流式传输会话开始之前运行脚本”。

设置会话结束时的自动化

如果要在用户与 WorkSpaces 应用程序断开连接时运行自动化脚本，请创建您的脚本并命名它`m2-user-teardown.cmd`。将脚本存储在用户的 “ WorkSpaces 应用程序主目录” 文件夹中。 AWS 大型机现代化提供的 WorkSpaces 应用程序映像在该位置查找具有该名称的脚本，如果存在，则运行该脚本。

> **Note**
>
> 脚本持续时间不能超过 WorkSpaces 应用程序设置的限制，目前为 60 秒。有关更多信息，请参阅《Amazon WorkSpaces 应用程序管理指南》中的 “流式传输会话结束后运行脚本”。

在 Rocket Enterprise Developer（前 Micro Focus Enterprise

您可以使用 Rocket Software（前身为 Micro Focus）运行时访问在 AWS 大型机现代化中部署的大型机数据集。您可以从 Rocket Enterprise Developer 实例中以表和列的形式查看迁移的数据集。通过这种方式查看数据集，您可以：

- 对迁移的数据文件执行 `SQL SELECT` 操作。
- 在迁移的大型机应用程序外部披露数据，无需更改应用程序。
- 轻松筛选数据并保存为 CSV 或其他文件格式。

> **Note**
>
> 步骤 1 和 2 是一次性活动。针对每个数据集重复步骤 3 和 4，以便创建数据库视图。

主题

- 步骤 1：设置与 Rocket Software 数据存储库（亚马逊 RDS 数据库）的 ODBC 连接
- 步骤 2：创建 MFDBFH.cfg 文件
- 步骤 3：为 copybook 布局创建结构 (STR) 文件
- 步骤 4：使用结构 (STR) 文件创建数据库视图
- 步骤 5：以表格和列的形式查看 Rocket Software（以前是 Micro Focus）

先决条件

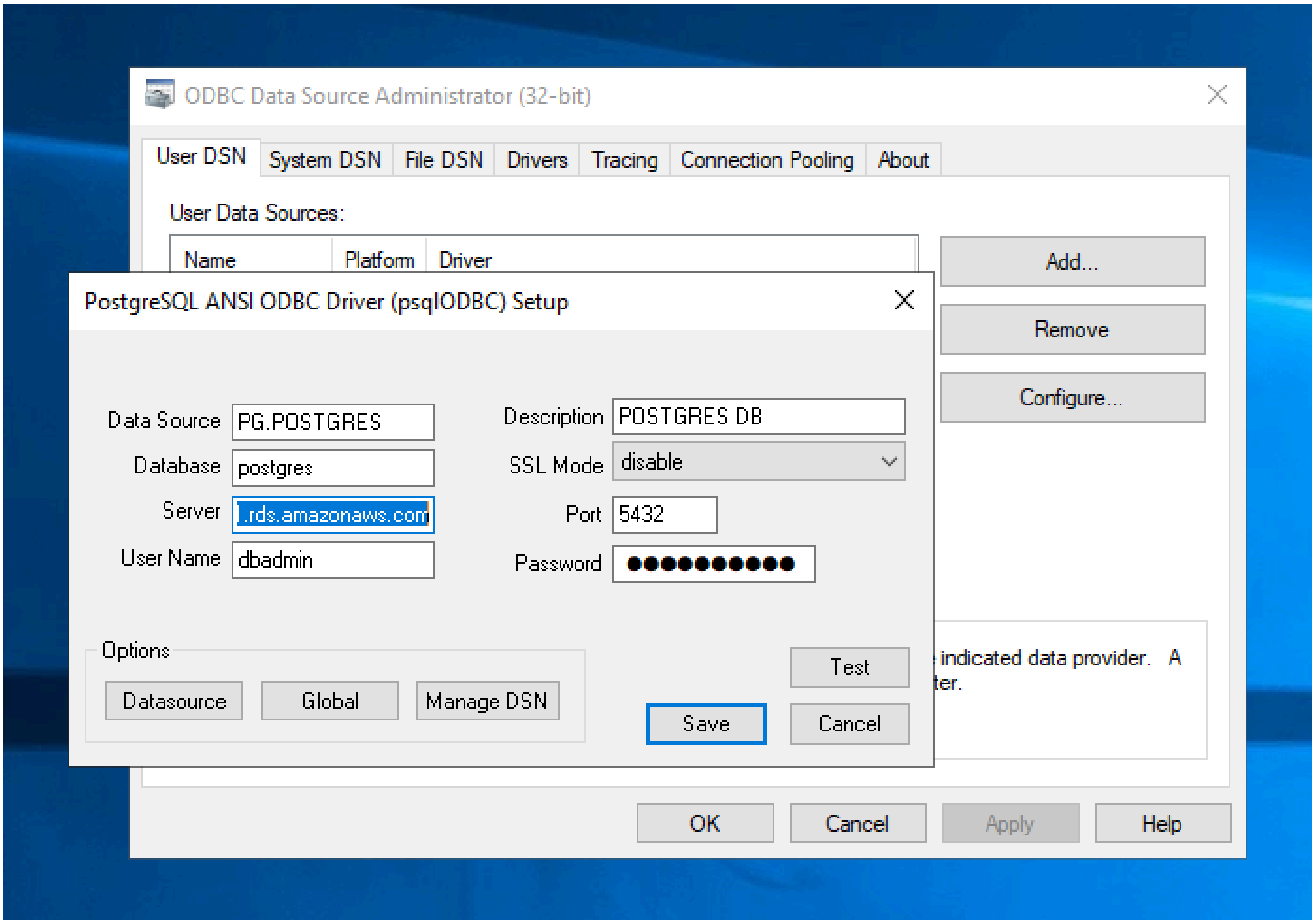
- 你必须能够通过 WorkSpaces 应用程序访问 Rocket Enterprise 开发者桌面。
- 您必须使用 Rocket Software 运行时引擎在 AWS 大型机现代化下部署和运行应用程序。
- 您将应用程序数据存储在 Aurora PostgreSQL 兼容版中。

步骤 1：设置与 Rocket Software 数据存储库（亚马逊 RDS 数据库）的 ODBC 连接

在此步骤中，设置与数据库的 ODBC 连接，该数据库包含要以表和列形式查看的数据。您只需执行此步骤一次。

1. 使用 WorkSpaces 应用程序直播 URL 登录 Rocket Enterprise 开发者桌面。
2. 打开 ODBC 数据来源管理器，选择用户 DSN，然后选择添加。
3. 在创建新数据来源中，选择 PostgreSQL ANSI，然后选择完成。
4. 通过提供必要的数据库信息为 `PG.POSTGRES` 创建数据来源，如下所示：

```
Data Source : PG.POSTGRES
Database    : postgres
Server      : rds_endpoint.rds.amazonaws.com
Port        : 5432
User Name   : user_name
Password    : user_password
```



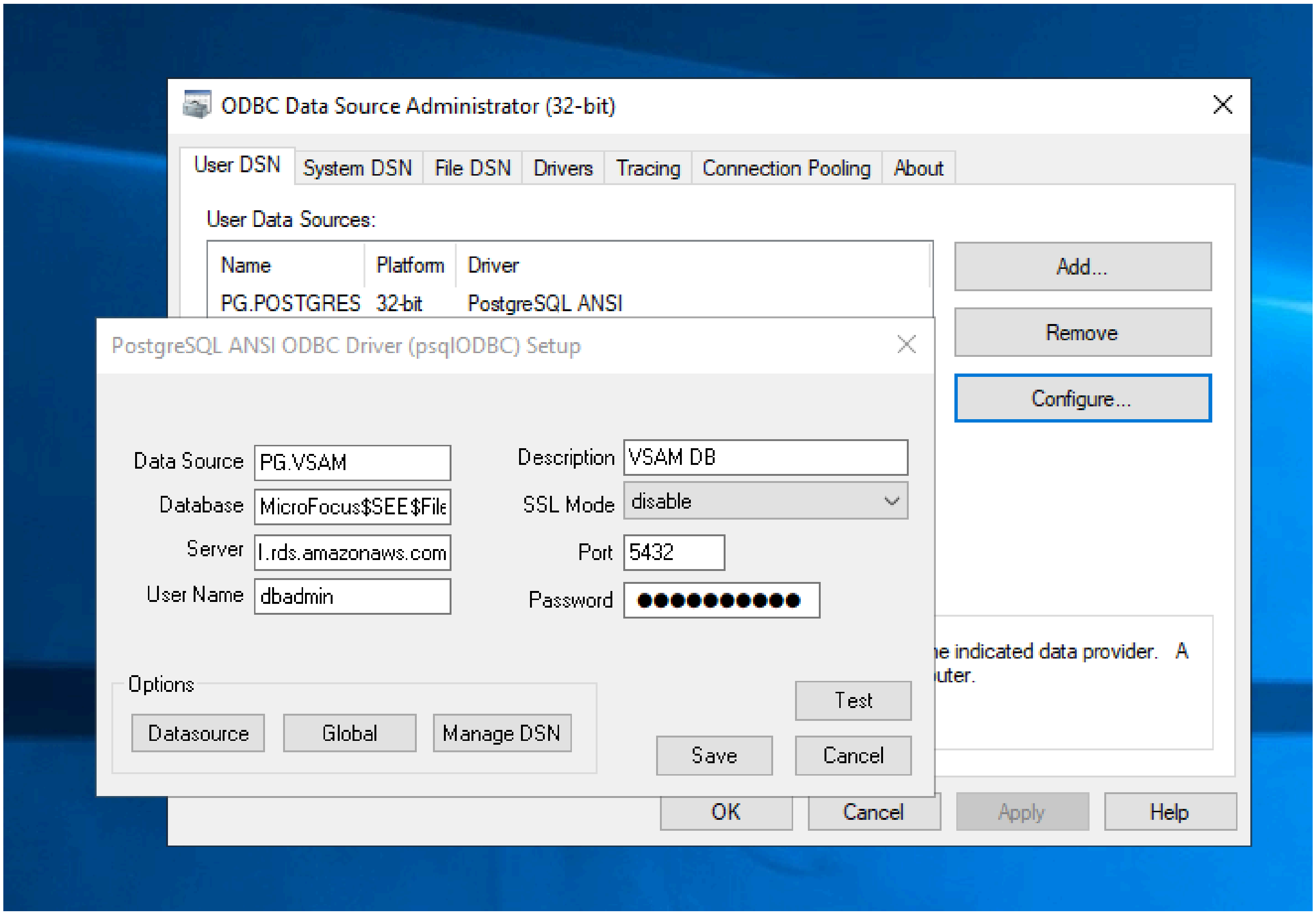
5. 选择测试，以便确保连接正常。如果测试成功，您应该会看到消息 `Connection successful`。

   如果测试不成功，请查看以下信息。

   - Amazon RDS 故障排除
   - 如何解决在连接我的 Amazon RDS 数据库实例时遇到的问题？

6. 保存数据来源。
7. 为 `PG.VSAM` 创建数据来源，测试连接，然后保存数据来源。提供以下数据库信息：

```
Data Source : PG.VSAM
Database    : MicroFocus$SEE$Files$VSAM
Server      : rds_endpoint.rds.amazonaws.com
Port        : 5432
User Name   : user_name
Password    : user_password
```



步骤 2：创建 MFDBFH.cfg 文件

在此步骤中，创建描述 Micro Focus 数据存储的配置文件。这是一个一次性配置步骤。

1. 在您的主文件夹（例如 `D:\PhotonUser\My Files\Home Folder\MFED\cfg\MFDBFH.cfg`）中，创建包含以下内容的 MFDBFH.cfg 文件。

```
<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. 通过运行以下命令查询 Micro Focus 数据存储来验证 MFDBFH 配置：

```
*###*
*## 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
```

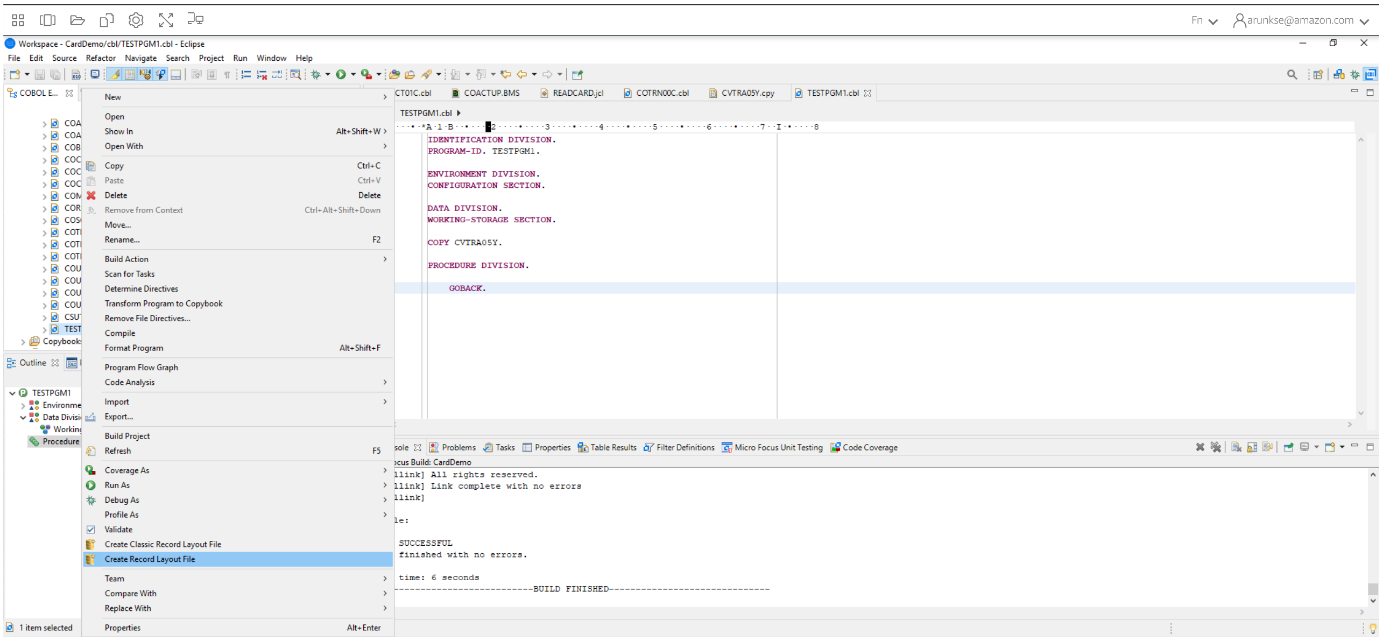
步骤 3：为 copybook 布局创建结构 (STR) 文件

在此步骤中，为 copybook 布局创建结构文件，以便以后使用该文件从数据集创建数据库视图。

1. 编译与您的 copybook 相关的程序。如果没有程序在使用 copybook，请创建并编译一个如下所示的简单程序，其中包含用于 copybook 的 COPY 语句。

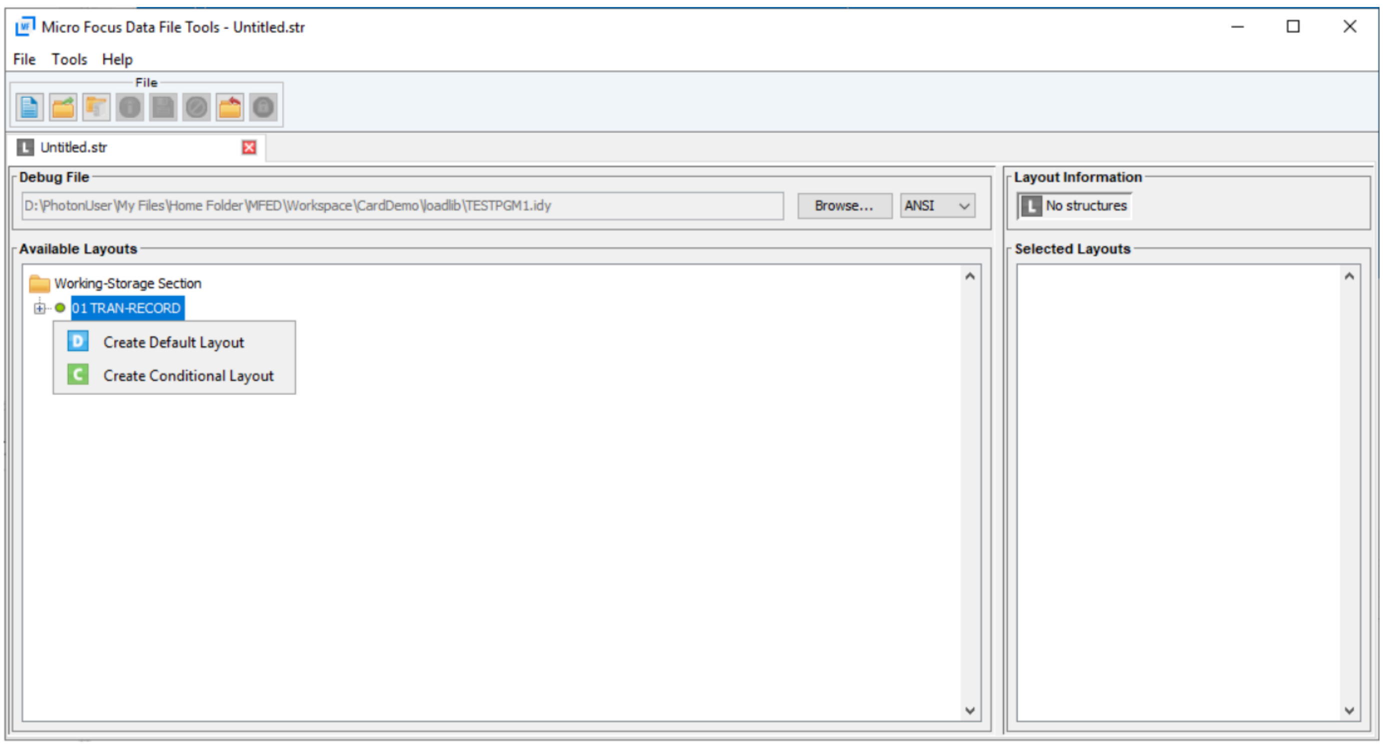
```
IDENTIFICATION DIVISION.
      PROGRAM-ID. TESTPGM1.

      ENVIRONMENT DIVISION.
      CONFIGURATION SECTION.

      DATA DIVISION.
      WORKING-STORAGE SECTION.

      COPY CVTRA05Y.

      PROCEDURE DIVISION.

      GOBACK.
```

2. 成功编译后，右键单击该程序并选择创建记录布局文件。这将使用编译期间生成的 .idy 文件打开 Micro Focus Data File Tools。

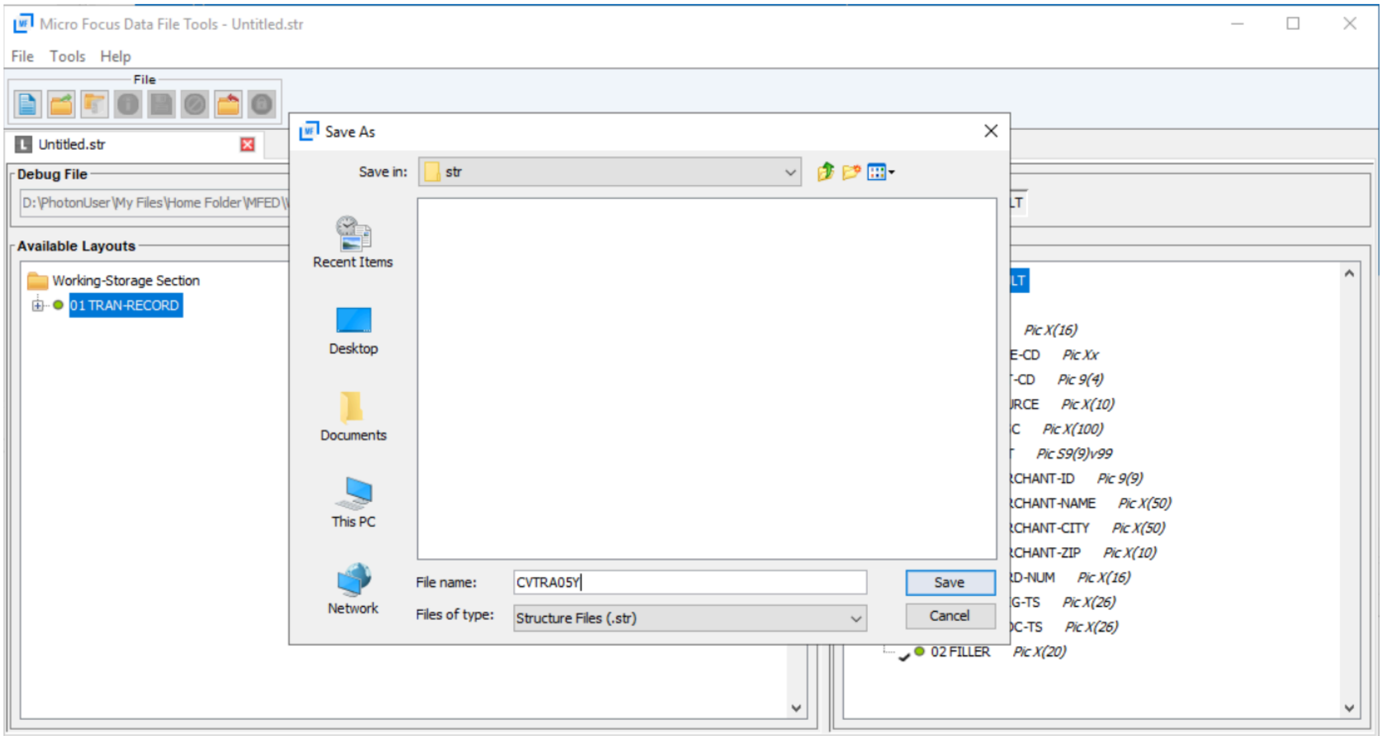


3. 右键单击记录结构，然后根据布局选择创建默认布局（单结构）或创建条件布局（多结构）。

   有关更多信息，请参阅 Micro Focus 文档中的[创建结构文件和布局](#)。



4. 创建布局后，从菜单中选择文件，然后选择另存为。浏览并将文件保存在主文件夹下，文件名与您的 copybook 相同。您可以选择创建名为 `str` 的文件夹，并将所有结构文件保存在此文件夹中。



步骤 4：使用结构 (STR) 文件创建数据库视图

在此步骤中，使用先前创建的结构文件为数据集创建数据库视图。

- 使用 dbfhview 命令为 Micro Focus 数据存储中已有的数据集创建数据库视图，如以下示例所示。

```
##
      ## 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
```

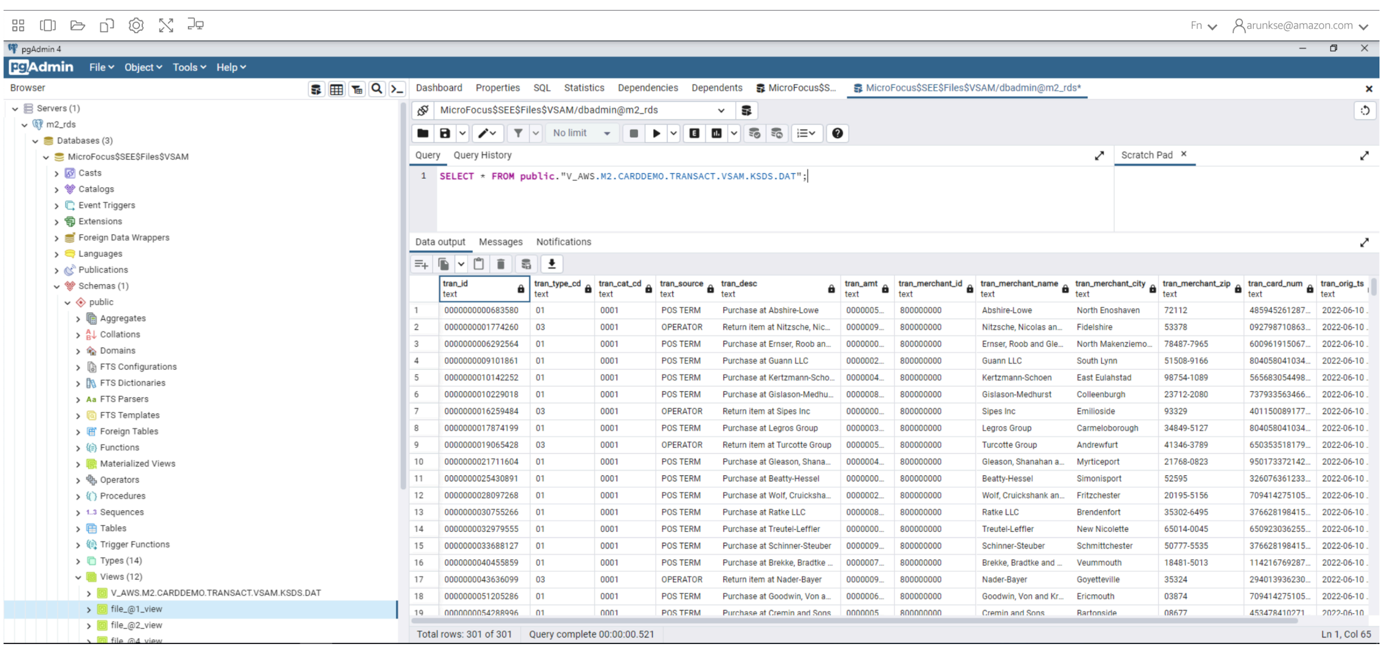
```
      Copyright (C) 1984-2022 Micro Focus. All rights reserved.

      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
```

步骤 5：以表格和列的形式查看 Rocket Software（以前是 Micro Focus）

在此步骤中，使用 pgAdmin 连接到数据库，以便您可以运行查询来以表和列的形式查看数据集。

- 使用 pgAdmin 连接到数据库 `MicroFocus$SEE$Files$VSAM` 并查询您在步骤 4 中创建的数据库视图。

```
SELECT * FROM public."V_AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS.DAT";
```



在企业开发人员中使用 Rocket Software（以前称为 Micro Focus）数据文件工具编辑数据

您可以使用 Rocket Software 运行时查看和编辑任何迁移数据集 AWS 的大型机现代化中的数据集。本文档中的步骤将指导您完成使用数据文件工具访问数据集的过程。
这使您可以根据需要查看和编辑迁移的数据集。

主题

- 先决条件
- 启动 Rocket 软件（前身为 Micro Focus）数据文件工具
- 编辑存储在 MFDBFH 数据库中的 VSAM 数据集
- 编辑存储在 MFDBFH 数据库中的非 vSAM 数据集
- 编辑存储在文件系统中的 VSAM 和非 vSAM 数据集 (EFS/FSx)

先决条件

在开始之前，必须部署包含数据集的应用程序 使用 Rocket Software 引擎在 AWS 大型机现代化服务下导入。

要继续编辑数据集，必须完成步骤 1，步骤 2， 以及（可选）the section called “在 Enterprise Developer 中以表的形式查看数据集”页面中的步骤 3 配置 ODBC 连接和 Micro Focus 数据存储库（即`MFDBFH`）。

> ⚠ Important
>
> 本指南假设你使用的是亚马逊 Aurora Postgres 作为 Micro Focus 数据存储 () `MFDBFH` 来存储您的应用程序数据。

启动 Rocket 软件（前身为 Micro Focus）数据文件工具

完成先决条件后，您可以通过设置`MFDBFH_CONFIG`环境变量来启动 Micro Focus 数据文件工具，以访问存储在数据库中的数据集（`MFDBFH`）。

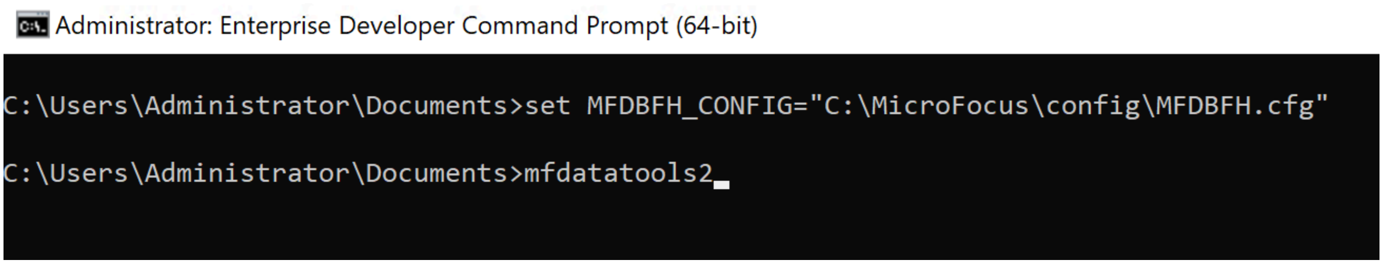
为此，请执行以下操作：

1. 登录 Micro Focus Enterprise Developer 桌面，然后从 “开始” 菜单中启动企业开发人员命令提示符（64 位）。
2. 使用`MFDBCH.cfg`文件的完整路径设置`MFDBFH_CONFIG`环境变量。

```
set MFDBFH_CONFIG="C:\MicroFocus\config\MFDBFH.cfg"
```

3. 使用以下命令从企业开发人员命令行启动 Micro Focus 数据文件工具。

```
mfdatatools2
```



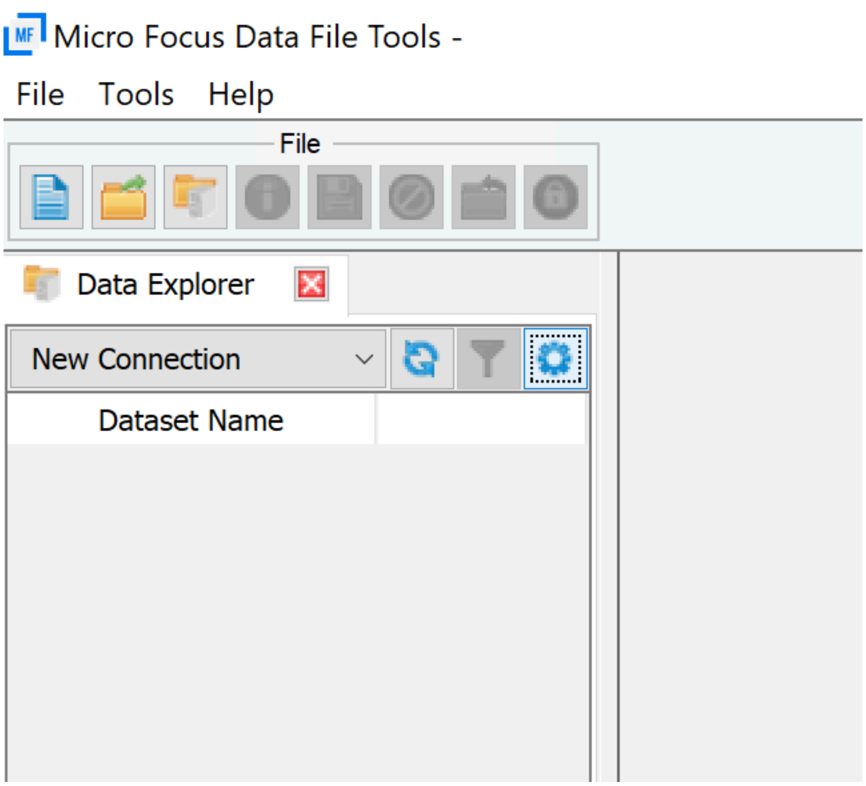
这将在单独的窗口中打开 Micro Focus 数据文件工具。

编辑存储在 MFDBFH 数据库中的 VSAM 数据集

启动 Micro Focus 数据文件工具后，即可打开存储在 Micro Focus 数据存储区中的 VSAM 数据集。

为此，请执行以下操作：

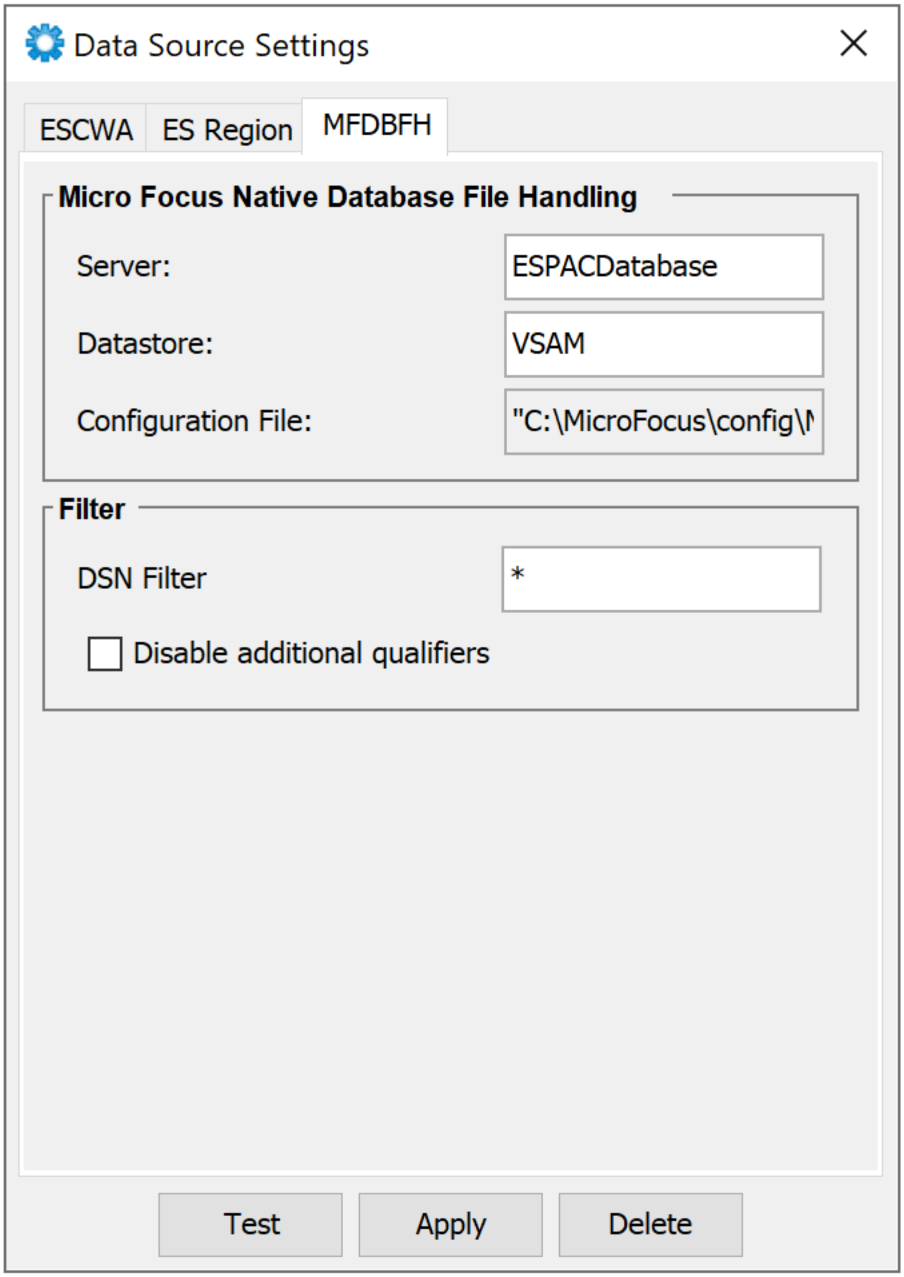
1. 从 Micro Focus 数据文件工具窗口的 “文件” 菜单中，选择 “数据资源管理器”。
2. 在数据资源管理器部分，选择设置（齿轮图标）以配置新连接。这将打开 “数据源设置” 窗口。



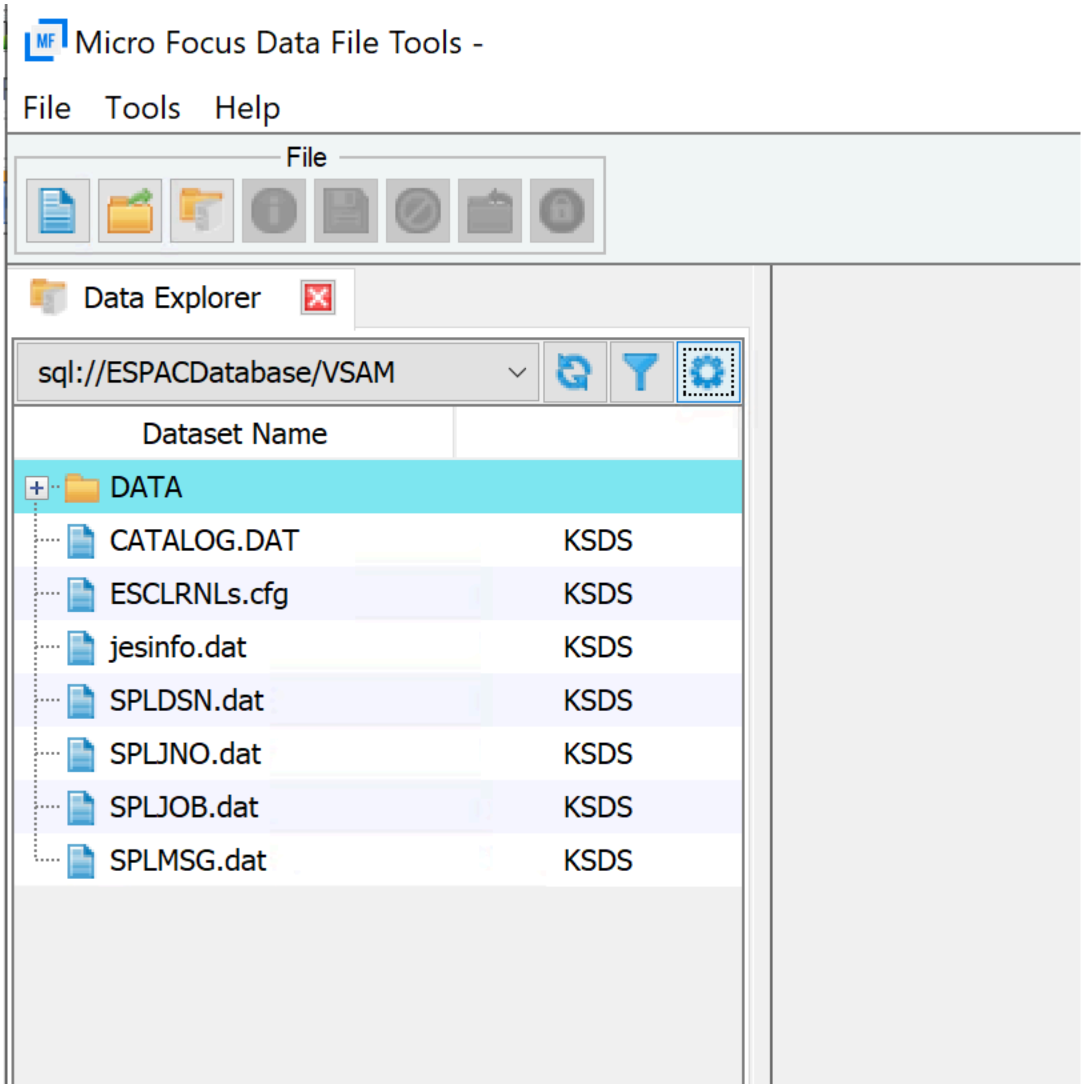
3. 在 “数据源设置” 窗口中，选择 MFDBFH 选项卡，然后输入以下值：

- Server：ESPACDatabase
- 数据存储：VSAM

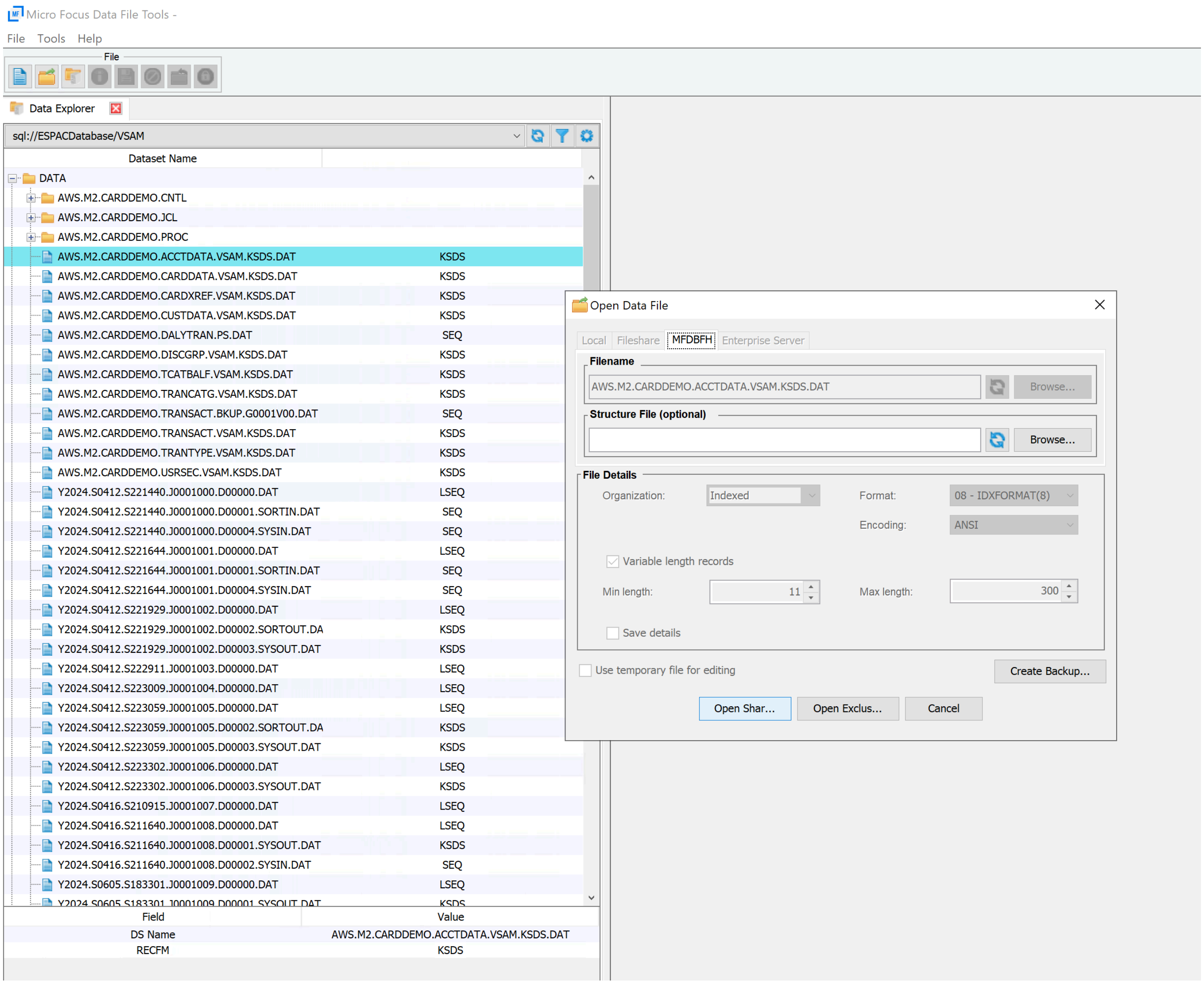
选择 “应用” 以保存配置。



现在，数据资源管理器会显示存储在中的所有数据集MFDBFH。



4. 展开相对路径DATA并双击要打开的 VSAM 数据集。
5. 在 “打开数据文件” 窗口中，选择 “打开共享文件” 或 “以独占方式打开” 以打开数据集。



现在，您可以查看或编辑打开的数据集。

编辑存储在 MFDBFH 数据库中的非 vSAM 数据集

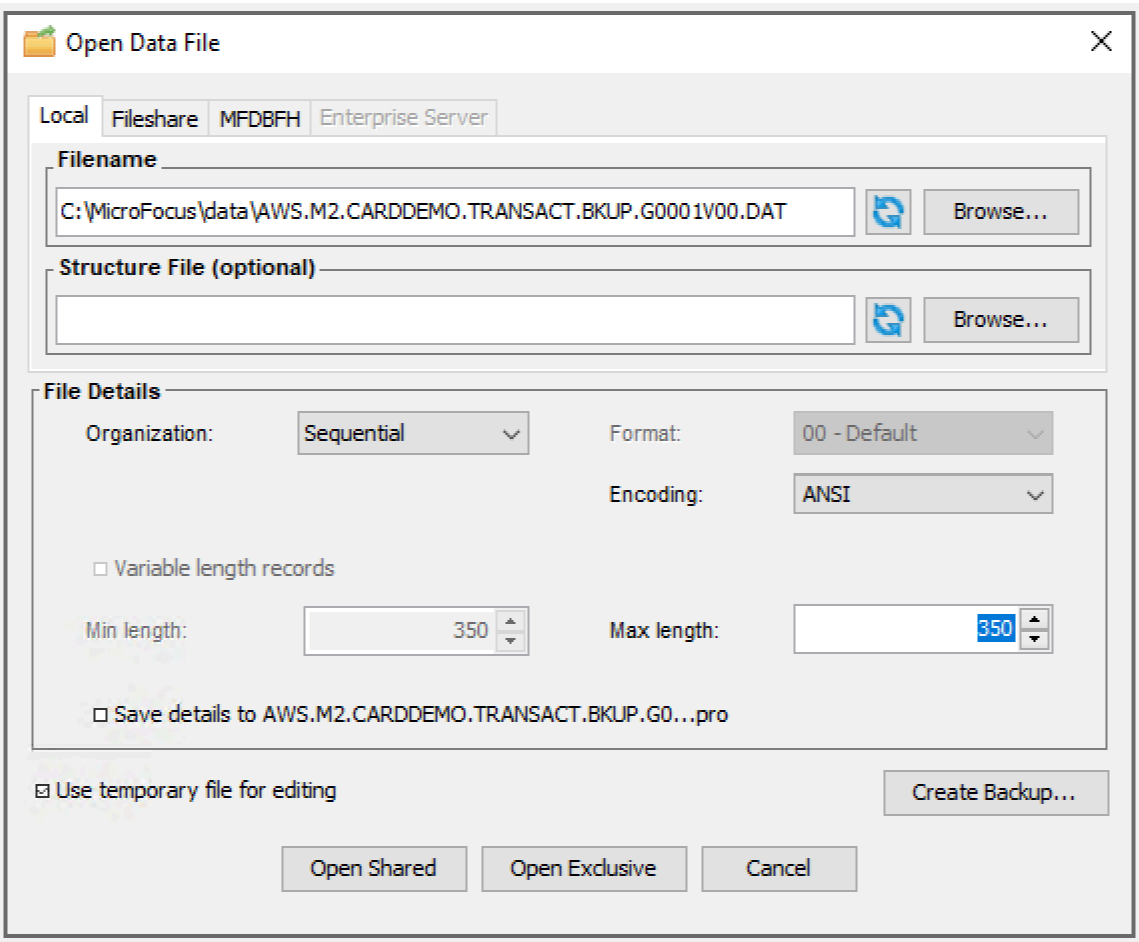
如果要编辑非 vSAM 数据集，可以打开存储在 Micro Focus 数据存储中的非 vSAM 数据集。

为此，请执行以下操作：

1. 在企业开发人员命令提示符（64 位）中运行`dbfhdeploy data extract`命令将非 vSAM 数据集下载到本地文件系统。

> **Note**
>
> 在运行此命令之前，请确保已将MFDBFH_CONFIG环境变量设置为MFDBFH.cfg文件的完整路径。
>
> ```
> dbfhdeploy data extract sql://ESPACDatabase/VSAM/
> AWS.M2.CARDDEMO.TRANSACT.BKUP.G0001V00.DAT?folder=/DATA C:\MicroFocus\data
> \AWS.M2.CARDDEMO.TRANSACT.BKUP.G0001V00.DAT
> ```

2. 从 “开始” 菜单中启动 Micro Focus 数据文件工具。
3. 从 Micro Focus 数据文件工具的 “文件” 菜单中，选择 “打开”，然后选择 “数据文件”。
4. 在 “打开数据文件” 窗口中，浏览本地文件系统中已下载的数据集。根据需要编辑 “文件详细信息”。然后选择 “打开共享” 或 “以独占方式打开” 以打开数据集。



现在，您可以查看或编辑打开的数据集。

可以使用 dbfhdeploy 命令行实用程序中的步骤或使用 dbfhdeploy 命令行实用程序将编辑the section called “为 应用程序导入数据集”或更新的数据集导回 Micro F ocus 数据存储区。

编辑存储在文件系统中的 VSAM 和非 vSAM 数据集 (EFS/FSx)

您也可以打开存储在文件系统中的数据集。

为此，请执行以下操作：

1. 装载 EFS/FSx 企业开发人员 EC2 实例上的文件系统。
2. 使用 Micro Focus 数据文件工具浏览并打开文件系统中的数据集。

火箭软件（前身为 Micro Focus）教程

本节中的教程可帮助您开始在 Rocket Software 运行时引擎中为 AWS 大型机现代化服务设置各种任务。这些教程用于设置示例应用程序、使用 Rocket Enterprise Developer 中的模板以及设置 Enterprise Analyzer。

主题

教程：为 BankDemo 示例应用程序设置 Rocket 软件（前身为 Micro Focus）版本

AWS 大型机现代化使您能够为迁移的应用程序设置构建和持续 integration/continuous 交付 (CI/CD) 管道。这些版本和管道使用 AWS CodeBuild AWS CodeCommit、和 AWS CodePipeline 来提供这些功能。 CodeBuild 是一项完全托管的生成服务，它可以编译源代码、运行单元测试并生成可随时部署的工件。 CodeCommit 是一项版本控制服务，使您能够在云端私下存储和管理 Git 存储库。 AWS CodePipeline 是一项持续交付服务，可让您对发布软件所需的步骤进行建模、可视化和自动化。

本教程演示 AWS CodeBuild 如何使用从 Amazon S3 编译 BankDemo 示例应用程序源代码，然后将编译后的代码导出回 Amazon S3。

AWS CodeBuild 是一项完全托管的持续集成服务，用于编译源代码、运行测试和生成随时可以部署的软件包。使用 CodeBuild，您可以使用预先打包的构建环境，也可以使用自己的构建工具创建自定义构建环境。此演示场景使用后者，它由一个使用预打包的 Docker 镜像的 CodeBuild 构建环境组成。

> **Important**
>
> 在启动大型机现代化项目之前，我们建议您了解适用于大型机的AWS 迁移加速计划 (MAP)，或者联系 AWS 大型机专家，了解实现大型机应用程序现代化所需的步骤。

主题

先决条件

开始本教程之前，请满足以下先决条件：

- 下载BankDemo 示例应用程序并将其解压缩到一个文件夹。源文件夹中包含 COBOL 程序和 Copybook 以及定义。此外，它还包含一个供参考的 JCL 文件夹，尽管您不需要构建 JCL。该文件夹还包含构建所需的元文件。
- 在 AWS 大型机现代化控制台中，选择工具。在分析、开发和构建资产中，选择使用我的 AWS 账户共享资产。

第 1 步：与 AWS 账户共享构建资产

在此步骤中，您需要确保与您的 AWS 账户共享构建资产，尤其是在使用资产的地区。

1. 打开 AWS 大型机现代化控制台，网址为https://console.aws.amazon.com/m2/。
2. 在左侧导航中，选择工具。
3. 在分析、开发和生成资产中，选择与我的 AWS 账户共享资产。

> **Important**
>
> 您需要在要进行构建的每个 AWS 区域执行一次此步骤。

步骤 2：创建 Amazon S3 存储桶

在此步骤中，您将创建两个 Amazon S3 存储桶。一个是用于存放源代码的输入存储桶，另一个是用于存放构建输出的输出存储桶。有关更多信息，请参阅《Amazon S3 用户指南》中的创建、配置和使用 Amazon S3 存储桶。

1. 要创建输入存储桶，登录 Amazon S3 控制台并选择创建存储桶。
2. 在常规配置中，提供存储桶的名称并指定要 AWS 区域 在哪里创建存储桶。示例名称是`codebuild-regionId-accountId-input-bucket`， AWS 区域 其中`regionId`是存储桶的，`accountId`是您的 AWS 账户 ID。

   > **Note**
   >
   > 如果您要在与美国东部（弗吉尼亚北部）不同的 AWS 区域 位置创建存储桶，请指定`LocationConstraint`参数。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的创建存储桶。

3. 保留所有其他设置并选择创建存储桶。
4. 重复步骤 1 – 3 来创建输出存储桶。例如，名称是`codebuild-regionId-accountId-output-bucket`， AWS 区域 其中`regionId`是存储桶的，`accountId`是您的 AWS 账户 ID。

   无论您为这些存储桶选择的名称如何，请务必在本教程中保持统一。

步骤 3：创建构建规范文件

在此步骤中，创建一个构建规范文件。此文件以 YAML 格式提供构建命令和相关设置， CodeBuild 用于运行构建。有关更多信息，请参阅《AWS CodeBuild 用户指南》 CodeBuild中的构建规范参考。

1. 在您在先决条件中解压缩的目录中创建一个名为 `buildspec.yml` 的文件。
2. 将以下内容添加到该文件并保存。无需对该文件进行任何更改。

```
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/**
```

这里`CODEBUILD_BUILD_ID`、`CODEBUILD_BUILD_ARN$CODEBUILD_SRC_DIR/source`、和`$CODEBUILD_SRC_DIR/target`是里面可用的环境变量 CodeBuild。有关更多信息，请参阅构建环境中的环境变量。

此时，您的目录结构应如下所示。

```
(root directory name)
    |-- build.xml
```

```
    |-- buildspec.yml
    |-- LICENSE.txt
    |-- source
        |... etc.
```

3. 将文件夹的内容压缩到名为 `BankDemo.zip` 的文件中。在本教程中，您无法压缩文件夹。因此，将文件夹的内容压缩到文件 `BankDemo.zip` 中。

步骤 4：上传源文件

在此步骤中，您将 BankDemo 示例应用程序的源代码上传到您的 Amazon S3 输入存储桶。

1. 登录到 Amazon S3 控制台，并在左侧导航窗格中，选择存储桶。然后选择您之前创建的输入存储桶。
2. 在对象下，选择上传。
3. 在文件和文件夹部分，选择添加文件。
4. 导航到并选择 `BankDemo.zip` 文件。
5. 选择上传。

步骤 5：创建 IAM 策略

在此步骤中，创建两个 IAM 策略。一项策略授予 AWS 大型机现代化访问和使用包含 Rocket Software 构建工具的 Docker 镜像的权限。此策略不是为客户自定义的。另一项策略授予 AWS 大型机现代化与输入和输出存储桶以及生成的 A mazon CloudWatch 日志进行交互的权限。 CodeBuild

要了解有关创建 IAM 策略的信息，请参阅《IAM 用户指南》中的编辑 IAM 策略。

创建允许访问 Docker 映像的策略

1. 在 IAM 控制台中，将以下策略文档复制并粘贴到策略编辑器中。

   JSON

```
{
    "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-*-<region>-prod"
        }
    ]
}
```

2. 提供策略名称，例如 m2CodeBuildPolicy。

创建允许 AWS 大型机现代化与存储桶和日志交互的策略

1. 在 IAM 控制台中，将以下策略文档复制并粘贴到策略编辑器中。请务必更新regionId到 AWS 区域，并更新accountId到您的 AWS 账户。
2. 提供策略名称，例如 BankdemoCodeBuildRolePolicy。

步骤 6：创建 IAM 角色

在此步骤中，在 CodeBuild 将之前创建的 IAM 策略与该新 IAM 角色关联后，您将创建一个允许您与 AWS 资源交互的新 IAM 角色。

有关创建服务角色的信息，请参阅 IAM 用户指南中的创建角色以向 AWS 服务委派权限。

1. 登录 IAM 控制台，然后在左侧导航窗格中选择角色。
2. 选择创建角色。

3. 在可信实体类型下，选择 AWS 服务。
4. 在其他 AWS 服务的使用案例下 CodeBuild，选择，然后CodeBuild再次选择。
5. 选择下一步。
6. 在添加权限页面上，选择下一步。您稍后给该角色分配策略。
7. 在角色详细信息下，提供角色的名称，例如 BankdemoCodeBuildServiceRole。
8. 在选择可信实体下，验证策略文档是否如下所示：

   JSON

```
{
            "Version":"2012-10-17",
            "Statement": [
              {
                "Effect": "Allow",
                "Principal": {
                  "Service": "codebuild.amazonaws.com"
                },
                "Action": "sts:AssumeRole"
              }
            ]
          }
```

9. 选择创建角色。

步骤 7：将 IAM 策略附加到 IAM 角色

在此步骤中，将您之前创建的两个 IAM 策略附加到 BankdemoCodeBuildServiceRole IAM 角色。

1. 登录到 IAM 控制台，然后在左侧导航窗格中，选择角色。
2. 在角色中，选择您之前创建的角色，例如 BankdemoCodeBuildServiceRole。
3. 在权限策略中，依次选择添加权限和附加策略。
4. 在其他权限策略中，选择您之前创建的策略，例如 m2CodeBuildPolicy 和 BankdemoCodeBuildRolePolicy。
5. 选择附加策略。

第 8 步：创建 CodeBuild 项目

在此步骤中，您将创建 CodeBuild 项目。

1. 登录 CodeBuild 控制台并选择 “创建构建项目”。
2. 在项目配置部分中，提供项目的名称，例如 `codebuild-bankdemo-project`。
3. 在源部分中，对于源提供商，选择 Amazon S3，然后选择您之前创建的输入存储桶，例如`codebuild-regionId-accountId-input-bucket`。
4. 在 S3 对象密钥或 S3 文件夹字段中，输入您上传到 S3 存储桶的 zip 文件的名称。在本例中，此文件名为 `bankdemo.zip`。
5. 在环境部分中，选择自定义映像。
6. 在环境类型字段中，选择 Linux。
7. 在映像注册表中，选择其他注册表。
8. 在外部注册表 URL 字段中，
   - 对于火箭软件 v9：输入`673918848628.dkr.ecr.us-west-1.amazonaws.com/m2-enterprise-build-tools:9.0.7.R1`。如果您在 Rocket Software v9 <m2-region>中使用其他 AWS AWS 区域 `673918848628.dkr.ecr.<m2-region>.amazonaws.com/m2-enterprise-build-tools:9.0.7.R1`，则还可以指定提供 AWS 大型机现代化服务的区域（例如，`eu-west-3`）。
   - 对于 Rocket 软件版本 8：输入 `673918848628.dkr.ecr.us-west-2.amazonaws.com/m2-enterprise-build-tools:8.0.9.R1`
   - 对于 Rocket 软件 v7：输入 `673918848628.dkr.ecr.us-west-2.amazonaws.com/m2-enterprise-build-tools:7.0.R10`
9. 在服务角色下，选择现有服务角色，然后在角色 ARN 字段中，选择您之前创建的服务角色，例如 `BankdemoCodeBuildServiceRole`。
10. 在构建规范部分中，选择使用构建规范文件。
11. 在构件部分的类型下，选择 Amazon S3，然后选择您的输出存储桶，例如 `codebuild-regionId-accountId-output-bucket`。
12. 在名称字段中，输入存储桶中要包含构建输出构件的文件夹的名称，例如 `bankdemo-output.zip`。
13. 在构件打包下，选择 Zip。
14. 选择 Create build project（创建构建项目）。

步骤 9：启动构建

在此步骤中，启动构建。

1. 登录 CodeBuild 控制台。
2. 在左侧导航窗格中，选择构建项目。
3. 选择您之前创建的构建项目，例如 `codebuild-bankdemo-project`。
4. 选择启动构建。

此命令开始构建。构建以异步方式运行。此命令的输出是一个包含属性 ID 的 JSON。此属性 id是对您刚刚启动的版本的构 CodeBuild 建 ID 的引用。您可以在 CodeBuild 控制台中查看版本的状态。您还可以在控制台中查看有关构建执行的详细日志。有关更多信息，请参阅《AWS CodeBuild 用户指南》中的查看详细构建信息。

当前阶段完成后，即表示您的构建已成功完成，并且编译的构件已在 Amazon S3 上准备就绪。

步骤 10：下载输出构件

在此步骤中，从 Amazon S3 下载输出构件。Rocket Software 编译工具可以创建几种不同的可执行文件类型。在本教程中，它将生成共享对象。

1. 登录 Amazon S3 控制台。
2. 在存储桶 role="bold"> 部分中，选择您的输出存储桶的名称，例如 `codebuild-regionId-accountId-output-bucket`。
3. 选择下载 role="bold">。
4. 解压下载的文件。导航到目标文件夹以查看构建构件。其中包括 `.so` Linux 共享对象。

清理 资源

如果您不再需要为本教程创建的资源，请将其删除，以免产生额外费用。为此，请完成以下步骤：

- 删除您为本教程创建的 S3 存储桶。有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的删除存储桶。
- 删除您为本教程创建的 IAM 策略。有关更多信息，请参阅《IAM 用户指南》中的删除 IAM 策略。
- 删除您为本教程创建的 IAM 角色。有关更多信息，请参阅《IAM 用户指南》中的删除角色或实例配置文件。

- 删除您为本教程创建的 CodeBuild 项目。有关更多信息，请参阅《AWS CodeBuild 用户指南》CodeBuild中的删除构建项目。

教程：设置 WorkSpaces 应用程序以用于 Rocket Enterprise Analyzer 和 Rocket

AWS 大型机现代化通过 Amazon WorkSpaces 应用程序提供了多种工具。 WorkSpaces Applications 是一项完全托管的安全应用程序流服务，它允许您在不重写应用程序的情况下将桌面应用程序流式传输给用户。 WorkSpaces 应用程序使用户可以即时访问他们需要的应用程序，并在他们选择的设备上提供响应灵敏、流畅的用户体验。使用 WorkSpaces 应用程序托管特定于运行时引擎的工具，使客户应用程序团队能够直接通过其网络浏览器使用这些工具，与存储在 Amazon S3 存储桶或存储库中的应用程序文件进行交互。 CodeCommit

有关 WorkSpaces 应用程序中浏览器支持的信息，请参阅《亚马逊 WorkSpaces 应用程序管理指南》中的 “系统要求和功能支持（Web 浏览器）”。如果您在使用 WorkSpaces 应用程序时遇到问题，请参阅《Amazon WorkSpaces 应用程序管理指南》中的 AppStream2.0 用户问题疑难解答。

本文档适用于客户运营团队的成员。它描述了如何设置 Amazon A WorkSpaces pplications 队列和堆栈来托管用于 AWS 大型机现代化的火箭企业分析器和火箭企业开发者工具。Rocket Enterprise Analyzer 通常在评估阶段使用，Rocket Enterprise Developer 通常在 AWS 大型机现代化方法的迁移和现代化阶段使用。如果您计划同时使用 Enterprise Analyzer 和 Enterprise Developer，则必须为每个工具创建单独的实例集和堆栈。每个工具都需要各自的实例集和堆栈，因为它们的许可条款不同。

> **Important**
>
> 本教程中的步骤基于可下载的 CloudFormation 模板 cfn-m2-.yml。appstream-fleet-ea-ed

主题

- 清理资源

先决条件

- 下载模板：cfn-m2-appstream-fleet-ea-ed .yml。
- 获取您的默认 VPC 的 ID 和安全组。有关默认 VPC 的更多信息，请参阅 Amazon VPC 用户指南 VPCs中的默认。有关默认安全组的更多信息，请参阅 Amazon EC2 用户指南中的默认和自定义安全组。
- 请确保您具有以下权限：
  - 在应用程序中创建堆栈、队列和用户。 WorkSpaces
  - CloudFormation 使用模板创建堆栈。
  - 在 Amazon S3 中创建存储桶和并将文件上传到存储桶
  - 从 IAM 下载凭证（`access_key_id` 和 `secret_access_key`）。

步骤 1：获取 WorkSpaces 应用程序镜像

在此步骤中，您将与您的 AWS 账户共享 Enterprise Analyzer 和企业开发人员的 WorkSpaces 应用程序映像。

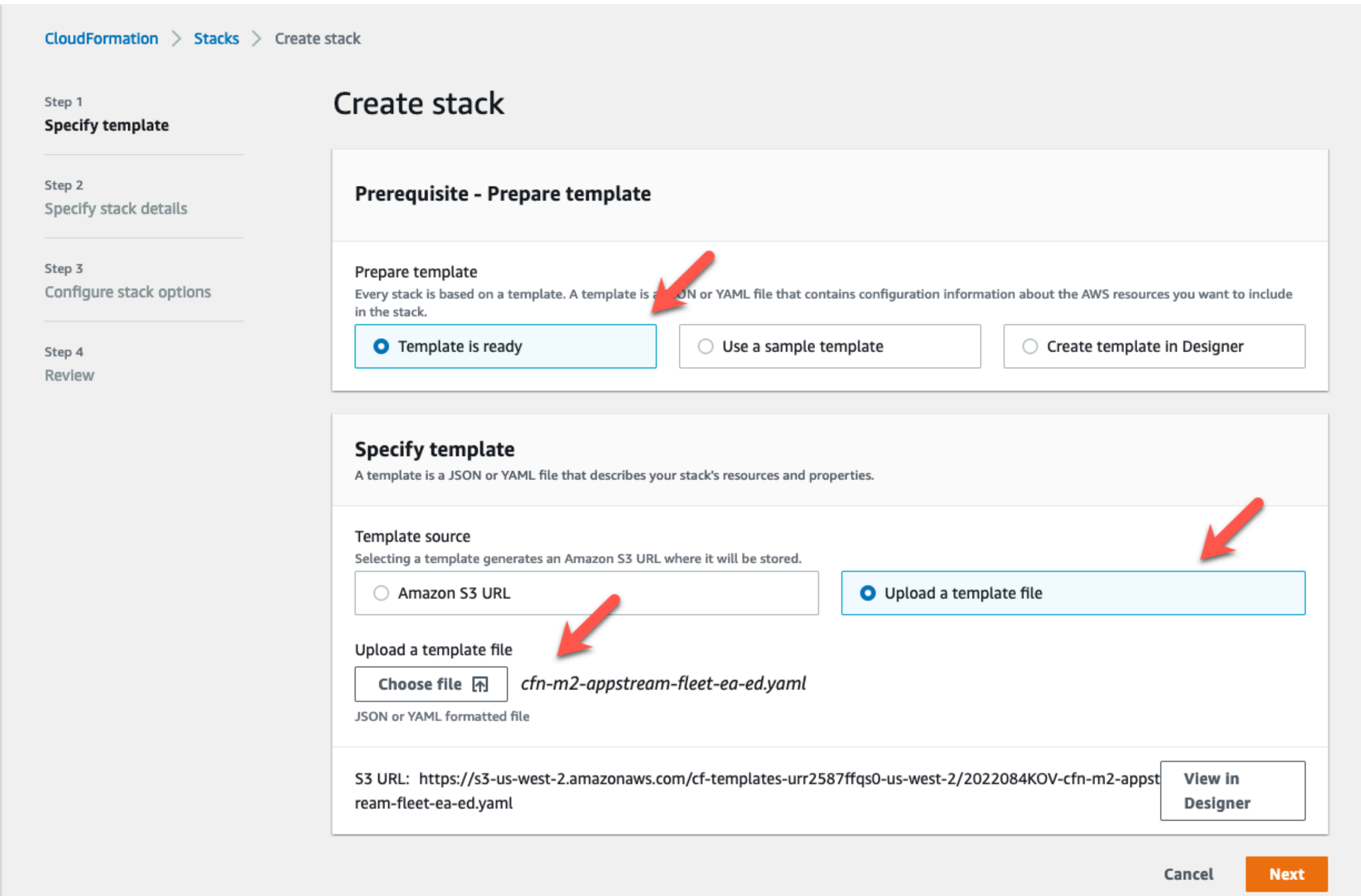
1. 打开 AWS 大型机现代化控制台，网址为https://console.aws.amazon.com/m2/。
2. 在左侧导航中，选择工具。
3. 在分析、开发和生成资产中，选择与我的 AWS 账户共享资产。

步骤 2：使用 CloudFormation 模板创建堆栈

在此步骤中，您将使用下载的 CloudFormation 模板创建用于运行 Rocket Enterprise Analyzer 的 WorkSpaces 应用程序堆栈和队列。您可以稍后重复此步骤来创建另一个 WorkSpaces 应用程序堆栈和队列来运行 Rocket Enterprise Developer，因为每个工具都需要自己的队列和 WorkSpaces 应用程序中的堆栈。有关 CloudFormation 堆栈的更多信息，请参阅《AWS CloudFormation 用户指南》中的使用堆栈。

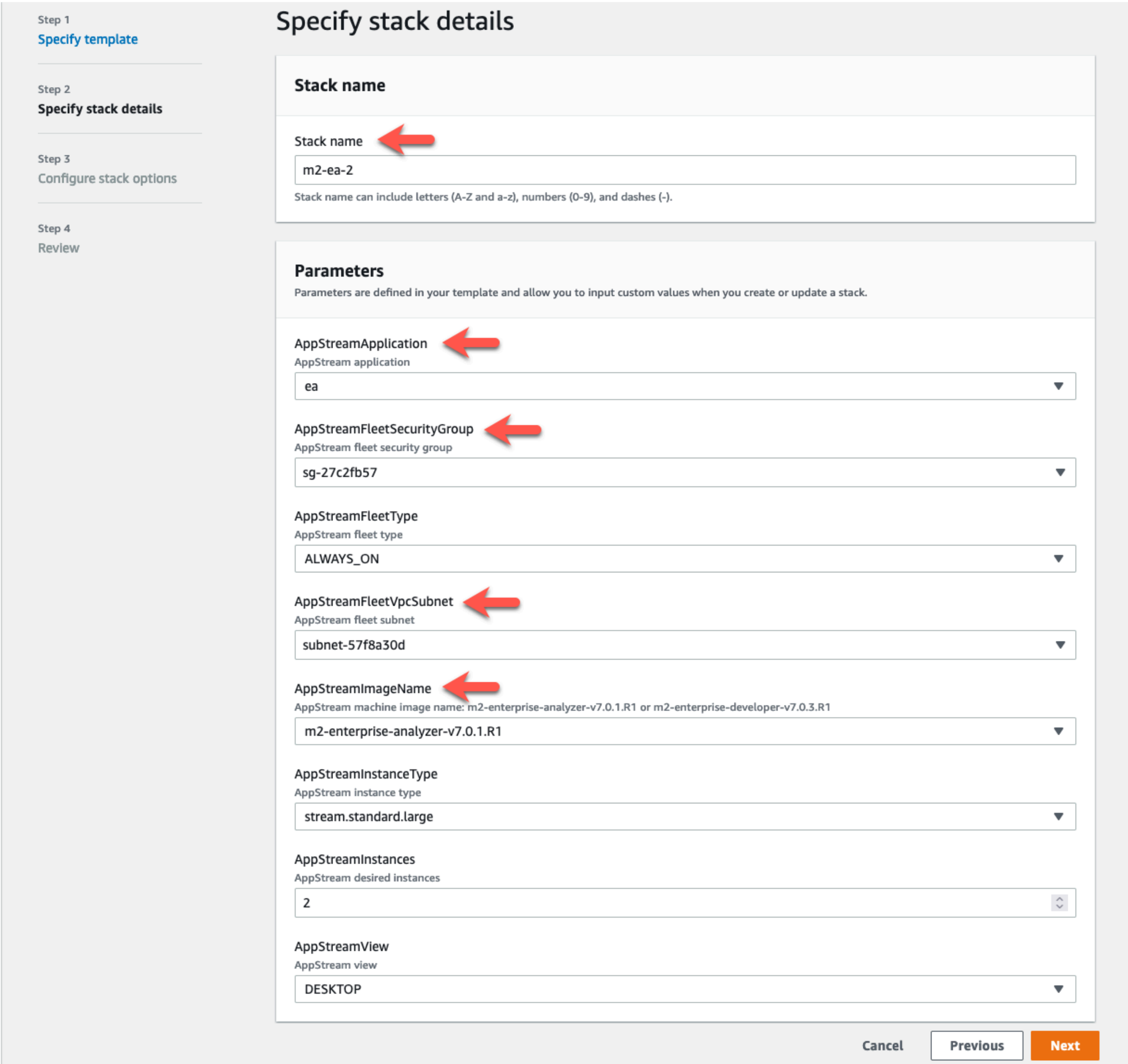
> **Note**
>
> AWS 大型机现代化在企业分析器和企业开发人员使用的标准 WorkSpaces 应用程序定价基础上增加了额外费用。有关更多信息，请参阅 AWS Mainframe Modernization 定价。

1. 如有必要，请下载 cfn-m2-appstream-fleet-ea-ed .yml 模板。
2. 打开 CloudFormation 控制台，选择创建堆栈并使用新资源（标准）。
3. 在先决条件 – 准备模板中，选择模板已就绪。
4. 在指定模板中，选择上传模板文件。
5. 在上传模板文件中，选择选择文件并上传 cfn-m2-appstream-fleet-ea-ed .ym l 模板。
6. 选择下一步。



7. 在指定堆栈集详细信息页面上，输入以下信息：

   - 在堆栈名称中，输入您所选的名称。例如 **m2-ea**。
   - 在中 AppStreamApplication，选择 ea。
   - 在中 AppStreamFleetSecurityGroup，选择默认 VPC 的默认安全组。
   - 在中 AppStreamFleetVpcSubnet，选择默认 VPC 内的子网。
   - 在中 AppStreamImageName，选择以开头的图像`m2-enterprise-analyzer`。此镜像包含当前支持的 Rocket Enterprise Analyzer 工具版本。
   - 接受其他字段的默认值，然后选择下一步。



8. 接受所有默认值，然后选择下一步。
9. 在审核页面上，确保所有参数都符合您的预期。
10. 滚动到底部，选择我确认 AWS CloudFormation 可能使用自定义名称创建 IAM 资源，然后选择创建堆栈。

创建堆栈和实例集需要 20 到 30 分钟。您可以选择 “刷新” 以在 CloudFormation 事件发生时查看事件。

步骤 3：在 “ WorkSpaces 应用程序” 中创建用户

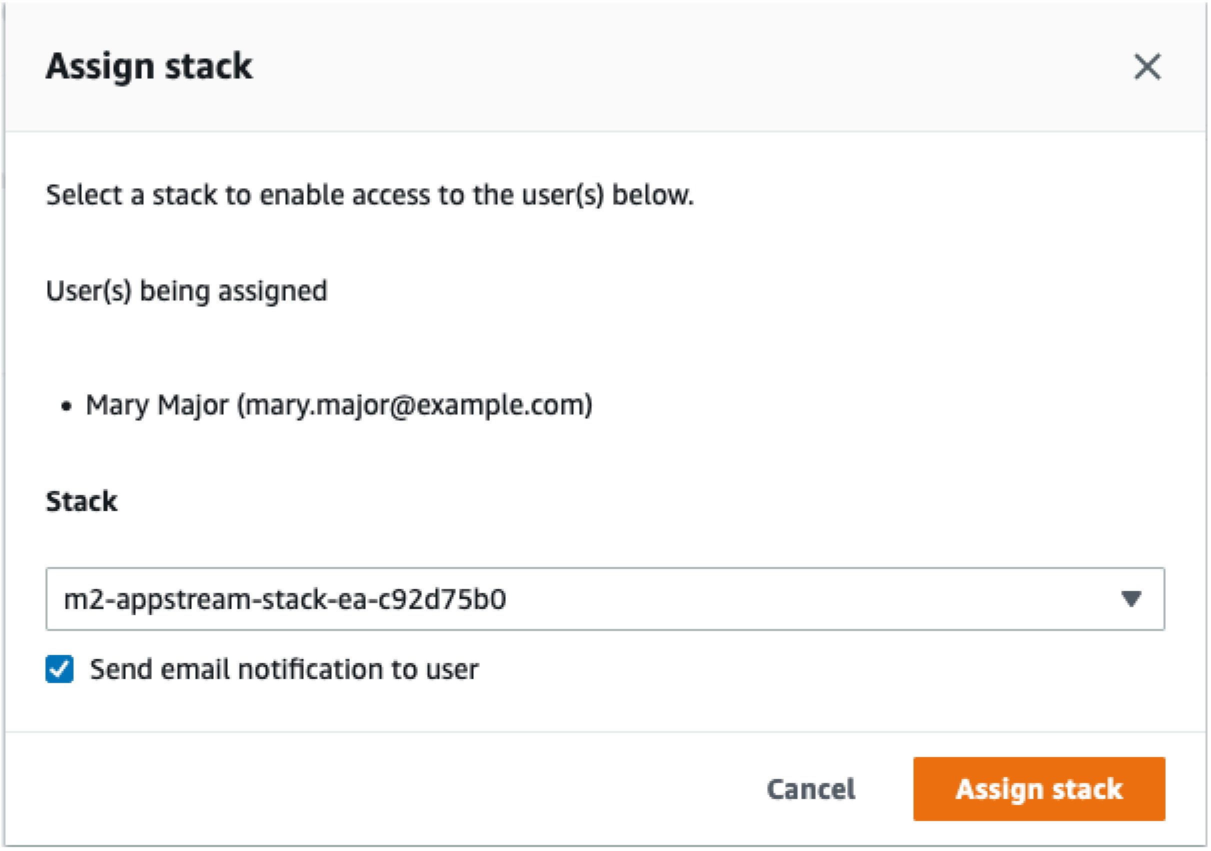
在等待 CloudFormation 堆栈创建完成时，可以在 WorkSpaces 应用程序中创建一个或多个用户。这些用户是将在 WorkSpaces 应用程序中使用企业分析器的用户。您需要为每个用户指定一个电子邮件地址，并确保每个用户都有足够的权限在 Amazon S3 中创建存储桶、将文件上传到存储桶以及链接到存储桶以映射其内容。

1. 打开 WorkSpaces 应用程序控制台。
2. 在左侧导航窗格中，选择 用户池。
3. 选择创建用户。
4. 提供一个电子邮件地址，用户可以在其中收到使用 WorkSpaces 应用程序的电子邮件邀请、名字和姓氏，然后选择创建用户。
5. 如有必要，请重复此步骤以创建更多用户。每个用户的电子邮件地址必须是唯一的。

有关创建 WorkSpaces 应用程序用户的更多信息，请参阅《Amazon WorkSpaces 应用程序管理指南》中的 WorkSpaces 应用程序用户池。

创建 CloudFormation 完堆栈后，您可以将创建的用户分配给堆栈，如下所示：

1. 打开 WorkSpaces 应用程序控制台。
2. 选择用户名。
3. 依次选择操作和分配堆栈。
4. 在分配堆栈中，选择以 `m2-appstream-stack-ea` 开头的堆栈。
5. 选择分配堆栈。



将用户分配到堆栈会使 WorkSpaces 应用程序通过您提供的地址向该用户发送电子邮件。此电子邮件包含指向 WorkSpaces 应用程序登录页面的链接。

步骤 4：登录 WorkSpaces 应用程序

在此步骤中，您将使用 WorkSpaces 应用程序发送给您在中创建的用户的电子邮件中的链接登录 WorkSpaces 应用程序步骤 3：在 “ WorkSpaces 应用程序” 中创建用户。

1. 使用 WorkSpaces 应用程序发送的电子邮件中提供的链接登录 WorkSpaces 应用程序。
2. 如果出现提示，请更改密码。您看到的 “ WorkSpaces 应用程序” 屏幕类似于以下内容：

3. 选择桌面。
4. 在任务栏上，选择搜索，然后输入 **D:** 以导航到主文件夹。

> **Note**
>
> 如果跳过此步骤，则在尝试访问主文件夹时可能会出现`Device not ready`错误。

在任何时候，如果您在登录 WorkSpaces 应用程序时遇到问题，都可以按照以下步骤重新启动 WorkSpaces 应用程序群并尝试重新登录。

1. 打开 WorkSpaces 应用程序控制台。
2. 在左侧导航窗格中，选择实例集。
3. 选择您要使用的实例集。
4. 选择操作，然后选择停止。
5. 等待实例集停止。
6. 选择操作，然后选择启动。

此过程大约需要 10 分钟。

步骤 5：验证 Amazon S3 中的存储桶（可选）

您用于创建堆栈的 CloudFormation 模板完成的任务之一是在 Amazon S3 中创建两个存储桶，这两个存储桶是跨工作会话保存和恢复用户数据和应用程序设置所必需的。这些存储桶如下所示：

- 名称以 `appstream2-` 开头。此存储桶将数据映射到 WorkSpaces 应用程序中的主文件夹 (`D:\PhotonUser\My Files\Home Folder`)。

> **Note**
>
> 对于给定的电子邮件地址，主文件夹是唯一的，并且在给定帐户中的所有舰队和堆栈之间共享。 AWS 主文件夹的名称是用户电子邮件地址的 SHA256 哈希值，存储在基于该哈希值的路径上。

- 名称以 `appstream-app-settings-` 开头。此存储桶包含 WorkSpaces 应用程序的用户会话信息，并包括浏览器收藏夹、IDE 和应用程序连接配置文件以及用户界面自定义等设置。有关更多信息，请参阅《Amazon 应用程序管理指南》中的 WorkSpaces 应用程序设置持久性工作原理。

要验证存储桶是否已创建，请执行以下步骤：

1. 打开 Amazon S3 控制台。
2. 在左侧导航中，选择存储桶。
3. 在按名称查找存储桶中，输入 **`appstream`** 以筛选列表。

如果您看到存储桶，则无需采取进一步的操作。请注意存储桶存在即可。如果您没有看到存储桶，则可能是 CloudFormation 模板未完成运行，或者发生了错误。转到 CloudFormation 控制台并查看堆栈创建消息。

后续步骤

现在， WorkSpaces 应用程序基础架构已设置完毕，您可以设置并开始使用 Enterprise Analyzer。有关更多信息，请参阅 教程：在 WorkSpaces 应用程序上设置企业分析器。您还可以设置 Enterprise Developer。有关更多信息，请参阅 教程：在 WorkSpaces 应用程序上设置 Rocket 企业开发人员。

清理资源

创建 WorkSpaces 应用程序队列和堆栈中描述了清理已创建堆栈和队列的过程。

删除 WorkSpaces 应用程序对象后，账户管理员还可以根据需要清理 Amazon S3 存储桶中的应用程序设置和主文件夹。

> **Note**
> 给定用户的主文件夹在所有队列中都是唯一的，因此，如果同一账户中的其他 WorkSpaces 应用程序堆栈处于活动状态，则可能需要保留该文件夹。

最后， WorkSpaces 应用程序目前不允许您使用控制台删除用户。而是必须使用服务 API 和 CLI 进行删除。有关更多信息，请参阅《Amazon WorkSpaces 应用程序管理指南》中的用户池管理。

教程：在 Rocket Enterprise Developer（前身为 Micro

本教程介绍如何在 Rocket Enterprise 开发者中使用模板和预定义项目。它提供了三个用例。所有用例都使用示例中提供的 BankDemo 示例代码。要下载该示例，请选择 `bankdemo.zip`。

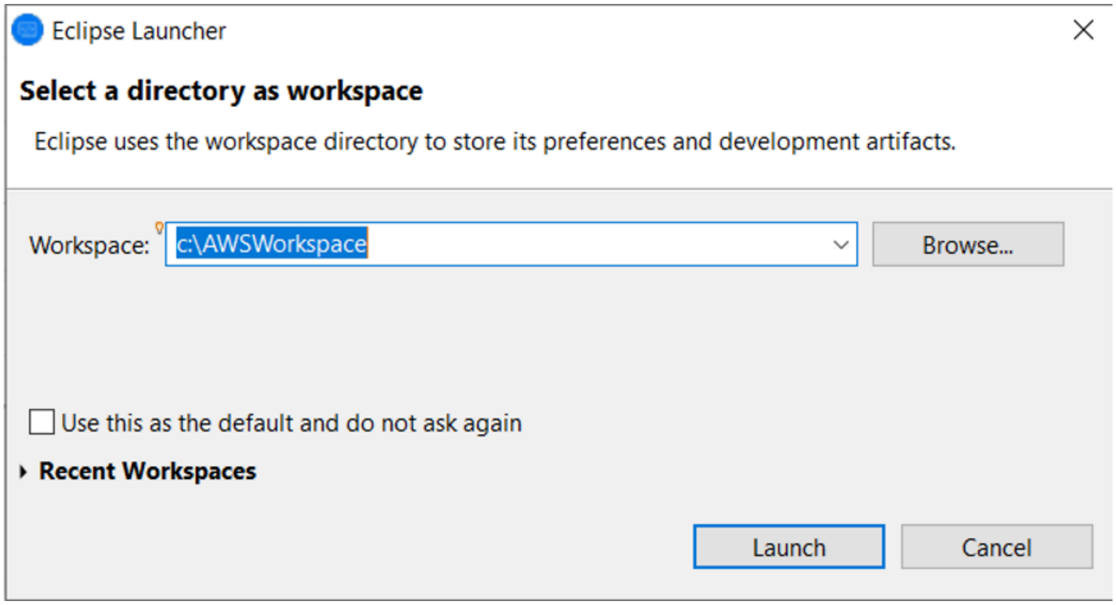
> **Important**
> 如果您使用适用于 Windows 的 Enterprise Developer 版本，则编译器生成的二进制文件只能在与 Enterprise Developer 一并提供的 Enterprise Server 上运行。您无法在基于 Linux AWS 的大型机现代化运行时下运行它们。

主题

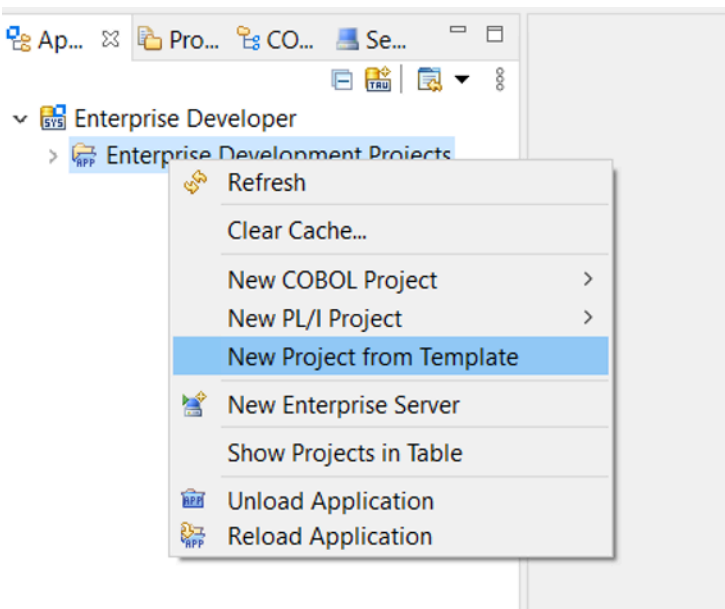
用例 1 – 使用包含源组件的 COBOL 项目模板

此用例要求您在演示预设置步骤中，将源组件复制到模板目录结构中。为了避免源有两个副本，在 `bankdemo.zip` 中，这点已从提供的原始 `AWSTemplates.zip` 中进行了更改。

1. 启动 Enterprise Developer 并指定所选工作区。

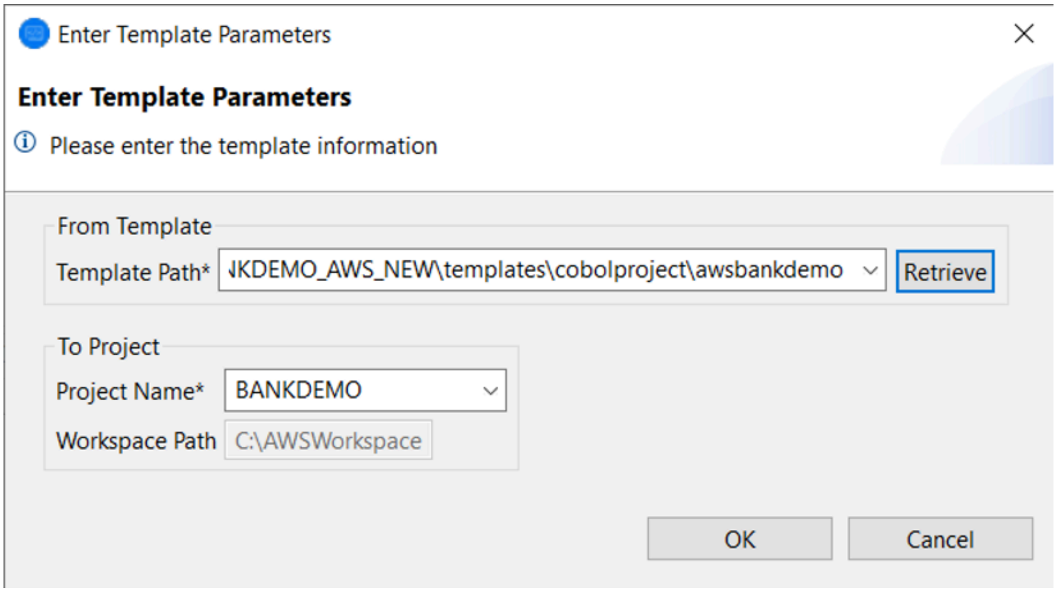


2. 在应用程序资源管理器视图中，从企业开发项目树视图项的上下文菜单中选择从模板新建项目。

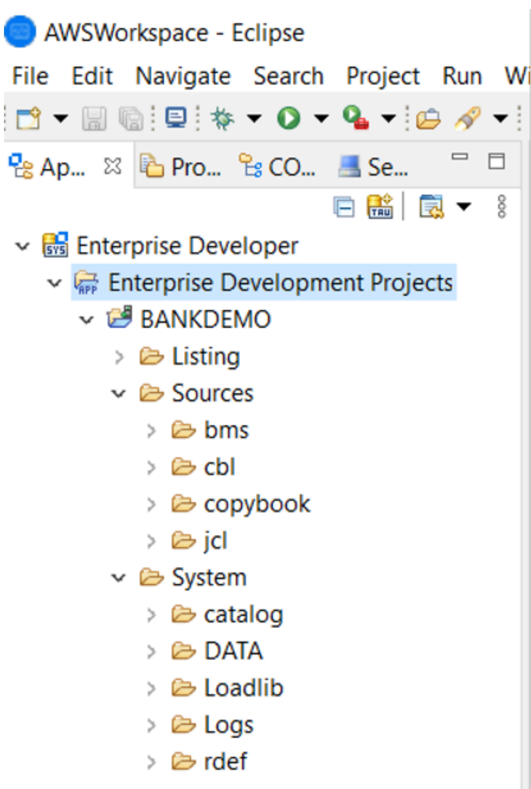


3. 输入模板参数，如图所示。

> **Note**
>
> “模板路径”是指 ZIP 的解压位置。



4. 选择“确定”将基于提供的模板创建一个本地开发 Eclipse 项目，该项目具有完整的源和执行环境结构。



`System` 结构包含一个完整的资源定义文件，其中包含 BANKDEMO 所需的条目、添加了条目的必需目录以及相应的 ASCII 数据文件。

由于源模板结构包含所有源项目，这些文件将复制到本地项目，因此会在 Enterprise Developer 中自动构建。

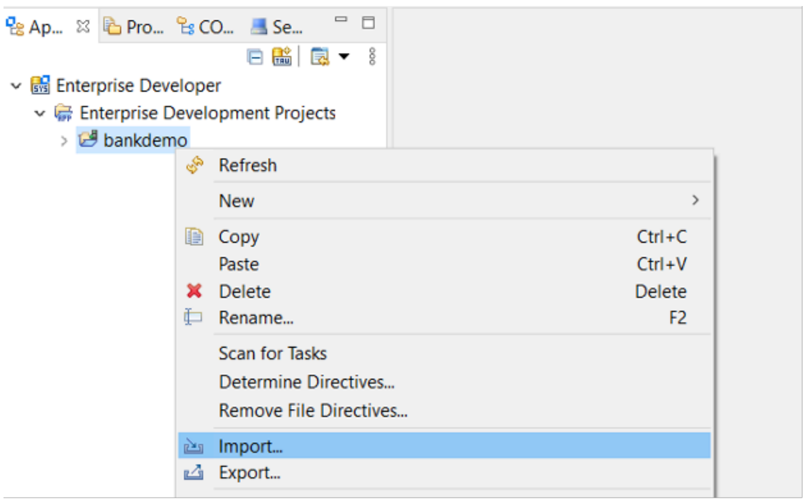
用例 2 – 使用不包含源组件的 COBOL 项目模板

步骤 1 到 3 与用例 1 – 使用包含源组件的 COBOL 项目模板相同。

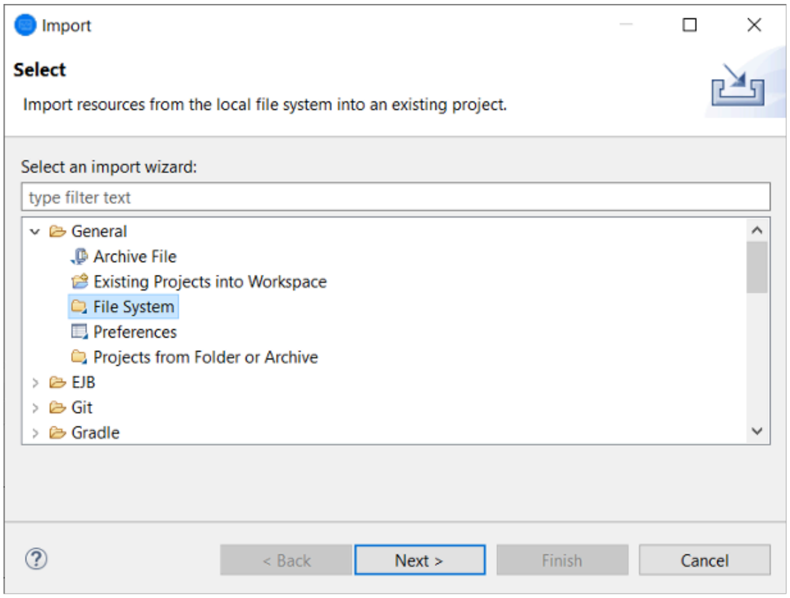
此用例中的System结构还包含一个完整的资源定义文件，其中包含所需的条目 BankDemo、添加了条目的必需目录以及相应的 ASCII 数据文件。

不过，模板源结构不包含任何组件。您必须将组件从您正在使用的任何源存储库导入到项目中。

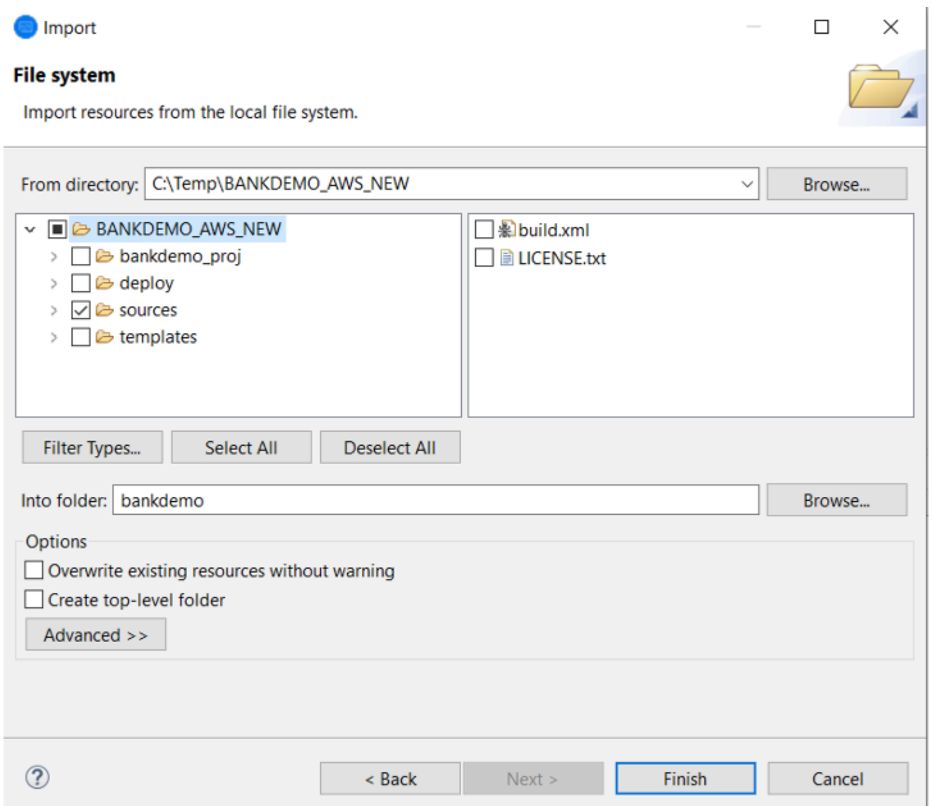
1. 选择项目名称。从相关的上下文菜单中，选择导入。



2. 在显示的对话框中，在 General 部分下，选择 File System，然后选择“下一步”。



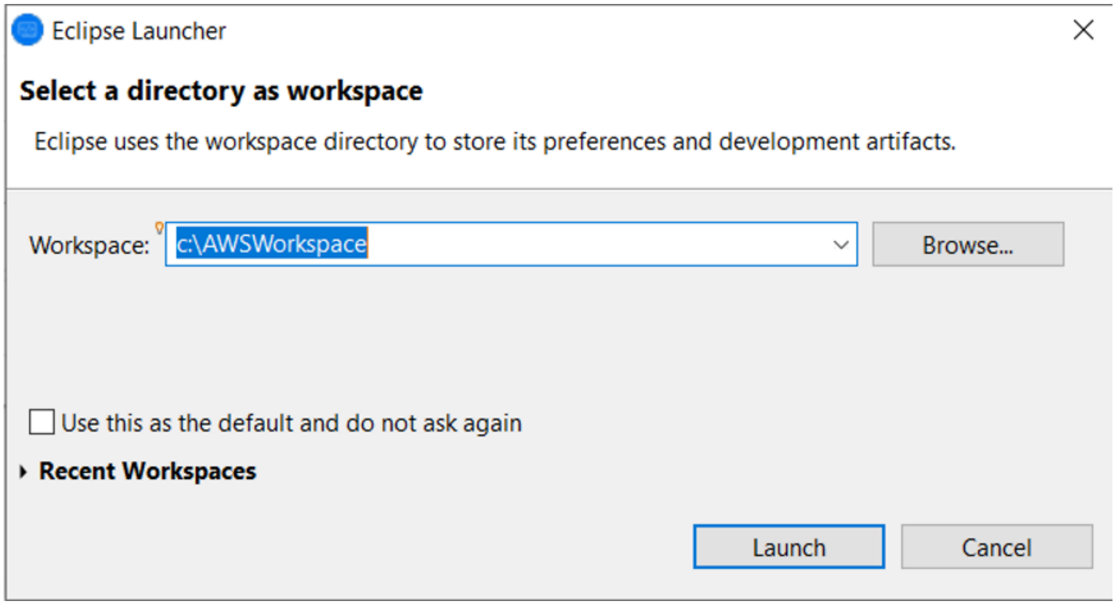
3. 通过浏览文件系统以指向存储库文件夹来填充从目录字段。选择要导入的所有文件夹，例如 `sources`。`Into folder` 字段将预先填充。选择完成。



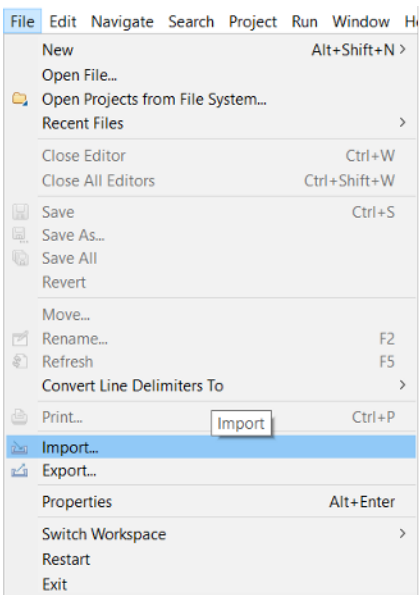
源模板结构包含所有源项目之后，即可在 Enterprise Developer 中自动构建。

用例 3 – 使用链接到源文件夹的预定义 COBOL 项目

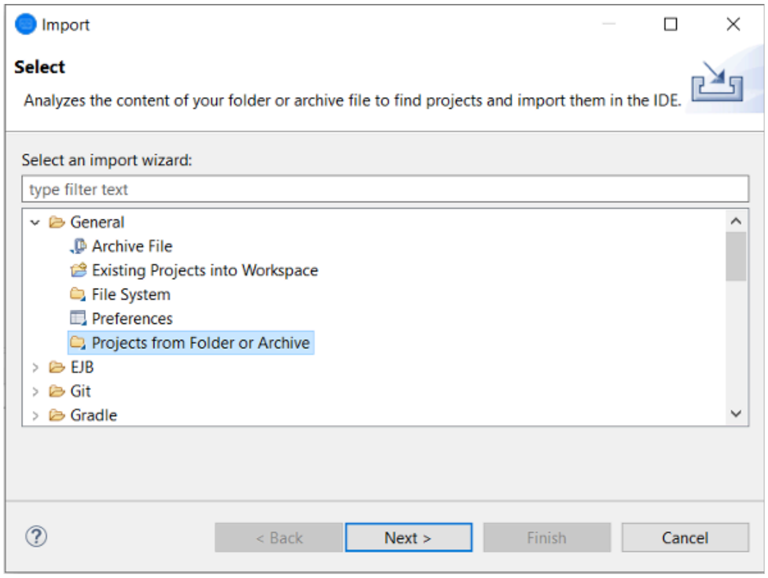
1. 启动 Enterprise Developer 并指定所选工作区。



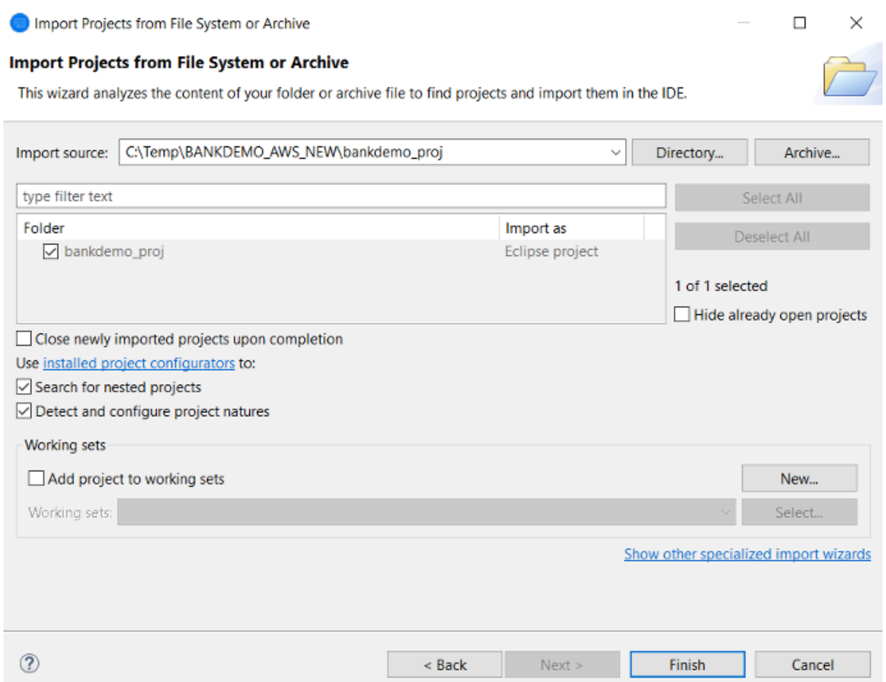
2. 从文件菜单中，选择导入。



3. 在显示的对话框中，在 General下，选择 Projects from Folder or Archive，然后选择下一步。



4. 填充导入源，选择目录，然后浏览文件系统以选择预定义的项目文件夹。其中包含的项目具有指向同一存储库中源文件夹的链接。

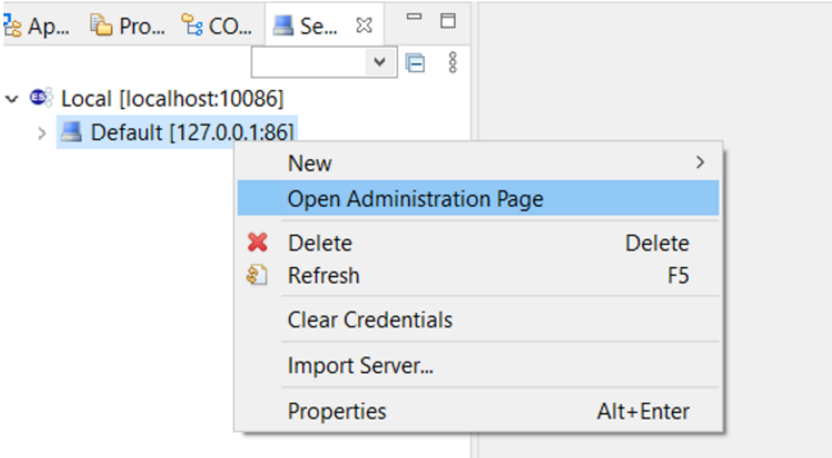


选择完成。

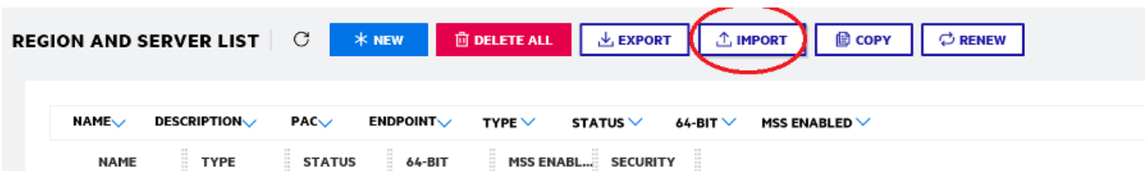
由于项目由源文件夹的链接填充，因此代码是自动生成的。

使用区域定义 JSON 模板

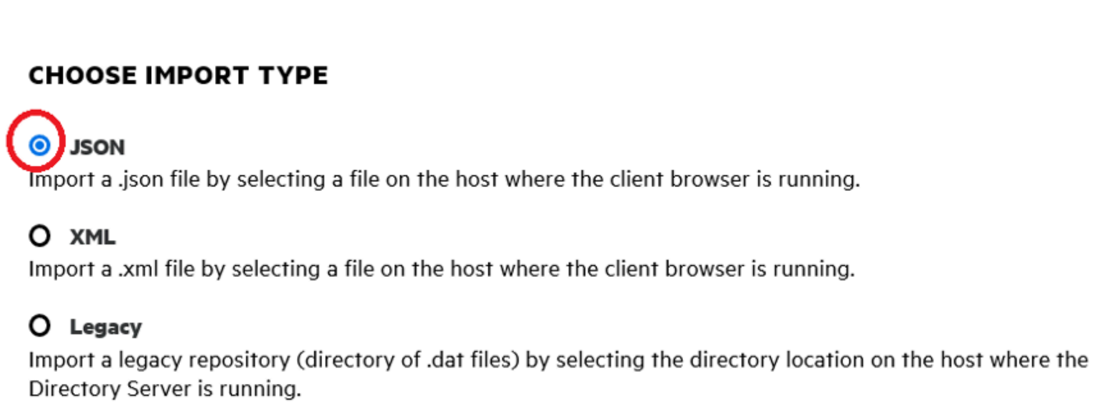
1. 切换到服务器资源管理器视图。从相关的上下文菜单中选择打开管理页面，这将启动默认浏览器。



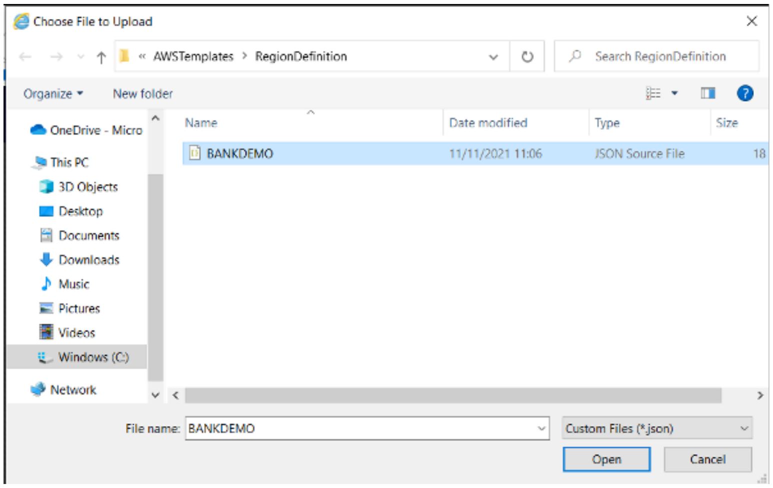
2. 从显示的 Enterprise Server 常用 Web 管理 (ESCWA) 屏幕中，选择导入。



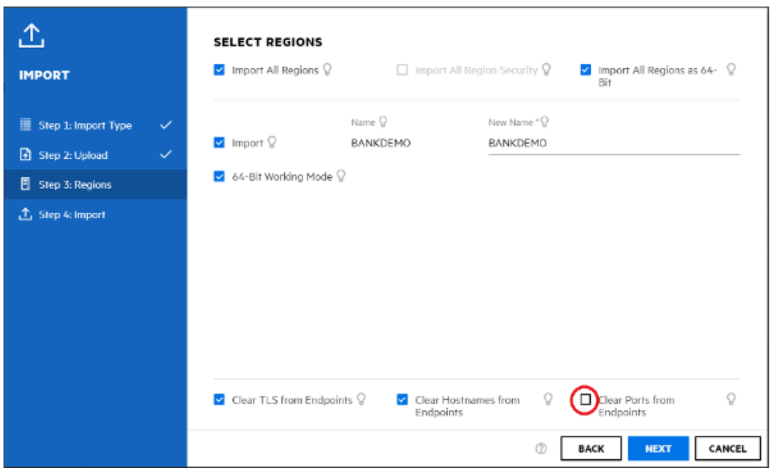
3. 选择 JSON 导入类型，然后选择下一步。



4. 上传提供的 `BANKDEMO.JSON` 文件。

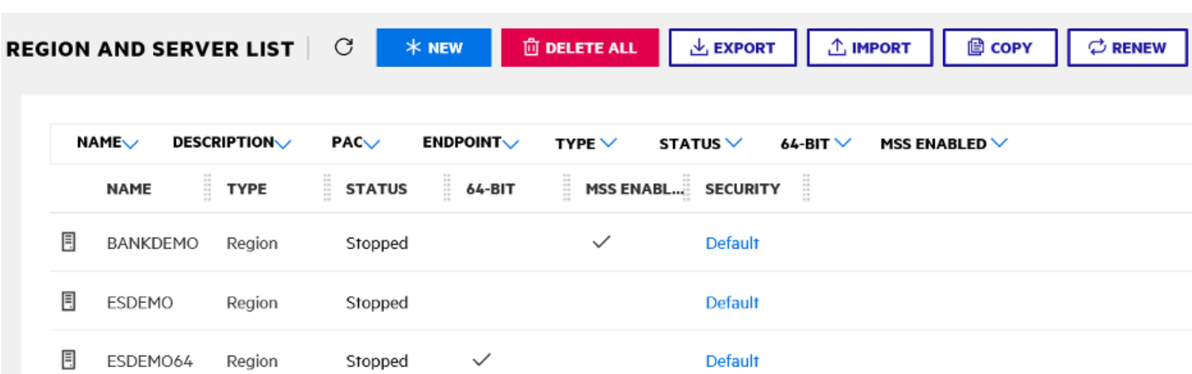


选择后，选择下一步。



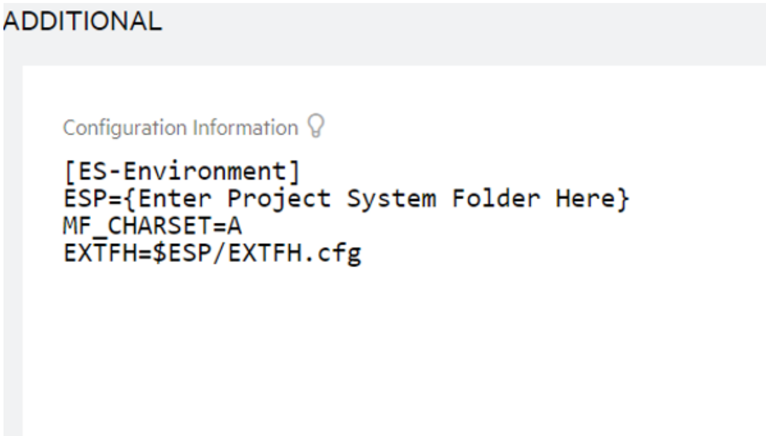
在选择区域面板上，确保未选中从端点清除端口选项，然后继续在面板中选择下一步，直到显示执行导入面板。然后，从左侧导航窗格中选择导入。

最后，单击完成。BANKDEMO 区域随后将被添加到服务器列表中。



5. 导航至 BANKDEMO 区域的常规属性。

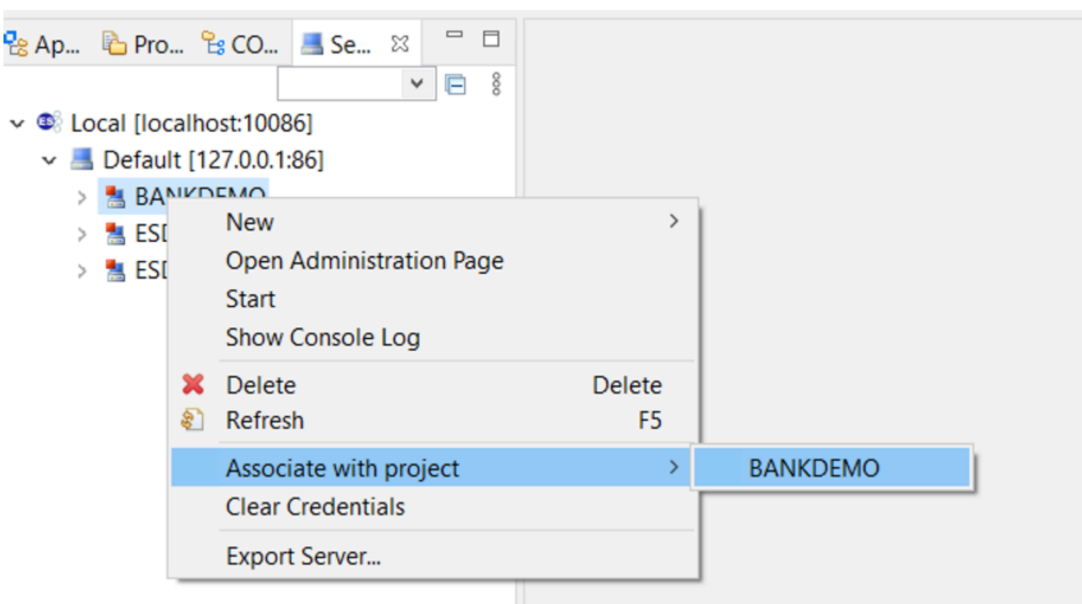
6. 滚动到配置部分。
7. 需将 ESP 环境变量设置为与前面步骤中创建的 Eclipse 项目相关的 `System` 文件夹。应为 `workspacefolder/projectname/System`。



8. 单击应用。

   该区域现已完全配置为与 Eclipse COBOL 项目配合运行。

9. 最后，回到 Enterprise Developer 中，将导入的区域与项目关联。



企业开发人员环境现已准备就绪，其完整工作版本为 BankDemo。您可以针对该区域编辑、编译和调试代码。

> **Important**
>
> 如果您使用适用于 Windows 的 Enterprise Developer 版本，则编译器生成的二进制文件只能在与 Enterprise Developer 一并提供的 Enterprise Server 上运行。您无法在基于 Linux AWS 的大型机现代化运行时下运行它们。

教程：在 WorkSpaces 应用程序上设置企业分析器

本教程介绍如何设置 Rocket Enterprise Analyzer（前身为 Micro Focus Enterprise Analyzer）来分析一个或多个 Enterprise Analyzer 工具可根据其对应用程序源代码和系统定义的分析提供多个报告。

此设置旨在促进团队协作。安装使用 Amazon S3 存储桶与虚拟磁盘共享源代码。这样做会在 Windows 计算机上使用 Rclone。借助运行 PostgreSQL 的常用 Amazon RDS 实例，团队中的任何成员都可以访问所有请求的报告。

团队成员还可以将 Amazon S3 支持的虚拟磁盘挂载到个人机器上，并从其工作站更新源存储桶。如果他们连接到其他本地内部系统，还可以在其机器上使用脚本或任何其他形式的自动化。

该设置基于 AWS 大型机现代化与客户共享的 WorkSpaces 应用程序 Windows 映像。安装还基于 WorkSpaces 应用程序队列和堆栈的创建，如中所述。教程：设置 WorkSpaces 应用程序以用于 Rocket Enterprise Analyzer 和 Rocket

> **Important**
>
> 本教程中的步骤假设您使用可下载的 CloudFormation 模板 cfn-m2-.yml 来设置 WorkSpaces 应用程序。appstream-fleet-ea-ed有关更多信息，请参阅 教程：设置 WorkSpaces 应用程序以用于 Rocket Enterprise Analyzer 和 Rocket。
> 要执行本教程中的步骤，您必须设置 Enterprise Analyzer 实例集和堆栈，并且它们必须正在运行。

有关企业分析器功能和交付内容的完整描述，请参阅 Rocket Software（前身为 Micro Focus）网站上的企业分析器文档。

映像内容

除了 Enterprise Analyzer 应用程序本身之外，该映像还包含以下工具和库。

第三方工具

- Python
- Rclone
- pgAdmin
- git-scm
- PostgreSQL ODBC 驱动程序

`C:\Users\Public` 中的库

- BankDemo 企业开发者的源代码和项目定义:`m2-bankdemo-template.zip`.
- 大型机的 MFA 安装包：`mfa.zip`。有关更多信息，请参阅《Micro Focus Enterprise Developer》文档中的大型机访问权限概述。
- Rclone 的命令和配置文件（教程中的使用说明）：`m2-rclone.cmd` 和 `m2-rclone.conf`。

主题

先决条件

- 将您要分析的客户应用程序的源代码和系统定义上传到 S3 存储桶。系统定义包括 CICS CSD、DB2 对象定义等。您可以在存储桶中创建一个文件夹结构，该结构表示您要如何组织应用程序构件。例如，解压缩 BankDemo 示例时，其结构如下：

```
demo
     |--> jcl
     |--> RDEF
     |--> transaction
```

```
    |--> xa
```

- 创建并启动运行 PostgreSQL 的 Amazon RDS 实例 此实例将存储 Enterprise Analyzer 生成的数据和结果。您可以与应用程序团队的所有成员共享此实例。此外，在数据库中创建一个名为 `m2_ea`（或任何其他合适的名称）的空架构。为授权用户定义凭证，允许其创建、插入、更新和删除此架构中的项目。您可以从 Amazon RDS 控制台或从账户管理员处获取数据库名称、服务器端点 URL 和 TCP 端口。
- 确保您已设置了对您的的编程访问权限 AWS 账户。有关更多信息，请参阅《Amazon Web Services 一般参考》中的编程访问。

步骤 1：设置

1. 使用您在 WorkSpaces 应用程序的欢迎电子邮件中收到的 URL 开始与 WorkSpaces 应用程序的会话。
2. 使用您的电子邮件作为用户 ID，并定义您的永久密码。
3. 选择 Enterprise Analyzer 堆栈。
4. 在 “ WorkSpaces 应用程序” 菜单页面上，选择 “桌面” 以访问队列正在进行流式传输的 Windows 桌面。

步骤 2：在 Windows 上创建基于 Amazon S3 的虚拟文件夹

> **Note**
>
> 如果您在 AWS 大型机现代化预览版中已经使用过 Rclone，则必须更新`m2-rclone.cmd`到位于中的较新版本。`C:\Users\Public`

1. 使用文件资源管理器将 `C:\Users\Public` 中提供的 `m2-rclone.conf` 和 `m2-rclone.cmd` 文件复制到您的主文件夹 `C:\Users\PhotonUser\My Files\Home Folder`。
2. 使用您的 AWS 访问密钥和相应的密钥以及您的密钥更新`m2-rclone.conf`配置参数 AWS 区域。

```
[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. 在 `m2-rclone.cmd` 中，进行以下更改：

   - 将 `amzn-s3-demo-bucket` 更改为您的 Amazon S3 存储桶名称。例如 `m2-s3-mybucket`。
   - 将 `your-s3-folder-key` 更改为您的 Amazon S3 存储桶密钥。例如 `myProject`。
   - 将 `your-local-folder-path` 更改为要从包含应用程序文件的 Amazon S3 存储桶同步应用程序文件的目录路径。例如 `D:\PhotonUser\My Files\Home Folder\m2-new`。此同步目录必须是主文件夹的子目录， WorkSpaces 应用程序才能在会话开始和结束时对其进行正确备份和恢复。

```
:loop
timeout /T 10
"C:\Program Files\rclone\rclone.exe" sync m2-s3:amzn-s3-demo-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. 打开 Windows 命令提示符，使用 cd 命令切换到 `C:\Users\PhotonUser\My Files\Home Folder`（如果需要），然后运行 `m2-rclone.cmd`。此命令脚本连续循环运行，每 10 秒将您的 Amazon S3 存储桶和密钥同步到本地文件夹。您可以根据需要调整超时。您应该在 Windows 文件资源管理器的 Amazon S3 存储桶中看到该应用程序的源代码。

要将新文件添加到您正在处理的文件集中或更新现有文件，请将文件上传到 Amazon S3 存储桶，这些文件将在 `m2-rclone.cmd` 中定义的下一次迭代中同步到您的目录。同样，如果要删除某些文件，请将其从 Amazon S3 存储桶中删除。下次同步操作会将其从您的本地目录中删除。

步骤 3：为 Amazon RDS 实例创建 ODBC 源

1. 要启动 EA_Admin 工具，请导航到浏览器窗口左上角的应用程序选择器菜单，然后选择 MF EA_Admin。
2. 从管理菜单中，选择 ODBC 数据来源，然后从用户 DSN 选项卡中选择添加。
3. 在“创建新数据来源”对话框中，选择 PostgreSQL Unicode 驱动程序，然后选择完成。
4. 在 PostgreSQL Unicode ODBC 驱动程序 (psqlODBC) 设置对话框中，定义并记下所需的数据来源名称。使用您之前创建的 RDS 实例中的值填写以下参数：

描述

可选描述，可帮助您快速识别此数据库连接。

数据库

您之前创建的 Amazon RDS 数据库。

服务器

Amazon RDS 端点。

端口

Amazon RDS 端口。

用户名称

Amazon RDS 实例中所定义的用户名称。

密码

Amazon RDS 实例中所定义的密码。

5. 选择测试以验证与 Amazon RDS 的连接是否成功，然后选择保存以保存您的新用户 DSN。
6. 等待直到看到确认创建了正确工作区的消息，然后选择确定以完成 ODBC 数据来源操作并关闭 EA_Admin 工具。
7. 再次导航到应用程序选择器菜单，然后选择“Enterprise Analyzer”以启动该工具。选择新建。
8. 在工作区配置窗口中，输入您的工作区名称并定义其位置。如果您在此配置下工作，该工作区可以是基于 Amazon S3 的磁盘，也可以是您的主文件夹（如果需要）。
9. 选择选择其他数据库以连接到您的 Amazon RDS 实例。
10. 从选项中选择 Postgre 图标，然后选择确定。
11. 在选项 – 定义连接参数下的 Windows 设置中，输入您创建的数据来源的名称。此外，输入数据库名称、架构名称、用户名和密码。选择确定。
12. 等待 Enterprise Analyzer 创建用于存储结果所需的所有表、索引等。此过程可能需要几分钟时间。Enterprise Analyzer 会确认数据库和工作区何时准备就绪可供使用。
13. 再次导航到应用程序选择器菜单，然后选择“Enterprise Analyzer”以启动该工具。
14. Enterprise Analyzer 启动窗口出现在新的所选工作区位置中。选择确定。
15. 在左侧窗格中，导航到存储库，选择存储库名称，然后选择向工作区添加文件/文件夹。选择存储应用程序代码的文件夹，将其添加到工作区。如果你愿意，你可以使用前面的 BankDemo示例代

码。Enterprise Analyzer 提示您验证这些文件时，请选择验证以启动初始 Enterprise Analyzer 验证报告。该过程可能需要几分钟才能完成，具体取决于您的应用程序的大小。

16. 展开工作区，以便查看已添加到工作区的文件和文件夹。对象类型和圈复杂度报告也显示在图表查看器窗格的上象限中。

现在，您可以使用 Enterprise Analyzer 来完成所有需要的任务。

后续会话

1. 使用您在 WorkSpaces 应用程序的欢迎电子邮件中收到的 URL 开始与 WorkSpaces 应用程序的会话。
2. 使用您的电子邮件和永久密码登录。
3. 选择 Enterprise Analyzer 堆栈。
4. 启动 `Rclone` 以连接到 Amazon S3 支持的磁盘（如果您使用此选项共享工作区文件）。
5. 启动 Enterprise Analyzer 来完成任务。

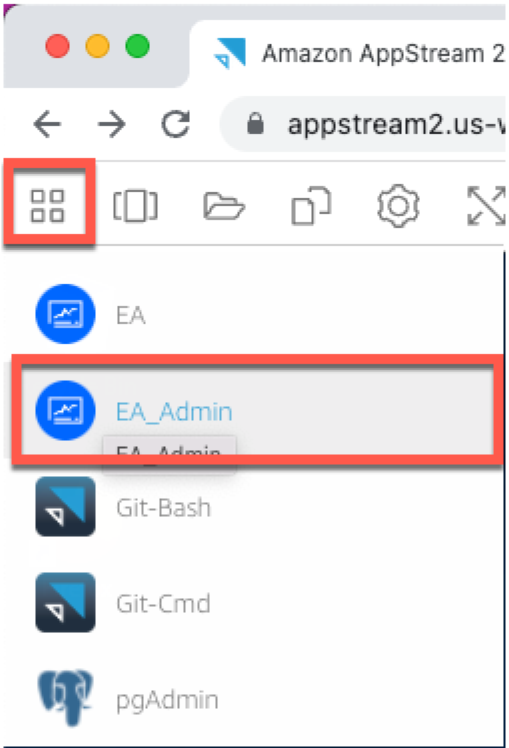
对工作区连接进行故障排除

当您尝试重新连接到 Enterprise Analyzer 工作区时，可能会看到类似如下错误：

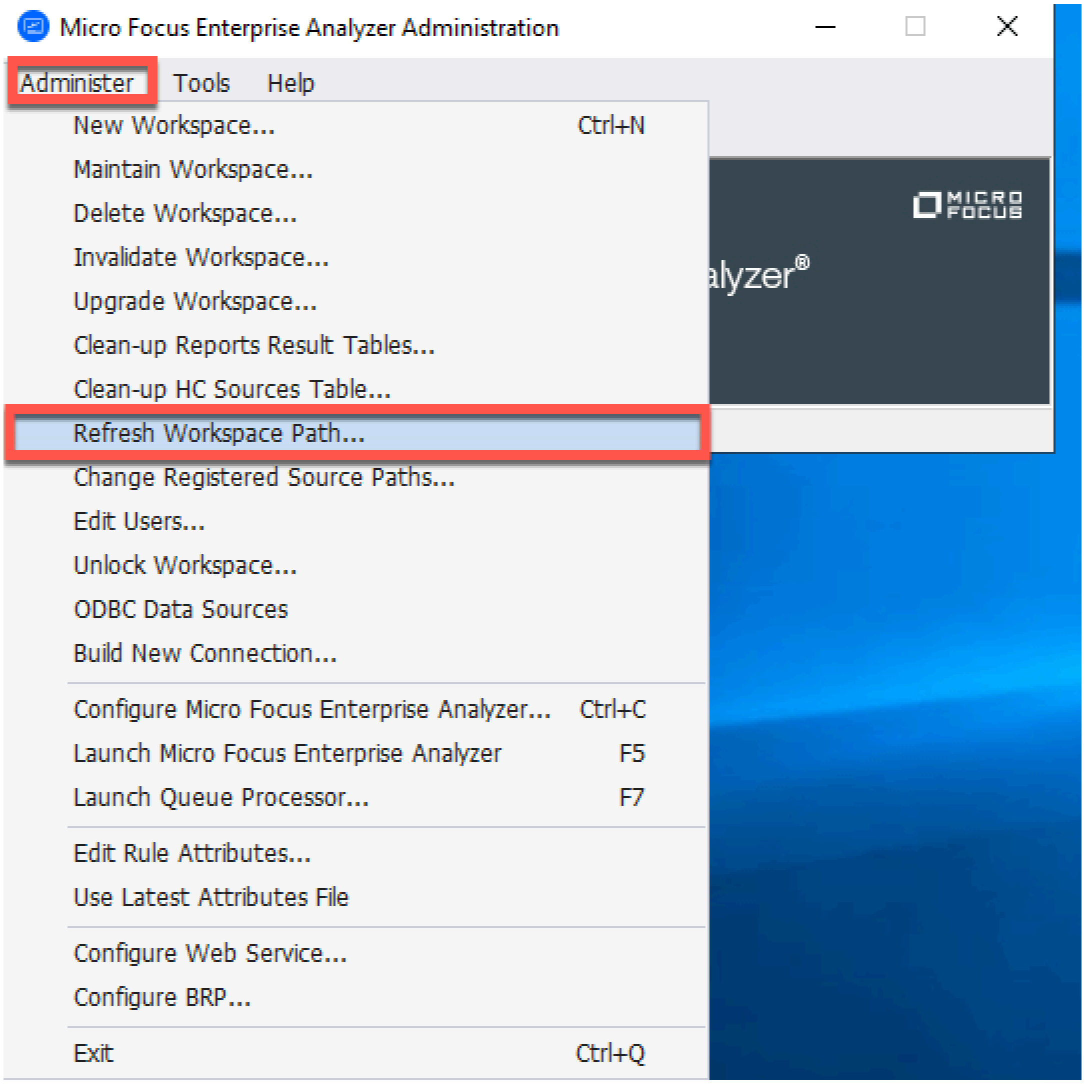
```
Cannot access the workspace directory D:\PhotonUser\My Files\Home Folder\EA_BankDemo.
 The workspace has been created on a non-shared disk of the EC2AMAZ-E6LC33H computer.
 Would you like to correct the workspace directory location?
```

要解决此问题，请选择确定来清除消息，然后完成以下步骤。

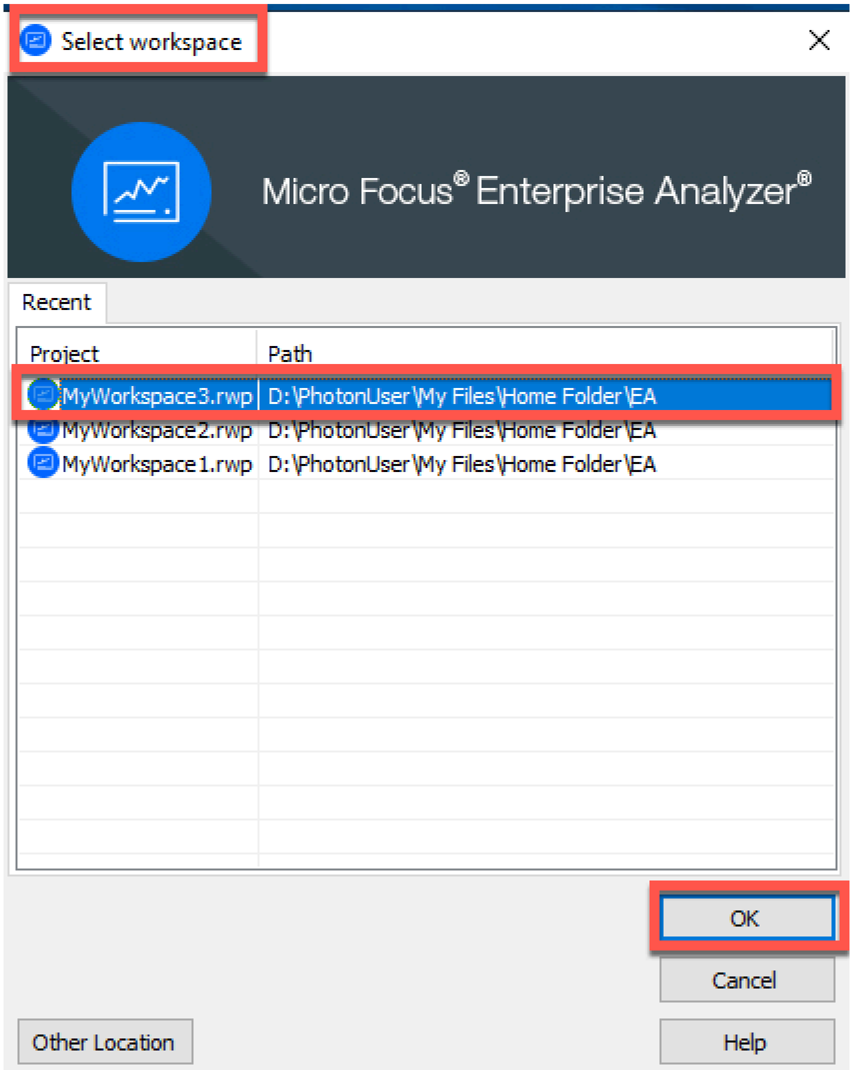
1. 在 “ WorkSpaces 应用程序” 中，选择工具栏上的 “启动应用程序” 图标，然后选择 ea_admin 以启动 En terprise Analyzer 管理工具。



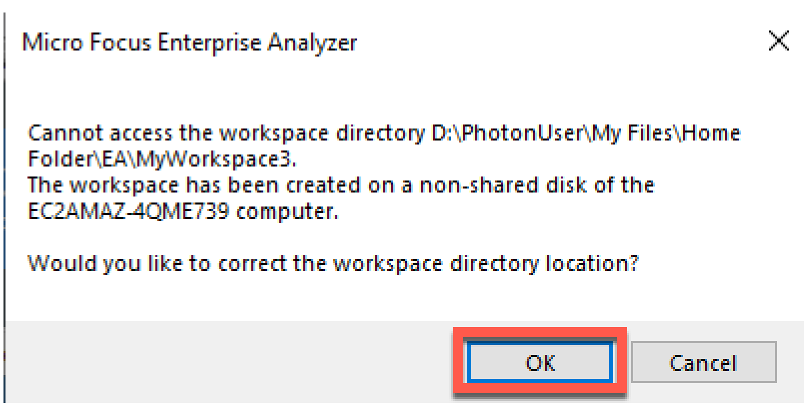
2. 从管理菜单中，选择刷新工作区路径...。



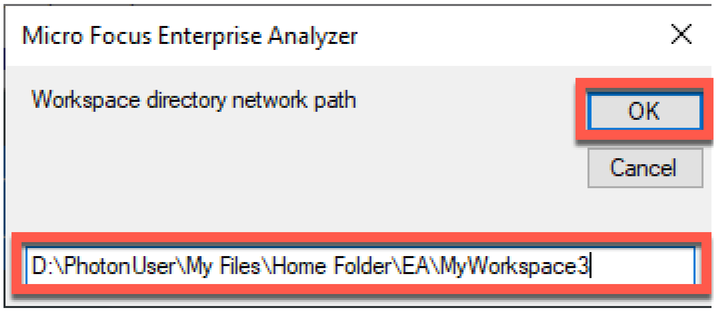
3. 在选择工作区下，选择所需的工作区，然后选择确定。



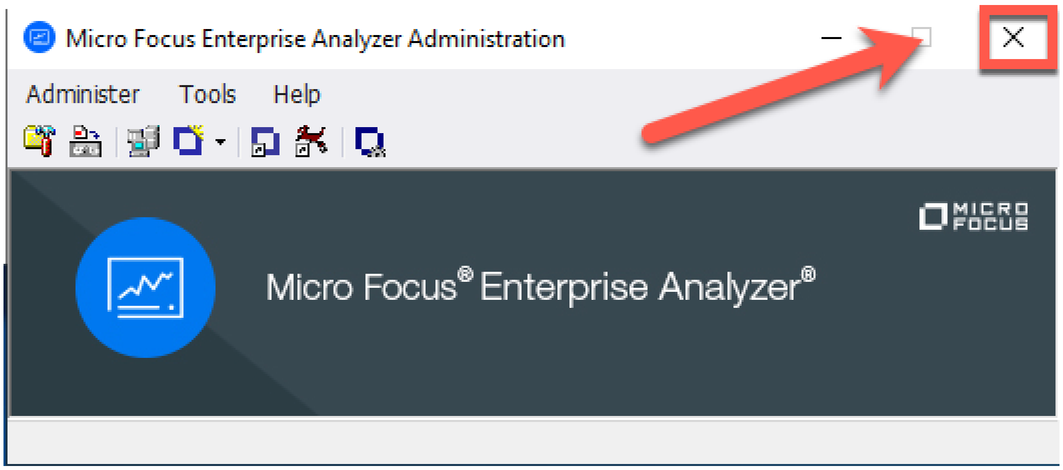
4. 选择确定，确认错误消息。



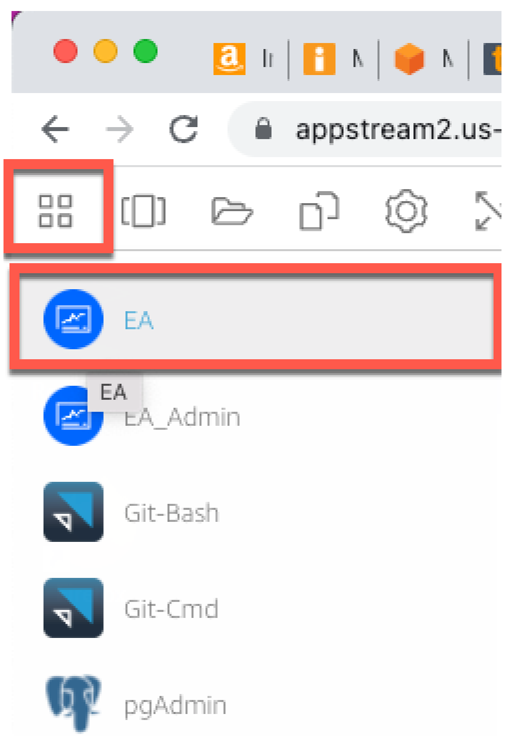
5. 在工作区目录网络路径下，输入工作区的正确路径，例如 `D:\PhotonUser\My Files\Home Folder\EA\MyWorkspace3`。



6. 关闭 Micro Focus Enterprise Analyzer Administration 工具。



7. 在 “ WorkSpaces 应用程序” 中，选择工具栏上的 “启动应用程序” 图标，然后选择 EA 启动 Micro Focus E nterprise Analyzer。



8. 重复步骤 3 – 5。

Micro Focus Enterprise Analyzer 此时应可使用现有工作区打开。

清理资源

如果您不再需要为本教程创建的资源，请将其删除，以免继续产生费用。完成以下步骤：

- 使用 EA_Admin 工具删除工作区。
- 删除您为本教程创建的 S3 存储桶。有关更多信息，请参阅《Amazon S3 用户指南》中的删除存储桶。
- 选择您为本教程创建的数据库。有关更多信息，请参阅删除数据库实例。

教程：在 WorkSpaces 应用程序上设置 Rocket 企业开发人员

本教程介绍如何为一个或多个大型机应用程序设置 Rocket Enterprise Developer（前身为 Micro Focus Enterprise Developer），以便使用企业开发者功能对其进行维护、编译和测试。该设置基于 AWS 大型机现代化与客户共享的 WorkSpaces 应用程序 Windows 映像，以及 WorkSpaces 应用程序队列和堆栈的创建，如中所述。教程：设置 WorkSpaces 应用程序以用于 Rocket Enterprise Analyzer 和 Rocket

> **Important**
>
> 本教程中的步骤假设您使用可下载的 CloudFormation 模板 cfn-m2- .yaml 来设置 WorkSpaces 应用程序。appstream-fleet-ea-ed有关更多信息，请参阅 教程：设置 WorkSpaces 应用程序以用于 Rocket Enterprise Analyzer 和 Rocket。
>
> 如果 Enterprise Developer 实例集和堆栈已启动并正在运行时，则必须执行此设置的步骤。

有关企业开发者 v7 功能和交付成果的完整描述，请在 Rocket Softwar e（前身为 Micro Focus）网站上查看其up-to-date 在线文档 (v7.0)。

映像内容

除了企业开发者本身，该图像还包含包含 Rumba（ TN3270 模拟器）的图像。此外，它还包含以下工具和库。

第三方工具

- Python
- Rclone
- pgAdmin
- git-scm
- PostgreSQL ODBC 驱动程序

`C:\Users\Public` 中的库

- BankDemo 企业开发者的源代码和项目定义:`m2-bankdemo-template.zip`.
- 大型机的 MFA 安装包：`mfa.zip`。有关更多信息，请参阅《Micro Focus Enterprise Developer》文档中的大型机访问权限概述。
- Rclone 的命令和配置文件（教程中的使用说明）：`m2-rclone.cmd` 和 `m2-rclone.conf`。

如果您需要访问尚未加载到 CodeCommit 存储库但在 Amazon S3 存储桶中可用的源代码，例如将源代码初始加载到 git 中，请按照中所述的步骤创建虚拟 Windows 磁盘教程：在 WorkSpaces 应用程序上设置企业分析器。

主题

- 先决条件
- 步骤 1：由个人 Enterprise Developer 用户设置
- 步骤 2：在 Windows 上创建基于 Amazon S3 的虚拟文件夹（可选）
- 步骤 3：克隆存储库
- 后续会话
- 清理资源

先决条件

- 一个或多个装有待维护应用程序源代码的 CodeCommit 存储库。存储库的设置应符合上述CI/CD管道的要求，以便通过结合使用这两种工具来实现协同效应。
- 每个用户都必须拥有账户管理员根据 AWS 的身份验证和访问控制中的信息定义的一个或多个存储库的证书 CodeCommit。 CodeCommit 这些证书的结构在 AWS CodeCommit 的身份验证和访问控制中进行了审查，IAM 授权的完整参考见CodeCommit 权限参考：管理员可以为不同的角色定义不同的 IAM 策略，这些角色具有每个存储库的特定角色的证书，并限制其对用户的授权仅限于他在给定存储库上必须完成的特定任务集。 CodeCommit 因此，账户管理员将为 CodeCommit 仓库的每位

维护者生成一个主用户，并通过选择适当的 IAM 访问策略向该用户授予 CodeCommit访问所需一个或多个存储库的权限。

步骤 1：由个人 Enterprise Developer 用户设置

1. 获取您的 IAM 凭证：

    1. 连接 AWS 控制台，网址为https://console.aws.amazon.com/iam/。
    2. 按照《AWS CodeCommit 用户指南》中适用于使用 Git 凭证的 HTTPS 用户的设置的步骤 3 中描述的过程操作。
    3. 复制 IAM 为您生成的 CodeCommit特定登录证书，方法是显示、复制这些信息，然后将其粘贴到本地计算机上的安全文件中，或者选择下载证书将此信息下载为.CSV 文件。您需要该信息才能连接到 CodeCommit。

2. 根据欢迎电子邮件中收到的网址开始与 WorkSpaces 应用程序的会话。使用您的电子邮件作为用户名并创建密码。
3. 选择您的 Enterprise Developer 堆栈。
4. 在菜单页面上，选择桌面以访问实例集正在进行流式传输的 Windows 桌面。

步骤 2：在 Windows 上创建基于 Amazon S3 的虚拟文件夹（可选）

如果需要 Rclone（见上文），请在 Windows 上创建基于 Amazon S3 的虚拟文件夹：（如果所有应用程序工件都完全来自访问权限，则可选）。 CodeCommit

> **Note**
>
> 如果您在 AWS 大型机现代化预览版中已经使用过 Rclone，则必须更新`m2-rclone.cmd`到位于中的较新版本。`C:\Users\Public`

1. 使用文件资源管理器将 `C:\Users\Public` 中提供的 `m2-rclone.conf` 和 `m2-rclone.cmd` 文件复制到您的主文件夹 `C:\Users\PhotonUser\My Files\Home Folder`。
2. 使用您的 AWS 访问密钥和相应的密钥以及您的密钥更新`m2-rclone.conf`配置参数 AWS 区域。

    ```
    [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. 在 `m2-rclone.cmd` 中，进行以下更改：

   - 将 `amzn-s3-demo-bucket` 更改为您的 Amazon S3 存储桶名称。例如 `m2-s3-mybucket`。
   - 将 `your-s3-folder-key` 更改为您的 Amazon S3 存储桶密钥。例如 `myProject`。
   - 将 `your-local-folder-path` 更改为要从包含应用程序文件的 Amazon S3 存储桶同步应用程序文件的目录路径。例如 `D:\PhotonUser\My Files\Home Folder\m2-new`。此同步目录必须是主文件夹的子目录， WorkSpaces 应用程序才能在会话开始和结束时对其进行正确备份和恢复。

```
:loop
timeout /T 10
"C:\Program Files\rclone\rclone.exe" sync m2-s3:amzn-s3-demo-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. 打开 Windows 命令提示符，使用 cd 命令切换到 `C:\Users\PhotonUser\My Files\Home Folder`（如果需要），然后运行 `m2-rclone.cmd`。此命令脚本连续循环运行，每 10 秒将您的 Amazon S3 存储桶和密钥同步到本地文件夹。您可以根据需要调整超时。您应该在 Windows 文件资源管理器的 Amazon S3 存储桶中看到该应用程序的源代码。

要将新文件添加到您正在处理的文件集中或更新现有文件，请将文件上传到 Amazon S3 存储桶，这些文件将在 `m2-rclone.cmd` 中定义的下一次迭代中同步到您的目录。同样，如果要删除某些文件，请将其从 Amazon S3 存储桶中删除。下次同步操作会将其从您的本地目录中删除。

步骤 3：克隆存储库

1. 导航到浏览器窗口左上角的应用程序选择器菜单，然后选择“Enterprise Developer”。
2. 在您的主文件夹中完成 Enterprise Developer 所需的工作区创建，选择`C:\Users\PhotonUser\My Files\Home Folder`（又或 `D: \PhotonUser\My Files\Home Folder`）作为工作区的位置。

3. 在 Enterprise Developer 中，进入项目资源管理器，右键单击并选择 “导入...”、“Git”、“Git”、“从 Gi t Clone URI 中的项目” 来克隆 CodeCommit 仓库。然后，输入您的 CodeCommit特定登录凭据并完成 Eclipse 对话框以导入代码。

中的 CodeCommit git 存储库现已克隆到您的本地工作区中。

您的 Enterprise Developer 工作区现已准备就绪，可以开始对您的应用程序进行维护工作。特别是，您可以使用与 Enterprise Developer 集成的本地企业服务器 (ES) 实例以交互方式调试和运行应用程序，以便在本地验证您的更改。

> **Note**
>
> 本地企业开发人员环境（包括本地企业服务器实例）在 Windows 下运行，而 AWS 大型机现代化则在 Linux 下运行。我们建议您在向 AWS 大型机现代化提供的 Linux 环境中运行补充测试，然后再将新应用程序投入生产。 CodeCommit

后续会话

当您选择 WorkSpaces 应用程序管理下的文件夹（例如用于克隆 CodeCommit 存储库的主文件夹）时，该文件夹将在各个会话中透明地保存和恢复。您可以在下次需要使用该应用程序时完成以下步骤：

1. 根据欢迎电子邮件中收到的网址开始与 WorkSpaces 应用程序的会话。
2. 使用您的电子邮件和永久密码登录。
3. 选择 Enterprise Developer 堆栈。
4. 启动 `Rclone` 以连接（见上文）到 Amazon S3 支持的磁盘（如果您使用此选项共享工作区文件）。
5. 启动 Enterprise Developer 来完成工作。

清理资源

如果您不再需要为本教程创建的资源，请将其删除，以免继续产生费用。完成以下步骤：

- 删除您为本教程创建的 CodeCommit 存储库。有关更多信息，请参阅《AWS CodeCommit 用户指南》中的删除 CodeCommit 存储库。
- 删除您为本教程创建的数据库。有关更多信息，请参阅删除数据库实例。

AWS 大型机现代化中可用的批处理工具

大型机应用程序通常使用批处理实用程序来执行特定功能，例如对数据进行排序、使用 FTP 传输文件、将数据加载到数据库（例如）DB2、从数据库中卸载数据等。

当您将应用程序迁移到 AWS 大型机现代化时，您需要功能等同的替代实用程序，这些实用程序可以执行与您在大型机上使用的相同任务。其中一些实用程序可能已经作为 AWS 大型机现代化运行时引擎的一部分提供，但我们提供了以下替代实用程序：

- M2SFTP – 使用 SFTP 协议启用安全的文件传输。
- M2WAIT – 等待指定时间后，再继续执行批处理作业中的下一步操作。
- TXT2PDF-将文本文件转换为 PDF 格式。
- M2DFUTIL - 为数据集提供备份、恢复、删除和复制功能，类似于大型机 ADRDSSU 实用程序所提供的支持。
- M2RUNCMD-允许你直接从 JCL 运行 Rocket Software（前身为 Micro Focus）命令、脚本和系统调用。

我们根据客户反馈开发了这些批处理实用程序，旨在使其提供与大型机实用程序相同的功能。目标是尽可能顺利地从大型机过渡到 AWS 大型机现代化。

主题

二进制位置

这些实用程序已预装在 Rocket Enterprise Developer (ED) 和火箭软件 (ES) 产品上。您可以在以下位置找到 ED 和 ES 所有变体的实用程序：

- Linux：`/opt/aws/m2/microfocus/utilities/64bit`
- Windows（32 位）：`C:\AWS\M2\MicroFocus\Utilities\32bit`

- Windows（64 位）：`C:\AWS\M2\MicroFocus\Utilities\64bit`

M2SFTP 批处理实用程序

M2SFTP 是一个 JCL 实用程序，旨在使用安全文件传输协议 (SFTP) 在系统之间执行安全的文件传输。该程序使用 Putty SFTP 客户端 `psftp` 来执行实际的文件传输。该程序的工作原理与大型机 FTP 实用程序类似，使用用户和密码进行身份验证。

> **Note**
> 不支持使用公钥进行身份验证。

要将您的大型机 FTP JCLs 转换为使用 SFTP，请更改为`PGM=FTP`。`PGM=M2SFTP`

主题

支持的平台

您可以在以下任何平台上使用 M2SFTP：

- AWS 大型机现代化火箭软件（前身为 Micro Focus）托管
- 火箭软件运行时（在亚马逊上 EC2）
- 火箭软件企业开发者 (ED) 和 Rocket Software 企业服务器 (ES) 产品的所有变体。

安装依赖项

在 Windows 上安装 Putty SFTP 客户端

- 下载并安装 Putty SFTP 客户端。

在 Linux 上安装 Putty SFTP 客户端：

- 运行以下命令来安装 Putty SFTP 客户端：

```
sudo yum -y install putty
```

配置 M2SFTP 以实现 AWS 大型机现代化托管

如果您迁移的应用程序在 AWS 大型机现代化管理版上运行，则需要按如下方式配置 M2SFTP。

- 为 MFFTP 设置相应的 Rocket 企业服务器环境变量。下面是几个示例：
  - MFFTP_TEMP_DIR
  - MFFTP_SENDEOL
  - MFFTP_TIME
  - MFFTP_ABEND

  您可以根据需要设置任意数量的变量。您可以在 JCL 中使用 ENVAR DD 语句设置它们。有关这些变量的更多信息，请参阅 Micro Focus 文档中的 MFFTP 控制变量。

要测试配置，请参阅样本 JCLs。

在亚马逊上配置 M2SFTP 以实现 AWS 大型机现代化运行时 EC2 （包括 WorkSpaces 应用程序）

如果您迁移的应用程序在 Amazon AWS 的大型机现代化运行时上运行 EC2，请按如下方式配置 M2SFTP。

1. 更改 Micro Focus JES 程序路径以包含批处理实用程序的二进制位置。如果您需要指定多个路径，请在 Linux 上使用冒号 (:) 分隔路径，在 Windows 上使用分号 (;) 分隔路径。

   - Linux：`/opt/aws/m2/microfocus/utilities/64bit`
   - Windows（32 位）：`C:\AWS\M2\MicroFocus\Utilities\32bit`
   - Windows（64 位）：`C:\AWS\M2\MicroFocus\Utilities\64bit`

2. 为 MFFTP 设置相应的 Rocket 企业服务器环境变量。下面是几个示例：

   - `MFFTP_TEMP_DIR`
   - `MFFTP_SENDEOL`
   - `MFFTP_TIME`
   - MFFTP_ABEND

   您可以根据需要设置任意数量的变量。您可以在 JCL 中使用 `ENVAR DD` 语句设置它们。有关这些变量的更多信息，请参阅 Micro Focus 文档中的 MFFTP 控制变量。

要测试配置，请参阅样本 JCLs。

样本 JCLs

要测试安装，您可以使用以下其中一个示例 JCL 文件。

M2 SFTP1 .jcl

此 JCL 展示了如何调用 M2SFTP，以便将文件发送到远程 SFTP 服务器。请注意 `ENVVAR DD` 语句中设置的环境变量。

```
//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
/*
//*
//
```

M2 SFTP2 .jcl

此 JCL 展示了如何调用 M2SFTP，以便接收来自远程 SFTP 服务器的文件。请注意 ENVVAR DD 语句中设置的环境变量。

```
//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**
> 我们强烈建议将 FTP 凭证存储在 NETRC 文件中，并限制只有经过授权的用户才能访问。

Putty SFTP (PSFTP) 客户端命令参考

PSFTP 客户端不支持所有 FTP 命令。以下列表显示了 PSFTP 支持的所有命令。

命令	说明
!	运行本地命令
bye	完成 SFTP 会话
cd	更改远程工作目录
chmod	更改文件权限和模式
关闭	完成 SFTP 会话但不退出 PSFTP

命令	说明
del	删除远程服务器上的文件
dir	列出远程文件
exit	完成 SFTP 会话
入	将文件从服务器下载到本地机器上
help	提供帮助
lcd	更改本地工作目录
lpwd	打印本地工作目录
ls	列出远程文件
mget	一次下载多个文件
mkdir	在远程服务器上创建目录
mput	一次上传多个文件
mv	移动或重命名远程服务器上的文件
open	连接到主机
put	将文件从本地机器上传到服务器
pwd	打印远程工作目录
quit	完成 SFTP 会话
reget	继续下载文件
ren	移动或重命名远程服务器上的文件
reput	继续上传文件
rm	删除远程服务器上的文件

命令	说明
rmdir	删除远程服务器上的目录

后续步骤

要使用 SFTP 将文件上传和下载到亚马逊简单存储服务，您可以将 M2SFTP 与结合使用 AWS Transfer Family，如以下博客文章中所述。

- 使用 AWS SFTP 逻辑目录构建简单的数据分发服务
- 启用密码身份验证以供 AWS Transfer for SFTP 使用 AWS Secrets Manager

M2WAIT 批处理理实用程序

M2WAIT 是一个大型机实用程序，允许您在 JCL 脚本中引入等待期，即通过以秒、分钟或小时为单位指定持续时间。您可以直接从 JCL 调用 M2WAIT，方法是将要等待的时间作为输入参数传递。在内部，M2WAIT 程序调用 Rocket Software（前身为 Micro Focus）提供的模块C$SLEEP等待指定的时间。

> **Note**
>
> 您可以使用 Micro Focus 别名来替换 JCL 脚本中的名称。有关更多信息，请参阅 Micro Focus 文档中的 JES 别名。

主题

- 支持的平台
- 配置 M2WAIT 以实现 AWS 大型机现代化托管
- 在亚马逊上配置 M2WAIT 以实现 AWS 大型机现代化运行时 EC2 （包括 WorkSpaces 应用程序）
- 示例 JCL

支持的平台

您可以在以下任何平台上使用 M2WAIT：

- AWS 大型机现代化火箭软件（前身为 Micro Focus）托管

- 火箭软件运行时（在亚马逊上 EC2）
- 火箭软件企业开发者 (ED) 和 Rocket Software 企业服务器 (ES) 产品的所有变体。

配置 M2WAIT 以实现 AWS 大型机现代化托管

如果您迁移的应用程序在 AWS 大型机现代化管理版上运行，则需要按如下方式配置 M2WAIT。

- 通过传递输入参数（如示例 JCL 中所示）在 JCL 中使用 M2WAIT 程序。

在亚马逊上配置 M2WAIT 以实现 AWS 大型机现代化运行时 EC2（包括 WorkSpaces 应用程序）

如果您迁移的应用程序在 Amazon AWS 的大型机现代化运行时上运行 EC2，请按如下方式配置 M2WAIT。

1. 更改 Micro Focus JES 程序路径以包含批处理理实用程序的二进制位置。如果您需要指定多个路径，请在 Linux 上使用冒号 (`:`) 分隔路径，在 Windows 上使用分号 (`;`) 分隔路径。
   - Linux：`/opt/aws/m2/microfocus/utilities/64bit`
   - Windows（32 位）：`C:\AWS\M2\MicroFocus\Utilities\32bit`
   - Windows（64 位）：`C:\AWS\M2\MicroFocus\Utilities\64bit`
2. 通过传递输入参数（如示例 JCL 中所示）在 JCL 中使用 M2WAIT 程序。

示例 JCL

要测试安装，您可以使用 `M2WAIT1.jcl` 程序。

此示例 JCL 展示了如何调用 M2WAIT 并采用几个不同的持续时间将其进行传递。

```
//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=*
//*
//
```

TXT2PDF 批处理实用程序

TXT2PDF 是一种大型机实用程序，通常用于将文本文件转换为 PDF 文件。此实用程序使用相同的 TXT2 PDF 源代码（z/OS免费软件）。我们对其进行了修改，使其在 AWS 大型机现代化火箭软件（前身为 Micro Focus）运行时环境下运行。

主题

支持的平台

您可以在以下任何平台上使用 TXT2 PDF：

- AWS 大型机现代化火箭软件托管
- 火箭软件运行时（在亚马逊上 EC2）

- 火箭企业开发者 (ED) 和 Rocket Enterprise Server (ES) 产品的所有变体。

配置 TXT2 PDF 以实现 AWS 大型机现代化托管

如果您迁移的应用程序在 AWS 大型机现代化管理版上运行，请按以下方式配置 TXT2 PDF。

- 创建一个名为 AWS.M2.REXX.EXEC 的 REXX EXEC 库。下载这些 REXX 模块并将其复制到库中。
    - TXT2PDF.rex- TXT2 PDF z/OS 免费软件（已修改）
    - TXT2PDFD.rex- TXT2 PDF z/OS 免费软件（未修改）
    - TXT2PDFX.rex- TXT2 PDF z/OS 免费软件（已修改）
    - M2GETOS.rex – 检查操作系统类型（Windows 或 Linux）

要测试配置，请参阅示例 JCL。

在 Amazon 上为 AWS 大型机现代化运行时配置 TXT2 PDF EC2 （包括 WorkSpaces 应用程序）

如果您迁移的应用程序在 Amazon AWS 的大型机现代化运行时上运行 EC2，请按如下方式配置 TXT2 PDF。

1. 将 Rocket Software 环境变量MFREXX_CHARSET设置为适当的值，例如 ASCII 数据的 “A”。

    > Important
    >
    > 输入错误的值可能会导致数据转换问题（从 EBCDIC 到 ASCII），使生成的 PDF 无法读取或无法操作。我们建议设置 MFREXX_CHARSET 来匹配 MF_CHARSET。

2. 更改 Micro Focus JES 程序路径以包含批处理实用程序的二进制位置。如果您需要指定多个路径，请在 Linux 上使用冒号 (:) 分隔路径，在 Windows 上使用分号 (;) 分隔路径。
    - Linux：/opt/aws/m2/microfocus/utilities/64bit
    - Windows（32 位）：C:\AWS\M2\MicroFocus\Utilities\32bit
    - Windows（64 位）：C:\AWS\M2\MicroFocus\Utilities\64bit
3. 创建一个名为 AWS.M2.REXX.EXEC` 的 REXX EXEC 库。下载这些 REXX 模块并将其复制到库中。

- TXT2PDF.rex- TXT2 PDF z/OS 免费软件（已修改）
- TXT2PDFD.rex- TXT2 PDF z/OS 免费软件（未修改）
- TXT2PDFX.rex- TXT2 PDF z/OS 免费软件（已修改）
- M2GETOS.rex – 检查操作系统类型（Windows 或 Linux）

要测试配置，请参阅示例 JCL。

示例 JCL

要测试安装，您可以使用以下其中一个示例 JCL 文件。

TXT2PDF1.jcl

此示例 JCL 文件使用 DD 名称进行 TXT2 PDF 转换。

```
//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

此示例 JCL 使用 DSN 名称进行 TXT2 PDF 转换。

```
//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=*
//*
//
```

修改

为了让 TXT2 PDF 程序在 AWS 大型机现代化 Rocket Software 运行时环境中运行，我们进行了以下更改：

- 修改源代码以确保与 Rocket Software REXX 运行时兼容
- 为确保该程序可以同时在 Windows 和 Linux 操作系统上运行而进行了更改
- 为支持 EBCDIC 和 ASCII 运行时而进行了修改

参考

TXT2PDF 参考文献和源代码：

- 文本转 PDF 转换器
- z/OS免费软件 TCP/IP 和邮件工具
- TXT2PDF 用户参考指南

M2DFUTIL 批处理实用程序

M2DFUTIL 是一个 JCL 实用程序，提供数据集备份、还原、删除和复制功能，类似于大型机 ADRDSSU 实用程序提供的支持。该程序保留了 ADRDSSU 中的许多 SYSIN 参数，从而简化了迁移到这个新实用程序的过程。

主题

支持的平台

您可以在以下任何平台上使用 M2DFUTIL：

- Windows 上的火箭软件（前身为 Micro Focus）ES（64 位和 32 位）
- Linux 上的火箭软件 ES（64 位）

平台要求

M2DFUTIL 依赖于调用脚本来执行正则表达式测试。在 Windows 上，必须安装 Windows Services for Linux (WSL) 才能运行此脚本。

计划的未来支持

目前不可用但未来可用的大型机 ADRDSSU 实用程序功能包括：

- M2 托管
- VSAM
- 对文件重命名的 COPY 支持
- 对 RESTORE 的 RENAME 支持

- 多个 INCLUDE 和 EXCLUDE
- 通过 DSORG、CREDT、EXPDT 进行子查询的 BY 子句
- 用于重试入队失败的 MWAIT 子句
- 对 DUMP/RESTORE 的 S3 存储支持

资产位置

该实用程序的加载模块在 Linux 上称为 `M2DFUTIL.so`，在 Windows 上称为 `M2DFUTIL.dll`。可在以下位置找到此加载模块：

- Linux：`/opt/aws/m2/microfocus/utilities/64bit`
- Windows（32 位）：`C:\AWS\M2\MicroFocus\Utilities\32bit`
- Windows（64 位）：`C:\AWS\M2\MicroFocus\Utilities\64bit`

用于正则表达式测试的脚本称为 `compare.sh`。可在以下位置找到此脚本：

- Linux：`/opt/aws/m2/microfocus/utilities/scripts`
- Windows（32 位）：`C:\AWS\M2\MicroFocus\Utilities\scripts`

在亚马逊上配置 M2DFUTIL 或 AWS 大型机现代化运行时 EC2 （包括 AppStream 2.0）

使用以下内容配置您的 Enterprise Server 区域：

- 在 [ES-Environment] 中添加以下变量
  - `M2DFUTILS_BASE_LOC` - DUMP 输出的默认位置
  - `M2DFUTILS_SCRIPTPATH` - 资产位置中记录的 `compare.sh` 脚本的位置
  - `M2DFUTILS_VERBOSE` - [VERBOSE 或 NORMAL]。此变量控制 `SYSPRINT` 输出中的细节级别
- 验证加载模块路径是否已添加到 `JES\Configuration\JES Program Path` 设置中
- 验证实用程序目录中的脚本是否具有运行权限。您可以在 Linux 环境中使用 `chmod + x <script name>` 命令添加运行权限

一般语法

DUMP

提供将文件从当前编目位置复制到备份位置的功能。此位置当前必须是一个文件系统。

流程

DUMP 将执行下列操作：

1. 创建目标位置目录。
2. 将该目标位置目录归类为 PDS 成员。
3. 通过处理 INCLUDE 参数来确定要包含的文件。
4. 通过处理 EXCLUDE 参数来取消选择包含的文件。
5. 确定是否对转储的文件执行 DELETED 操作。
6. 将文件入队以待处理。
7. 复制这个文件。
8. 将已复制文件编目的 DCB 信息导出到目标位置中的副文件，以帮助将来执行 RESTORE 操作。

语法

```
DUMP
TARGET ( TARGET LOCATION  )    -
INCLUDE ( DSN. )
[ EXCLUDE ( DSN ) ]
[ CANCEL | IGNORE ]
[ DELETE ]
```

必需参数

以下是必需的 DUMP 参数：

- `SYSPRINT DD NAME` - 用于包含其他日志信息
- `TARGET` - 目标位置。它可以是：
  - 转储位置的完整路径
  - 在 M2DFUTILS_BASE_LOC 变量中定义的位置中创建的子目录名称
- `INCLUDE` - 要么是单个命名的 DSNAME，要么是有效的大型机 DSN 搜索字符串
- `EXCLUDE` - 要么是单个命名的 DSNAME，要么是有效的大型机 DSN 搜索字符串

可选参数

- CANCEL - 如果出现任何错误，则取消。已处理的文件将被保留
- （默认）IGNORE - 忽略任何错误并处理直到结束
- DELETE - 如果未出现 ENQ 错误，则会删除该文件并取消编目

删除

提供批量删除和取消编目文件的功能。文件未备份。

过程

DELETE 将执行下列操作：

1. 通过处理 INCLUDE 参数来确定要包含的文件。
2. 通过处理 EXCLUDE 参数来取消选择包含的文件。
3. 将文件入队以待处理。将处置设置为“OLD”、“DELETE”、“KEEP”。

语法

```
DELETE
INCLUDE ( DSN )
[ EXCLUDE ( DSN ) ]
[ CANCEL | IGNORE ]
[ DELETE ]
```

必需参数

以下是必需的 DELETE 参数：

- `SYSPRINT DD NAME` - 用于包含其他日志信息
- `INCLUDE` - 要么是单个命名的 DSNAME，要么是有效的大型机 DSN 搜索字符串
- `EXCLUDE` - 要么是单个命名的 DSNAME，要么是有效的大型机 DSN 搜索字符串

可选参数

- CANCEL - 如果出现任何错误，则取消。已处理的文件将被保留
- （默认）IGNORE - 忽略任何错误并处理直到结束

RESTORE

提供恢复以前使用 DUMP 备份的文件的功能。除非使用 RENAME 来更改已恢复的 DSNAME，否则文件将恢复到原始编目位置。

流程

RESTORE 将执行下列操作：

1. 验证源位置目录。
2. 通过处理目录导出文件来确定要包含的文件。
3. 通过处理 EXCLUDE 参数来取消选择包含的文件。
4. 将文件入队以待处理。
5. 未根据其导出信息编目的目录文件。
6. 如果已对文件进行编目并且导出目录信息相同，则当设置了 REPLACE 选项时，RESTORE 将替换编目的数据集。

语法

```
RESTORE
SOURCE ( TARGET LOCATION )
INCLUDE ( DSN )
[ EXCLUDE ( DSN ) ]
[ CANCEL | IGNORE ]
[ REPLACE]
```

必需参数

以下是必需的 RESTORE 参数：

- `SYSPRINT DD NAME` - 用于包含其他日志信息
- `SOURCE` - 源位置。它可以是：
  - 转储位置的完整路径
  - 在 M2DFUTILS_BASE_LOC 变量中定义的位置中创建的子目录名称
- `INCLUDE` - 要么是单个命名的 DSNAME，要么是有效的大型机 DSN 搜索字符串
- `EXCLUDE` - 要么是单个命名的 DSNAME，要么是有效的大型机 DSN 搜索字符串

可选参数

- CANCEL - 如果出现任何错误，则取消。已处理的文件已保留
- （默认）IGNORE - 忽略任何错误并处理直到结束
- REPLACE - 如果正在恢复的文件已经编目并且目录记录相同，则替换已编目文件

样本 JCLs

DUMP 作业

此作业将创建一个名为 `TESTDUMP` 的子目录。这是 M2DFUTILS_BASE_LOC 变量指定的默认备份位置。它将为这个名为 `M2DFUTILS.TESTDUMP` 的备份创建一个 PDS 库。导出的目录数据存储在名为 `CATDUMP.DAT` 的备份目录中的行序文件中。所有选定文件都将复制到此备份目录。

```
//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
/*
//
```

DELETE 作业

此作业将从目录中删除与 INCLUDE 参数匹配的所有文件。

```
/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
```

```
/*
 //
```

RESTORE 作业

此作业将从 TESTDUMP 备份位置恢复与 INCLUDE 参数匹配的文件。如果编目文件与 CATDUMP 导出中的文件相同，并且指定了 REPLACE 选项，则编目文件将被替换。

```
//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
/*
//
```

M2RUNCMD 批处理实用程序

你可以使用 M2RUNCMD（一个批处理实用程序）直接从 JCL 运行 Rocket Software（前身为 Micro Focus）命令、脚本和系统调用，而不必从终端或命令提示符处运行它们。命令的输出将记录到批处理作业的后台处理日志中。

主题

- 支持的平台
- 在亚马逊上配置 M2RUNCMD 以实现 AWS 大型机现代化运行时 EC2（包括 AppStream 2.0）
- 样本 JCLs

支持的平台

您可以在以下平台上使用 M2RUNCMD：

- 火箭软件运行时（在亚马逊上 EC2）
- 火箭软件企业开发者 (ED) 和 Rocket Software 企业服务器 (ES) 产品的所有变体。

在亚马逊上配置 M2RUNCMD 以实现 AWS 大型机现代化运行时 EC2（包括 AppStream 2.0）

如果您迁移的应用程序在 Amazon AWS 的大型机现代化运行时上运行 EC2，请按如下方式配置 M2RUNCMD。

- 更改 Micro Focus JES 程序路径以包含批处理实用程序的二进制位置。如果您必须指定多个路径，请在 Linux 上使用冒号 (:) 分隔路径，在 Windows 上使用分号 (;) 分隔路径。
  - Linux：`/opt/aws/m2/microfocus/utilities/64bit`
  - Windows（32 位）：`C:\AWS\M2\MicroFocus\Utilities\32bit`
  - Windows（64 位）：`C:\AWS\M2\MicroFocus\Utilities\64bit`

样本 JCLs

要测试安装，您可以使用以下任一示例 JCLs。

RUNSCRL1.jcl

此示例 JCL 创建一个脚本并运行它。第一步创建一个名为 `/tmp/TEST_SCRIPT.sh` 的脚本，其中包含来自 SYSUT1 流内数据的内容。第二步设置运行权限并运行第一步中所创建的脚本。你也可以选择只执行第二步来运行已经存在的 Rocket Software 和系统命令。

```
//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

运行的命令或脚本的输出将写入 SYSOUT 日志。对于每个已执行的命令，它都会显示命令、输出和返回代码。

```
************ 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

此示例 JCL 使用 RUNCMD 来运行多个命令。

```
//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 大型机现代化中的文件传输

借助 AWS Mainframe Modernization 文件传输功能，您可以将大型机数据集传输并转换到 Amazon S3，以用于大型机现代化、迁移和增强用例。此功能简化了将数据集从大型机传输到的 AWS Cloud 过程。主要特征包括：发现源大型机数据集和构件，提高可扩展性和效率，以便更快地将数据传输到 Amazon S3。文件传输功能支持各种大型机数据集类型，例如顺序、PDS、GDS、GDG 和 VSAM KSDS。该服务可将数据集传输到中间 Amazon S3 存储桶，将其转换为指定的目标代码页，然后将其移动到所需的目标 S3 存储桶。

主题

什么是 AWS Mainframe Modernization 文件传输功能？

借助 AWS 大型机现代化文件传输，您可以使用完全托管的服务传输和转换数据集和文件，从而加快和简化 AWS 大型机现代化服务和 Amazon S3 的现代化、迁移和增强用例。

主题

AWS Mainframe Modernization 文件传输功能的优势

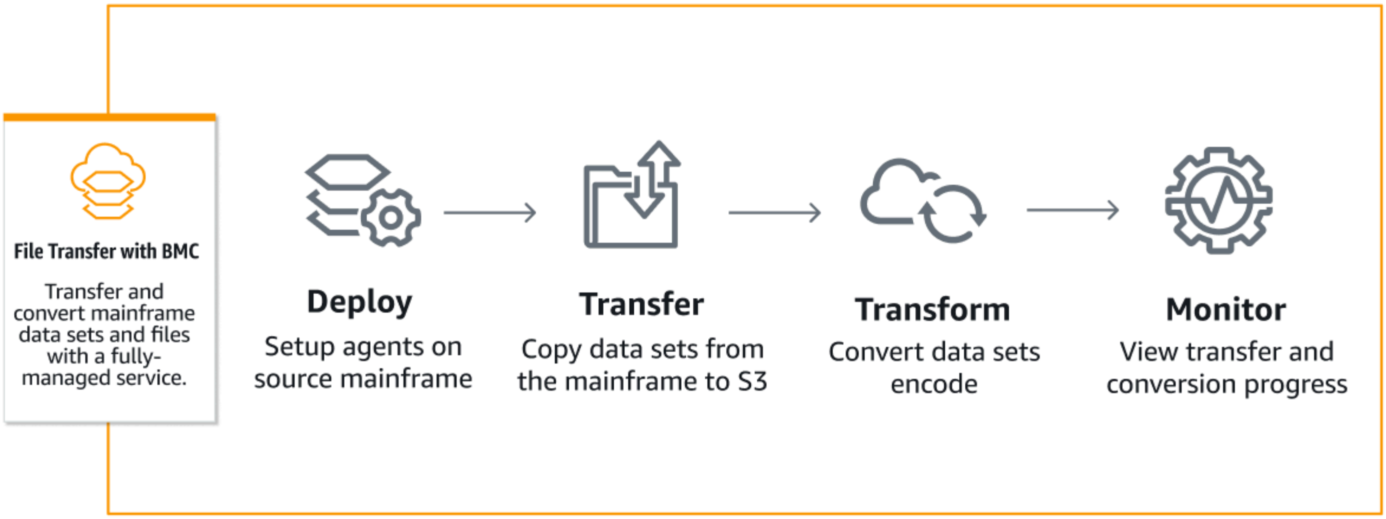
借助 AWS Mainframe Modernization 文件传输功能，您可以将数据集从大型机传输到 Amazon S3。一些优势包括：

- 发现源大型机数据集和构件
- 自动传输和数据集转换

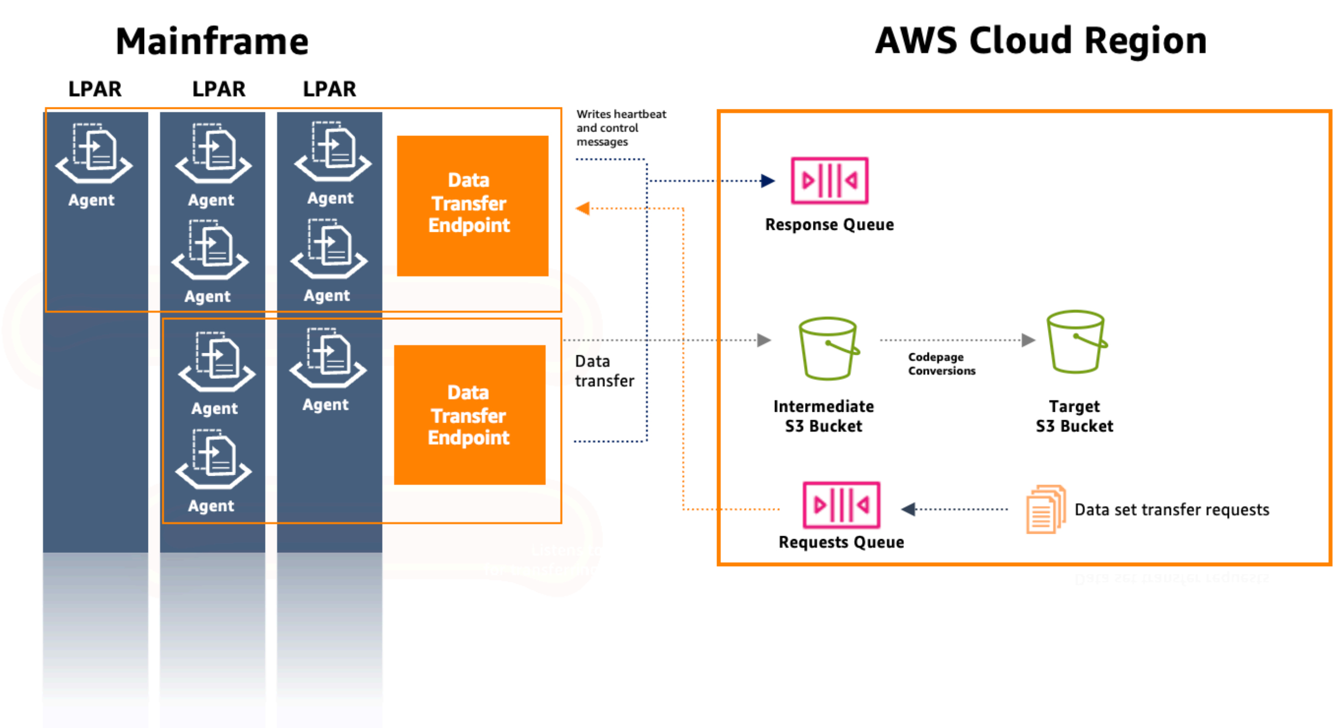
- 可扩展性、效率和速度提升，可更快地将数据集传输到 AWS

AWS Mainframe Modernization 文件传输功能的工作原理

下图从概念层面概述了 AWS Mainframe Modernization 文件传输功能的工作原理。



下图是 AWS Mainframe Modernization 文件传输功能的架构概述。



安装文件传输代理

您可以使用本文档作为 step-by-step指南，在源主机上安装代理。

> **Note**
> 本指南仅面向大型机系统程序员。

主题

步骤 1：为 M2 代理创建一个 zFS 数据集

使用下面的 JCL（作业控制语言）为 M2 代理安装创建一个 zFS：

```
DEFINE   EXEC   PGM=IDCAMS
SYSPRINT DD     SYSOUT=A
SYSIN    DD     *
 DEFINE CLUSTER (NAME(yourhlq.M2AGENT.ZFS) -
 VOLUMES(*) -
 LINEAR CYL(1000 200))
```

步骤 2：将数据集格式化为 zFS

创建数据集后，将其格式化为 zFS 文件系统。

一种方法是使用以下 JCL：

```
FORMAT  EXEC PGM=IOEAGFMT,
PARM=('-aggregate yourhlq.M2AGENT.ZFS -size 1200') ,
SYSPRINT DD  SYSOUT=*
```

提交此作业并检查其是否成功完成。

步骤 3：挂载文件系统

要挂载文件系统，请使用 MOUNT 命令。您可以在 ISPF 的命令行中或批量挂载文件系统。

例如：

```
MOUNT FILESYSTEM('yourhlq.M2AGENT.ZFS') TYPE(ZFS) MODE(RDWR) MOUNTPOINT('/usr/lpp/aws/
m2-agent')
```

您将在步骤 6 中使用此挂载点。

> **Note**
> 定义挂载路径是可选的，您应该使用现有目录来定义挂载路径。

步骤 4：验证挂载

使用 D OMVS,F 命令或在 Unix 系统服务（USS）中进行检查，来验证文件系统是否已正确挂载。

步骤 5：输入 OMVS

使用以下命令输入 OMVS：

```
TSO OMVS
```

步骤 6：设置代理安装目录环境变量

使用以下命令设置代理安装目录环境：

```
export AGENT_DIR=/usr/lpp/aws/m2-agent
```

> **Note**
> 挂载点在步骤 3 中定义。

步骤 7：设置工作目录环境变量

使用以下命令设置工作目录环境变量：

```
export WORK_DIR=$AGENT_DIR/tmp
```

步骤 8：创建工作目录

使用以下命令设置工作目录环境：

```
mkdir -p $WORK_DIR
```

步骤 9：复制代理 tar 文件并复制工作目录

使用 M2 代理链接从 AWS 下载代理 tar 文件。

传输机制取决于您的环境，但请确保以二进制模式传输该 tar 文件。

步骤 10：完成代理安装

按照以下步骤完成代理安装。

1. 使用以下命令将 m2 代理版本环境变量设置为当前正在安装的版本：

   ```
   export M2_AGENT_VERSION=1.0.0
   ```

2. 使用以下命令解压缩代理 tar 软件包：

   ```
   tar -xpf m2-agent-$M2_AGENT_VERSION.tar -C $AGENT_DIR
   ```

3. 使用以下命令创建指向当前代理安装目录的 `current-version` 符号链接：

```
ln -s $AGENT_DIR/m2-agent-v$M2_AGENT_VERSION $AGENT_DIR/current-version
```

4. 更新并提交 CPY#PDS 以创建文件传输代理数据集。

   > **Note**
   >
   > JCL 使用 SYS2.AWS.M2 HLQ。

   要创建文件传输代理，请更新稍后将在 JCL 中使用的三个符号变量 HLQ（高级限定符）、VOLSER 和 AGNTPATH：

   ```
   oedit $AGENT_DIR/current-version/installation/CPY#PDS
   ```

> **Note**
>
> 此 JCL 专为在大型机上设置代理安装的某些方面而量身定制。它分配必要的数据集，然后将特定文件从 Unix 文件系统复制到这些数据集。

配置文件传输代理

安装文件传输代理后，请按照以下步骤配置该代理。如果您需要安装新的代理，请按照the section called "安装文件传输代理"页面上的说明进行操作。

主题

步骤 1：配置权限和已启动任务控制（STC）

1. 按照说明，更新并提交其中一个 SYS2.AWS.M2.SAMPLIB(SEC#RACF)（用于设置 RACF 权限）或 SYS2.AWS.M2.SAMPLIB(SEC#TSS)（用于设置 TSS 权限）。这些成员由前面的 CPY#PDS 步骤创建。

   > **Note**
   >
   > SYS2.AWS.M2应替换为安装期间选择的高级限定符 (HLQ)。

2. 如果默认文件传输代理目录路径（/usr/lpp/aws/m2-agent）已更改，请更新 SYS2.AWS.M2.SAMPLIB(M2AGENT) STC JCL 中的 PWD 导出。
3. 根据您的网站标准更新 PROC：
   a. 根据您的安装要求更新 PROC 卡。
   b. 使用更新 STEPLIB。M2 LOADLIB PDSE ALIAS
   c. 编辑 PWD 以指向代理安装路径（仅包含此路径）。
   d. JAVA_HOME如有必要，请更新。
4. 更新 SYS2.AWS.M2.SAMPLIB(M2AGENT) JCL 并将其复制到PROCLIB串联 PROCLIBs 中的SYS1.PROCLIB或其中一个。
5. 使用以下命令将 SYS2.AWS.M2.LOADLIB 添加到 APF 列表：

   ```
   SETPROG APF ADD DSNAME(SYS2.AWS.M2.LOADLIB) SMS
   ```

6. 将代理的群组和所有者设置为代理 user/group (M2USER/M2GROUP)。在 OMVS 中使用以下命令：

   ```
   chown -R M2USER:M2GROUP $AGENT_DIR/current-version
   ```

   > **Note**
   >
   > 使用您在安全定义作业中使用的名称编辑 M2USER 和 M2GROUP。

步骤 2：创建 Amazon S3 存储桶

AWS Mainframe Modernization 文件传输功能需要一个中间 Amazon S3 存储桶作为工作区。建议专门为此创建一个存储桶。

（可选）为已传输的数据集创建一个新的目标 Amazon S3 存储桶。或者，您也可以使用现有的 Amazon S3 存储桶。有关创建 Amazon S3 存储桶的更多信息，请参阅创建存储桶。

步骤 3：创建用于加密的 AWS KMS 客户托管密钥

要在中创建客户管理的密钥 AWS KMS

1. 打开 AWS KMS 控制台，网址为https://console.aws.amazon.com/kms。
2. 在左侧导航窗格中，选择客户托管密钥。
3. 选择创建密钥。
4. 在配置密钥下，选择密钥类型为对称，将密钥用法选择为加密和解密。使用其他默认配置。
5. 选择下一步。
6. 在添加标签中，为密钥添加别名和描述。
7. 选择下一步。
8. 在定义密钥管理权限下，至少选择一个管理此密钥的 IAM 用户和角色。
9. 选择下一步。
10. 或者，在 “定义密钥管理权限” 下，选择至少一个可以使用此密钥的 IAM 用户和角色。
11. 选择下一步。
12. 在编辑密钥策略部分中，选择编辑，然后向密钥策略添加以下语法。这样， AWS 大型机现代化服务就可以读取和使用这些密钥进行加密/解密。

> **Important**
>
> 请将语句添加到现有语句中。不要替换策略中的已有内容。

```
{
    "Sid" : "Enable AWS M2 File Transfer Permissions",
    "Effect" : "Allow",
    "Principal" : {
        "Service" : "m2.amazonaws.com"
    },
```

```
    "Action" : [
        "kms:Encrypt",
        "kms:Decrypt"
    ],
   "Resource" : "*"
},
```

13. 选择下一步。
14. 在 “查看” 页面上，检查所有详细信息，然后选择 “完成”。

打开新创建的 KMS 密钥，复制并保存客户托管密钥的 ARN。该 ARN 稍后将在策略中使用。

步骤 4：为大型机凭证创建 AWS Secrets Manager 密钥

访问要传输的数据集需要大型机凭据，并且这些凭据必须作为 AWS Secrets Manager 机密存储。

创建密 AWS Secrets Manager 钥

1. 从 https://console.aws.amazon.com/secretsmanager 打开 Secrets Manager 控制台。
2. 选择存储新密钥。
3. 在选择密钥类型下，选择其他类型的密钥。
4. 使用有权访问数据`userId`集的大型机用户 ID 的密钥值。为密码字段使用密钥值 `password`。
5. 对于加密密钥，请选择之前创建的 AWS 客户托管密钥。
6. 选择下一步。
7. 在配置密钥页面上，输入名称和描述。
8. 在同一页面上，编辑资源权限并使用以下资源策略，以便 AWS 大型机现代化服务可以对其进行访问。

   JSON

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

```
                  "secretsmanager:DescribeSecret" ],
      "Resource" : "*"
    } ]
}
```

9. 选择 “保存” 以保存更新的权限。
10. 选择下一步。
11. 跳过 “配置轮换” 页面，然后选择 “下一步”。
12. 在审核页面上，检查所有配置并选择存储以保存密钥。

> **Important**
>
> `userId` 和 `password` 密钥区分大小写，必须按所示方式输入。

步骤 5：创建 IAM 策略

创建包含代理所需权限的新策略

1. 使用 https://console.aws.amazon.com/iam 打开 IAM 控制台。
2. 在 “访问管理” 下选择 “策略”。
3. 选择创建策略。
4. 在指定权限页面的策略编辑器下，从可视化编辑器切换到 JSON 编辑器并将内容替换为以下模板：
5.
6. 用您的账户替换请求队列 ARN 和响应队列 ARN 中的 `111122223333`。

> **Note**
>
> 这两个通配符 ARN 与数据传输端点初始化期间创建的两个 Amazon SQS 队列相匹配。创建文件传输端点后，可以选择将这两个 ARN 替换为 Amazon SQS 中的实际值。

7. 将 `file-transfer-endpoint-intermediate-bucket-arn` 替换为之前创建的传输存储桶的 ARN。在末尾保留“/*”通配符。
8. `kms-key-arn`替换为之前创建的 AWS KMS 密钥的 ARN。
9. 选择下一步。

10. 在查看并创建页面上，添加策略名称和描述。
11. 选择创建策略。

步骤 6：创建具有长期访问凭证的 IAM 用户

创建允许大型机代理连接到您的 AWS 账户的 IAM 用户。代理将与该用户连接，然后承担您定义的角色，该角色具有使用 Amazon SQS 响应和请求队列以及将数据集保存到 Amazon S3 存储桶的权限。

创建此 IAM 用户

1. 导航到 https://console.aws.amazon.com/iam 的 IAM 控制台。
2. 在 “访问管理” 下选择 “用户”。
3. 选择创建用户。
4. 在用户详细信息下添加一个有意义的用户名。例如 `Configure-ft-agent`。
5. 选择下一步。
6. 在权限选项中，选择直接附加策略选项，但不要附加任何权限策略。这些权限将由将要附加的角色管理。
7. 选择下一步。
8. 查看详细信息，然后选择创建用户。
9. 创建用户后，选择该用户并打开安全凭证选项卡。
10. 在访问密钥下，选择创建访问密钥。
11. 然后，在提示输入用例时选择其他。
12. 选择下一步。
13. 或者，您可以设置描述标签，例如`Access key for configuring file transfer agent`。
14. 选择创建访问密钥。
15. 复制并安全保存生成的访问密钥和私有访问密钥。稍后将会使用这两个密钥。

有关创建 IAM 访问密钥的更多信息，请参阅管理 IAM 用户的访问密钥。

> **Important**
>
> 选择完成之前，保存在访问密钥创建向导的最后一页上显示的访问密钥和秘密访问密钥。这些密钥用于配置大型机代理，以后无法检索。

> **Note**
>
> 保存用于与 IAM 角色建立信任关系的 IAM 用户 ARN。

步骤 7：创建代理将承担的 IAM 角色

为代理创建新的 IAM 角色

1. 从 https://console.aws.amazon.com/iam 打开 IAM 控制台，并选择角色。
2. 选择创建角色。
3. 在选择可信实体页面中的可信实体类型下，选择自定义信任策略。
4. 将自定义信任策略替换为以下内容，并将 `<iam-user-arn>` 替换为之前创建的用户的 ARN。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [ {
         "Sid": "FileTransferAgent",
         "Effect": "Allow",
         "Principal": {
            "AWS": "<IAM-User-arn>"
         },
         "Action": "sts:AssumeRole"
    } ]
}
```

5. 选择下一步。
6. 在添加权限中，筛选出您之前创建的策略名称，然后选择该策略。
7. 选择下一步。
8. 命名该角色，然后选择创建角色。

> **Note**
> 保存该角色名称，稍后您将使用该名称来配置大型机代理。

步骤 8：配置代理

配置文件传输代理

1. 导航到 $AGENT_DIR/current-version/config。
2. 使用以下命令编辑代理的配置文件 appication.properties，以添加环境配置：

```
oedit $AGENT_DIR/current-version/config/application.properties
```

例如：

```
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>
zos.complex-name=<File_Transfer_Endpoint_Name>
```

其中：

- AWS_ACCOUNT_ID是 AWS 账户的 ID。
- AWS_IAM_ROLE_NAME 是在the section called “步骤 7：创建代理将承担的 IAM 角色”中创建的 IAM 角色的名称。
- AWS_IAM_ROLE_ACCESS_KEY 是在the section called “步骤 6：创建具有长期访问凭证的 IAM 用户”中创建的 IAM 用户的访问密钥
- AWS_IAM_ROLE_SECRET_KEY 是在the section called “步骤 6：创建具有长期访问凭证的 IAM 用户”中创建的 IAM 用户的访问密钥。
- AWS_S3_BUCKET_NAME 是使用数据传输端点创建的传输存储桶的名称。
- AWS_REGION 是您在其中配置文件传输代理的区域。

> **Note**
>
> 通过定义多个环境，您可以将文件传输代理传输到多个区域和账户。 AWS

- （可选）。`zos.complex-name` 是您在创建文件传输端点时创建的复杂名称。

> **Note**
>
> 只有当您想自定义与创建文件传输端点时所定义的名称相同的复杂名称（默认为您的 sysplex 名称）时，才需要填写此字段。有关更多信息，请参阅 the section called “创建数据传输端点”。

> **Important**
>
> 可以有几个这样的部分，只要括号中的索引 (`[0]`) 为每个部分递增。

必须重新启动代理才能使更改生效。

要求

1. 添加或删除参数时，必须停止并启动代理。在 CLI 中使用以下命令启动文件传输代理：

   ```
   /S M2AGENT
   ```

   要停止 M2 代理，请在 CLI 中使用以下命令：

   ```
   /P M2AGENT
   ```

2. 您可以通过定义环境条目将文件传输代理配置为将数据传输到多个区域和账户。 AWS

   > **Note**
   >
   > 将这些值替换为您之前创建并配置的参数值。

   ```
   #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
```

为文件传输功能创建数据传输端点

数据传输端点支持与源大型机的连接，并支持高可用性、可扩展性和简化的代理管理。单个代理安装在大型机上 LPARs ，可以组合成一个数据传输端点。当请求传输数据集时，数据传输端点中的一个代理将处理该特定传输。要启动数据传输，数据传输端点上必须至少有一个代理处于联机状态。

此过程假设您已完成为 AWS 大型机现代化做好准备 和在源大型机上配置文件传输代理中的步骤。

创建数据传输端点

要为文件传输创建数据传输端点，请在 AWS 大型机现代化控制台中执行以下步骤。

创建数据传输端点

1. 打开 AWS 大型机现代化控制台，网址为https://console.aws.amazon.com/m2/。
2. 在 AWS 区域 选择器中，选择要将文件从大型机传输到 Amazon S3 存储桶的区域。
3. 在数据传输端点页面的文件传输下，选择创建数据传输端点。
4. 在数据传输端点先决条件页面上，阅读所有说明，以确保您已在源大型机上完成这些步骤。确认后，选择下一步。
5. 在配置数据传输端点页面上，为数据传输端点添加基本信息。

    1. 在基本信息部分中，输入您的数据传输端点名称。

> **Note**
>
> 该数据传输端点名称必须与 Sysplex 名称一致，除非您在代理配置中指定一个复杂名称。

2. 可选的描述。
3. 用于加密密钥的 KMS 密钥。

> **Note**
>
> 您必须为 KMS 添加以下基于资源的策略，以便 AWS Mainframe Modernization 服务可以读取和使用这些密钥来进行加密/解密：
>
> ```
> {
>   "Sid" : "Enable AWS M2 Permissions",
>   "Effect" : "Allow",
>   "Principal" : {
>       "Service" : "m2.amazonaws.com"
>             },
>    "Action" : [
>        "kms:Encrypt",
>        "kms:Decrypt"
>              ],
>    "Resource" : "*"
> }
> ```

4. 指定中间数据的 S3 位置，这是从大型机传输的数据集在转换并传输到目标 Amazon S3 存储桶之前所存放的中间 S3 位置。

   .

> **Note**
>
> 建议您为传输任务创建新的 Amazon S3 存储桶。有关更多信息，请参阅创建存储桶。您也可以通过选择浏览 S3 选项来浏览现有 Amazon S3 存储桶。

5. 输入必填字段后，选择下一步。

6. 在审核并创建数据传输端点页面上，检查您是否已完成先决条件，并审核基本信息。确认后，选择创建数据传输端点。

您将被重定向到数据传输端点概述页面，可以在其中查看所有数据传输端点的列表。您还可以看到可用或发生故障的数据传输端点。

您还可以按名称搜索数据传输端点，并访问每个可用代理的其他信息。

在文件传输功能中创建传输任务

传输任务用于指定要从大型机传输到 Amazon S3 的数据集，并支持您选择代码页转换选项。

以下说明假定您已完成为 AWS 大型机现代化做好准备中的步骤并创建了the section called “创建数据传输端点”。

主题

- 创建传输任务
- 查看传输任务

创建传输任务

要在文件传输中创建传输任务，请在 AWS 大型机现代化控制台中执行以下步骤。

创建传输任务

> **Important**
> 您必须至少有一个数据传输端点才能创建新的传输任务。

1. 打开 AWS 大型机现代化控制台，网址为https://console.aws.amazon.com/m2/。
2. 在 AWS 区域 选择器中，选择要将文件从大型机传输到 Amazon S3 存储桶的区域。
3. 在传输任务页面上，您可以选择任何数据传输端点来创建传输任务。
4. 在创建传输任务页面上，设置传输任务的属性。如果您之前没有创建过任何传输任务，可以通过选择创建传输任务选项来创建第一个传输任务。

   - 在此页面上，输入传输任务的基本信息，包括传输任务名称、描述和密钥。

> **Note**
>
> - 使用为数据传输端点定义的 KMS 密钥来加密密钥。该密钥应包含使用 `userId` 和 `password` 密钥访问大型机上的数据集所需的大型机凭证。有关更多信息，请参阅 AWS Secrets Manager 密钥。
> - 您必须使用以下基于资源的策略配置密钥，以便 AWS 大型机现代化服务可以访问它来执行数据传输任务。
>
>   JSON
>
> ```
> {
>   "Version":"2012-10-17",
>   "Statement" : [ {
>     "Effect" : "Allow",
>     "Principal" : {
>         "Service" : "m2.amazonaws.com"
>     },
>     "Action" : [ "secretsmanager:GetSecretValue",
>                  "secretsmanager:DescribeSecret" ],
>     "Resource" : "*"
>   } ]
> }
> ```

> **Note**
>
> 当前支持的最大传输数据集大小为 90 GB。

- 接下来，选择将从中传输来自大型机的目标数据集的目标 Amazon S3 存储桶位置。
- 将选择先前选择的数据传输端点。您也可以从可用端点中选择另一个端点。

5. 选择下一步。
6. 在添加数据集页面上的传输任务配置部分中，可以选择以二进制模式配置传输任务，或者转换并传输数据集。

   - 以二进制模式传输选项支持您以跳过代码页转换并保留其记录描述字（RDW）字节的方式来传输数据集。

- 传输并转换数据集选项支持您通过为数据集设置源代码页和目标代码页的方式来传输数据集。您可以在 the section called “支持的源和目标代码页”页面上查看可用于文件传输功能的代码页。

7. 在在大型机上搜索数据集中输入查询，以在大型机中搜索要包含在传输任务中的数据集。选择查看数据集。

   可在大型机数据集搜索条件中使用以下通配符：

   - 单星号（*）作为限定符（在句点之间或最后一个句点之后）匹配该位置上的单个限定符。
   - 限定符中的单个星号（*）匹配该位置的零个或多个字符。
   - 双星号（**）作为限定符（在句点之间或最后一个句点之后）匹配该位置上的零个或多个限定符。
   - 限定符中的双星号（**）不是有效的查询。
   - 单个百分号（%）匹配该位置上的任何单个字母数字或国家字符。在每个限定符中最多可以使用八个百分号。

   > **Note**
   >
   > 建议始终以句点后跟双星号（.**）结束搜索条件，然后根据需要进一步优化搜索。

   有关通配符规则的更多信息，请参阅 IBM 文档中的筛选数据集名称。

8. 这些数据集将加载到大型机数据集部分下，您可以在其中搜索或选择一个或多个要为其配置代码页转换的数据集。这些选定的数据集将显示在已添加的数据集部分中。如果未加载任何数据集，您需要重新执行步骤 7。

   > **Note**
   >
   > 您可以从多个搜索查询中选择数据集并将其添加到传输任务中。

9. 在已添加的数据集部分中，您将看到数据集的名称、类型和卷名。

   > **Important**
   >
   > 对于传输并转换数据集选项，您需要为选定的每个数据集手动输入源代码页和目标代码页。源代码页是源数据集格式，目标代码页是用于转换数据集并将其存储在目标 Amazon S3 存储桶中的目标数据集格式。

10. 确认已添加的数据集部分中的数据集（以及“传输并转换数据集”选项的源代码和目标代码页）后，选择下一步。
11. 在审核并创建页面上，您可以审核或编辑传输任务的相关信息。
12. 然后，选择创建传输任务。

> **Important**
>
> 选择 “创建传输任务” 按钮将开始数据传输，根据AWS 大型机现代化定价页面进行计费。该账单基于以数据集大小衡量的传输数据量 (GB)。

查看传输任务

要在 “文件传输” 中查看传输任务，您必须在 AWS 大型机现代化控制台中执行以下步骤。

查看传输任务

1. 打开 AWS 大型机现代化控制台，网址为https://console.aws.amazon.com/m2/。
2. 在 AWS 区域 选择器中，选择要将文件从大型机传输到 Amazon S3 存储桶的区域。
3. 在传输任务页面上，选择数据传输端点以查看传输任务。
4. 对于已有传输任务的端点，这些任务将显示在传输任务部分下。您可以选择从此列表中查看任何传输任务的详细信息。

教程：开始使用 AWS Mainframe Modernization 文件传输

AWS Mainframe Modernization 文件传输功能支持您为大型机现代化、迁移和增强用例传输和转换大型机数据集。

执行本教程中的步骤，了解 AWS Mainframe Modernization 文件传输的工作原理。

概览

文件传输功能包括以下部分：

1. 安装在源大型机上的代理。

2. 直接从 AWS 大型机现代化管理服务控制台访问数据集发现、传输和转换功能。

作为用户，您可以将数据集从大型机传输到 Amazon S3 存储桶。

主题

步骤 1：将代理二进制文件 tar 包从传输 AWS 到大型机逻辑分区

从 M2-agent tar 链接下载 tar 文件。

步骤 2：在源大型机上配置文件传输代理

在此步骤中，您将在源大型机上配置并启动 AWS Mainframe Modernization 文件传输代理。需要代理来促进文件传输服务功能和源大型机之间的通信。每个大型机至少需要一个代理。可以启动多个代理，以实现高可用性和增强的可扩展性。

按照the section called “配置文件传输代理”指南中的说明，在大型机上完成文件传输代理的安装。

步骤 3：创建数据传输端点

按照the section called “创建数据传输端点”页面上的步骤创建新的数据传输端点。

步骤 4：创建传输任务

按照the section called “创建传输任务”页面上的步骤创建和管理您的传输任务。

步骤 5：查看传输任务进度

您可以在 AWS 大型机现代化控制台中查看传输任务的进度。有关详细信息，请参阅the section called “查看传输任务”部分。

AWS Mainframe Modernization 文件传输功能中支持的源和目标编码

AWS Mainframe Modernization 文件传输功能支持各种数据集类型和代码页转换选项。

大型机数据集类型

AWS Mainframe Modernization 文件传输功能支持以下大型机数据集类型：

- 非 VSAM：顺序（PS）、PDS、GDS、GDG
- VSAM 类型：KSDS

支持的代码页

AWS Mainframe Modernization 文件传输功能支持以下代码页进行数据集转换（从/到）：

“”、“BIG5BIG5_HKSCS”、“CESU_8”、“EUC_JP”、“EUC_KR”、“GB18030”、“”、“GBK”、“IBM00858”、“IBM
01145"、“IBM01146”、“IBM01147”、“IBM01148”、“IBM01149”、“IBM01149”、“IBM037"、"
026"、"047"、"、""、"0"、"、"、"
0"、"、"、"、"、"0"、"00"、" "、"
0"、"" "、” GB2312 IBM1 IBM1 IBM273 IBM277 IBM278 IBM28 IBM284 IBM285 IBM29
IBM297 IBM42 IBM424 IBM437 IBM5 IBM775 IBM85 IBM852IBM855”、“”、“IBM860”、IBM857
“”、“”、“”、“”、“”、“IBM861”、“”、“”、“IBM862”、“”、“0IBM863”、“”、“IBM864”、“IBM_THAI”、“ISO_2022_C
IBM87 0201”、“JIS_X0212_1990”、“_R”、“_U”、IBM865 IBM866 IBM868 IBM869 IBM871 KOI8
KOI8“SHIFT_JIS”、“TIS_620”、“US_ASCII”、“UTF_16”、“UTF_16BE”、“UTF_16LE”、“UTF_32BE”、“UTF_3
_HKSCS_2001”、“X_ _SOLARIS”、“X_EUCJP_OPEN”、“X_EUC_JP_LINUX”，“X_EUC_TW”、“X_
006”、“X_ 025”、“X_ 046”、“X_ 097”、“X_ 098”、“X_”、“X_”、“X_”、“X_”、“X_”、“X_”、“X_” BIG5
BIG5 IBM1 IBM1 IBM1 IBM1 IBM1 IBM1112 IBM1122
IBM1123IBM1124”、“X_”、“X_IBM1129”、“X_IBM1166”、“X_”、“X_IBM1364”、“X_IBM1381”、“X_
IBM29626
C”、“X_IBM1383”、“X_”、“X_”、“X_”、“X_IBM3”、“X_IBM33722”、“X_”、“X_IBM737”、“X_IBM833”、“X_”、
CIBM93”、“X_”、“X_IBM933”、“X_”、“X_IBM935”、“X_IBM937”、“X_”、“X_IBM939”、“X_ IBM942
CIBM942”、“X_”、“X_” IBM943 IBM943 IBM948 IBM949 IBM949 IBM950”、“X_”、“X_ 0”、“X_”、“X
IBM964 _ISO_2022_CN_CNS”、“X_ISO_2022_CN_GB”、“X_ISO_8859_11”、“X_JIS IBM97
0208”、“X_JISAUTODETECT”、“X_JOHAB”、“X_MACARABIC”、“X_MAC_MAC_MAC_11ISCII91"、“X_JIS
_0213"、“X_ 0_HKSCS”、“X_
0_HKSCS_XP”、“X_MSWIN_936”、“X_PCK”、“X_SJIS_0213”、“X_UTF_16LE_BOM”、“X_UTF_32BE_BOM

MS932 MS95
MS9532LE_BOM”、“X_WINDOWS_50220”、“X_WINDOWS_50221”、“X_WINDOWS_874”、“X_WINDOWS_
022j” ISO2

AWS Transform 用于实现大型机应用程序的现代化

AWS Transform for mainframe 通过自动执行复杂而耗时的任务，例如分析代码库、规划转换、生成技术文档、提取业务逻辑、分解代码、波浪规划以及执行重构和重构，使您能够更快地将传统的 COBOL 大型机应用程序现代化为 Java 应用程序。它通过利用生成式人工智能和自动化来降低大型机现代化的成本和复杂性，同时保留您的任务关键型业务逻辑。 AWS Transform自然语言界面和以目标为导向的方法使您可以控制转型，使您能够专注于战略优先事项，而自动化则可以处理现代化之旅的繁重工作。

有关功能和关键特性、高级演练以及人工参与输入和处理的更多信息，请参阅《AWS Transform 用户指南》中的大型机应用程序的迁移和现代化。

主要优势

AWS Transform对大型机应用程序进行现代化改造的能力具有多重好处。其中一些包括：

- 利用@@ 生成式 AI 加速大型机现代化之旅： AWS Transform 帮助您在几个月内将 COBOL 代码转换为现代 Java 代码，而不是 以年为单位的传统时间表。
- 弥合知识差距：为您的大型机应用程序 AWS Transform 生成全面的文档，弥合知识差距，做出更明智的决策。
- 保留关键业务逻辑： AWS Transform 保留传统系统的关键业务逻辑，同时将其重构为经过云优化的现代 Java 应用程序。
- 业务和技术逻辑域分解： AWS Transform 自动将大型机代码库分解为不同的业务域，以减少代码库分析和分解所需的手动工作和时间。
- H@@ uman in the Loop (HITL) 功能： AWS Transform 提供一种自主的、以目标为导向的方法，让您可以控制大型机现代化之旅。

大型机应用程序控制台转型演练

在 AWS Transform Web 体验中，您可以将大型机应用程序从 COBOL 转换为 Java。要了解如何使用此功能，请按照《AWS Transform 用户指南》中 “大型机应用程序转型” 页面中的所有步骤进行操作。

AWS 使用 Precission 复制大型机现代化数据

AWS 大型机现代化提供各种 Amazon 机器映像 (AMIs)。它们 AMIs 便于快速配置 Amazon EC2 实例，为将数据从大型机系统复制到 AWS 使用 Precisial 创建量身定制的环境。本指南提供了访问和使用这些内容所需的步骤 AMIs。

先决条件

- 确保您拥有可以创建 Amazon EC2 实例的 AWS 账户的管理员访问权限。
- 确认您计划创建 Amazon EC2 实例的地区是否提供 AWS 大型机现代化服务。查看按区域划分的可用 AWS 服务列表。
- 确定将在其中创建亚马逊 EC2 实例的亚马逊虚拟私有云（亚马逊 VPC）。
- 在 Amazon VPC 中创建 Amazon EC2 实例时，请确保关联的路由表具有互联网网关或 NAT 网关。

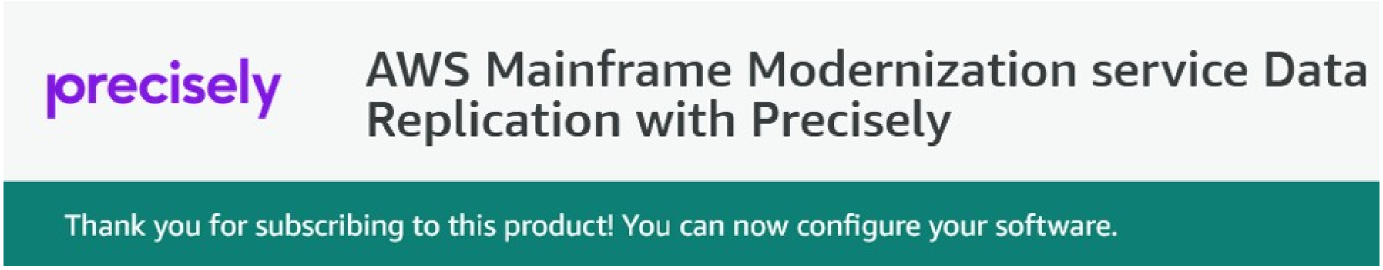
> Note
>
> 成功复制数据需要 AWS EC2 实例具有与 AWS Marketplace 的通信访问权限。如果 AWS Marketplace 出现连接问题，则复制过程将失败。

订阅亚马逊机器映像

当您订阅 AWS Marketplace 产品后，可以从该产品的 AMI 启动实例。

1. 登录 AWS 管理控制台 并在 https://console.aws.amazon.com/marketplac AWS Marketplace e 上打开控制台。
2. 选择管理订阅。
3. 根据您的用例，导航到以下任一链接：
   - 适用于 IBM z/OS 的数据复制：https://aws.amazon.com/marketplace/pp/prodview-doe2lroefogia
   - 适用于 IBM i 的数据复制：https://aws.amazon.com/marketplace/pp/prodview-iqrkflccxf7ko
4. 选择继续订阅。
5. 如果可以接受条款和条件，请选择接受条款。处理此订阅可能需要几分钟时间。

6. 等待出现感谢消息，如下所示。该消息确认您是否成功订阅了产品。



7. 在左侧导航窗格中，选择管理订阅。此视图显示您已订阅的所有订阅。

使用 Precist 启动 AWS 大型机现代化数据复制

1. 在 https://console.aws.amazon.com/marketplac AWS Marketplace e 上打开控制台。
2. 在左侧导航窗格中，选择管理订阅。
3. 找到您要启动的 AMI，选择启动新实例。
4. 在区域下，选择列为允许的区域。
5. 选择 “继续” 即可启动 EC2。此操作会将您带到 Amazon EC2 控制台。
6. 输入服务器的名称。
7. 选择与您的项目性能和成本要求相匹配的实例类型。建议的实例大小的起点是 `c5.2xLarge`。
8. 选择现有密钥对或创建新的密钥对并保存。有关密钥对的信息，请参阅亚马逊 EC2 用户指南中的亚马逊 EC2 密钥对和 Linux 实例。
9. 编辑网络设置并选择列为允许的 VPC 和相应的子网。
10. 选择现有安全组或者创建新的安全组。除了允许 SSH 访问（默认情况下在端口 22 上）之外，对于使用 Precision 服务器 EC2 实例进行数据复制，通常还允许 TCP 流量到达其默认端口 2626。
11. 为 Amazon EC2 实例配置存储。
12. 审核摘要并选择启动实例。要成功启动，实例类型必须有效。如果启动失败，请选择编辑实例配置并选择其他实例类型。
13. 看到成功消息后，选择连接到实例。
14. 打开亚马逊 EC2 控制台，网址为https://console.aws.amazon.com/ec2/。
15. 在左侧导航窗格中的实例菜单下，选择实例。
16. 在主窗格中，检查实例的状态。

创建 IAM 策略

要成功运行通过我们的 AWS Marketplace 列表部署 AWS 的大型机现代化 EC2 实例，您必须配置 IAM 角色和策略。这种专门定制的 IAM 设置不是可选的；它授权您的 Amazon EC2 实例与该服务进行交互。 AWS Marketplace IAM 角色和策略允许 AWS 大型机现代化准确记录使用数据，这对于精确计费至关重要。未能实现此配置可能会导致数据复制尝试失败和操作中断。

1. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在左侧的导航窗格中，选择策略。
3. 如果这是您首次选择策略，则会显示欢迎访问托管式策略页面。选择开始使用。
4. 在页面的顶部，选择创建策略。
5. 在策略编辑器部分，选择 JSON 选项。
6. 输入以下 JSON 策略。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": ["aws-marketplace:MeterUsage"],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

创建一个 IAM 角色

1. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在导航窗格中，选择角色，然后选择创建角色。
3. 在可信实体类型部分中，选择 AWS 服务。
4. 在 “用例” 部分的 “服务” 或 “用例” 下，选择 Amazon EC2。
5. 选择下一步。

6. 在策略列表中，从按类型筛选下拉列表中选择客户托管，然后输入您创建的策略的名称。选中该策略名称旁边的复选框。
7. 选择下一步。
8. 输入角色的名称和（可选）描述。
9. 查看信任策略和权限，然后选择创建角色。

将 IAM 角色附加到 Amazon EC2 实例

1. 打开亚马逊 EC2 控制台，网址为https://console.aws.amazon.com/ec2/。
2. 在导航窗格中，选择 Instances (实例)。
3. 选择您的亚马逊 EC2 实例。
4. 从操作菜单上，选择安全，然后选择修改 IAM 角色。
5. 选择要附加到您的实例的角色，然后选择更新 IAM 角色。

有关开始使用 IBM i AWS 的数据复制的更多信息，请参阅数据复制概述。

AWS Mainframe Modernization 使用 mLogica 进行代码转换

AWS Mainframe Modernization 使用 mLogica 进行代码转换（代码转换）是一项自动将z/OS大型机汇编器代码转换为 COBOL 的 AWS Mainframe Modernization 功能。您可以使用代码转换来提取汇编器图像，使用该 AWS CodeBuild 服务进行预期的代码转换。 AWS 账户

主题

- 什么是采用 mLogica 的汇编器转换？
- 了解汇编器转换的代码转换功能计费
- 代码转换功能概念
- 了解代码转换功能的组件和过程
- 教程：将代码从汇编器转换为 COBOL AWS Mainframe Modernization

什么是采用 mLogica 的汇编器转换？

AWS Mainframe Modernization 使用 mLogica 进行代码转换（代码转换）可自动将z/OS大型机汇编器代码转换为 COBOL。该服务在您的内部运行 AWS 账户 ，不会在外部传输或存储汇编器或 COBOL 源代码。 AWS 账户代码转换允许您的授权帐户使用该 AWS CodeBuild 服务提取汇编器映像，以进行预期的代码转换。

AWS Mainframe Modernization 使您能够为迁移的应用程序设置生成和持续 integration/continuous 交付 (CI/CD) 管道。这些版本和管道使用 AWS CodeBuild 和 Amazon S3 来提供此功能。 AWS CodeBuild 是一项完全托管的生成服务，它可以编译源代码、运行单元测试并生成可随时部署的工件。Amazon S3 是一项对象存储服务，可提供行业领先的可扩展性、数据可用性、安全性和性能。

主题

- 代码转换编译器
- 代码转换架构
- 自动化方法
- 安全性
- 其他资源

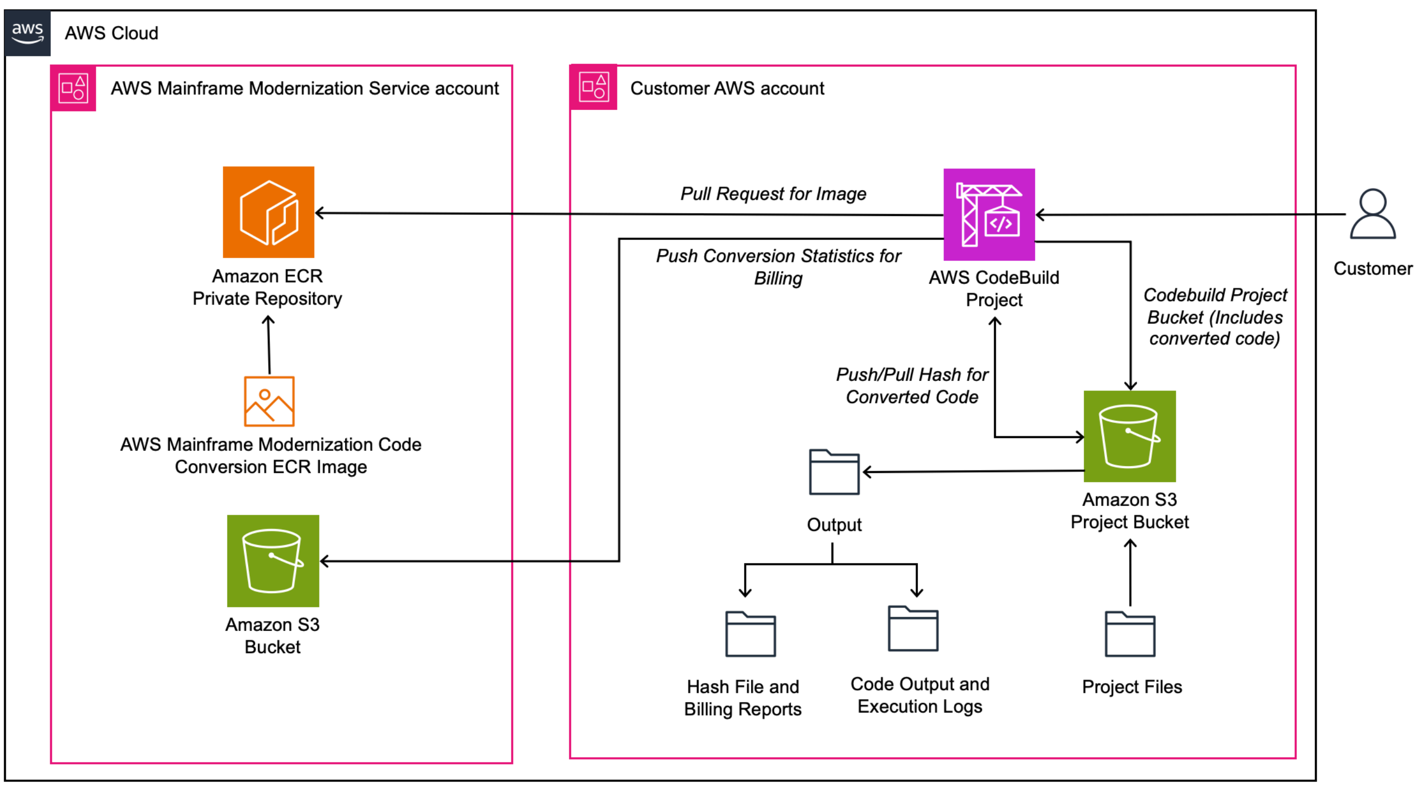
代码转换编译器

可以将代码转换功能配置为发出 COBOL，后者适合在具有不同编译器的多个目标环境中进行编译和运行。其中包括：

- 使用 Rocket Software（前身为 Micro Focus）和其他火箭企业服务器环境重新构建 M2
- 使用 NTT DATA Enterprise COBOL 重新构建 M2 平台 () UniKix
- mLogica LIBER*COBOL
- z/OS使用 IBM 企业 COBOL 的大型机
- Veryant isCOBOL

代码转换架构

以下是代码转换过程的架构图：



自动化方法

要使用代码转换 CodeBuild，需要将汇编器代码上传到 Amazon S3 存储桶，以便稍后配置转换参数并调用 CodeBuild 项目来执行转换过程中的每个步骤。目标 COBOL 代码会自动存储在 Amazon S3 存储桶中的指定路径中。

安全性

AWS Mainframe Modernization 代码转换支持转换，同时将所有源代码和目标代码保留在您的 AWS 账户。源汇编器代码、目标 COBOL 代码和配置文件存储在您的 Amazon S3 存储桶中。自动转换工具在您的 CodeBuild 环境中作为容器运行 AWS 账户。代码会始终保留在您的账户中。

要使转换工具能够访问您的 Amazon S3 存储桶，您需要向 AWS 服务 角色授予该存储桶的权限。配置时 CodeBuild，您将设置此服务角色，使其 CodeBuild 可以访问容器映像和访问您的 Amazon S3 存储桶。

其他资源

除此之外the section called “教程：将代码从汇编器转换为 COBOL”，还有一些其他资源，您可以在其中了解如何创建 CloudFormation 模板以及有关将 Assembler 转换为 COBOL 的其他信息。

- 从汇编器到 COBOL 的自动代码转换讲习会链接：https://catalog.workshops.aws/awsm2ccm-assembler-cobol/en-US。
- 博客文章：https://aws.amazon.com/blogs/migration-and-modernization/unlocking-new-potential-transform-your-assembler-programs-to-cobol-with-aws-mainframe-modernization/。

了解汇编器转换的代码转换功能计费

在进行实际转换之前，您需要参考此页面以了解代码转换功能的计费范围和流程。费用计算部分提到了对从汇编器到 COBOL 的转换按代码行收费的流程。

代码转换功能计费和范围

汇编器代码转换只有在完成转换步骤后才会在您的 AWS 账户 上生成费用（账单报告）。费用根据转换的代码行数计算。如果执行了多个转换步骤，例如添加了新的汇编器代码、更改了转换配置或应用了新版本容器，则只会对更改的行和/或新添加的行计算费用。我们不会对同一程序中相同代码行的转换收取两次费用。

> **Note**
>
> 更改了代码行的模块以及新程序或重命名程序中的所有代码行都会被收费。

为避免多次收费，代码转换功能会在项目存储桶中的 `<Project_bucket>/awsm2ccm-do-not-delete/<AWS_account_number>/Hash` 中为每个汇编器或宏模块存储一个编码二进制文件。这些编码文件不包含任何客户代码。

> **Important**
>
> 请勿手动编辑或删除这些文件。更改可能会导致对相同组件的转换多次计费。

AWS Mainframe Modernization 代码转换分析报告（“分析报告”）为客户提供有关预期转换范围、结果和计费的详细信息，以确保对实际转化的准确预期。转换有可能导致一些代码行没有被转换，一些代码行被部分转换，一些代码行被完全转换。分析报告会显示每个类别的代码行数。在处理任何程序、宏和副本的转换之前，必须先运行并阅读分析报告。客户在查看分析报告并同意报告的范围、预期结果和预计费用后，即可进一步执行转换。

> **Note**
>
> 执行 AWS Mainframe Modernization 代码转换 **`Convert`** 命令，即表示您确认自己已经运行并阅读了分析报告，并同意预期结果和可计费的代码行数。

转换范围

AWS Mainframe Modernization 代码转换处理配置的 S3 源位置的 scrlib 和 macrolib 目录中可用的所有汇编器、宏和抄写本组件的所有代码行。汇编器程序以及汇编器程序中引用的所有宏和副本都在范围内。未被汇编器程序引用的宏和副本组件将被视为超出范围，而不进行转换。在处理过程中，转换器会执行从整体上考虑每个范围内组件的高级算法。这些组件的所有代码行都参与处理，无论它们是完全转换、部分转换还是未转换。 AWS Mainframe Modernization 代码转换会忽略空行，也不会将它们算作代码行。注释行和包含任何其他文本的行（例如，嵌入到 JCL 中的汇编器的 JCL 语句）会算作计费的代码行。

费用计算

AWS Mainframe Modernization 范围内的组件全部收取代码转换费用。这意味着它会对每个作用域内组件中的每一行代码收费，包括无法转换、部分转换和完全转换的行。 AWS Mainframe Modernization 代码转换将提供给处理的组件的所有代码行相加（包括汇编程序、引用的抄写本和引用的宏），并使用代码行总数进行计费。

> **Note**
>
> 未被汇编器程序引用的副本和宏不被视为在范围内。

例如，假设一个程序有 1000 行代码：

- 700 行已完全转换
- 200 行已部分转换
- 100 行未转换

这 1000 行代码都会被处理，并全部计费。

改进转化

作为客户，如果您寻求更高的代码行转化率或有其他特定要求，则可以联系销售 AWS 代表，以获取其他参与选项，例如校准工作或专业服务帮助。

代码转换功能概念

要了解代码转换是如何发生的，了解一些关键概念，例如宏处理、代码页和 CodeBuild 非常重要。

主题

宏处理

大型机汇编器代码经常使用宏来封装功能以供重复使用。宏行为通常是在应用程序运行时环境中根据从汇编器程序传递的参数确定的。代码转换功能提供了几种机制用于在转换为 COBOL 之前扩展汇编器宏。

代码页（EBCDIC 与 ASCII）

大型机汇编器通常包含以与 EBCDIC 字符相对应的十六进制值表示的字符字面量。代码转换功能提供了一种可配置的功能，可以在为 ASCII 环境发出 COBOL 时自动管理 ASCII 中的字符字面量。

CodeBuild

可通过该 AWS CodeBuild 服务进行代码转换。 AWS CodeBuild 是一款构建自动化工具，最初是作为 CI/CD 管道的一部分而设计的。中 AWS Mainframe Modernization， AWS CodeBuild 用于自动执行 MCCAC 转换工具和其他工具，例如 Rocket Software（前身为 Micro Focus）COBOL 编译器。

了解代码转换功能的组件和过程

AWS Mainframe Modernization 代码转换过程包括各种组件，例如 AWS Mainframe Modernization 容器、S3 项目存储桶和日志文件位置。

主题

- [AWS Mainframe Modernization 容器](#)
- [S3 项目存储桶](#)
- [日志文件位置](#)
- [过程概述](#)

AWS Mainframe Modernization 容器

AWS Mainframe Modernization 代码转换容器在 AWS CodeBuild 项目中运行，并提供用于设置项目目录和配置文件、评估汇编器代码、展开汇编器宏以及将汇编器代码转换为 COBOL 的命令。

您将有权访问以下 AWS ECR 存储库：`381492161314.dkr.ecr.us-east-1.amazonaws.com/aws-mlogica-codebuild-prod`。

您可以通过以下两种方式之一来使用这些映像：

- 通过使用图像时，请使用最新的标签 AWS CodeBuild。使用映像时，您将使用以下路径：`381492161314.dkr.ecr.us-east-1.amazonaws.com/aws-mlogica-codebuild-prod`。这意味着 AWS CodeBuild 它将把最后一次推送到存储库中的任何一个镜像。
- 列出版本并从中进行选择。为此，请在 CLI 中使用以下命令列出存储库中的不同版本：

```
aws ecr describe-images \
  --registry-id 381492161314 \
  --repository-name aws-mlogica-codebuild-prod \
  --query 'imageDetails[*].{ImagePushedAt: imagePushedAt, ImageTags: imageTags}' \
  --output json | jq '[.[] | {ImageURI: (.ImageTags[] |
 "381492161314.dkr.ecr.us-east-1.amazonaws.com/aws-mlogica-codebuild-prod:" + .),
 ImagePushedAt: .ImagePushedAt}] | sort_by(.ImagePushedAt) | reverse'
```

  这会列出所有映像（每个映像都带有关联的标签），以及特定映像发布到存储库的时间。根据上面的代码，您将获得一个映像列表，映像上的标签代表代码转换实用程序的版本。您可以根据自己的需求选择合适的映像。

S3 项目存储桶

输入和输出代码、使用扩展宏更新的代码以及由 AWS Mainframe Modernization 代码转换生成的报告存储在您在中创建的项目存储桶中。 AWS 账户管理通过向 AWS 服务角色授予权限，您可以为 AWS Mainframe Modernization 代码转换提供对存储桶的访问权限。

日志文件位置

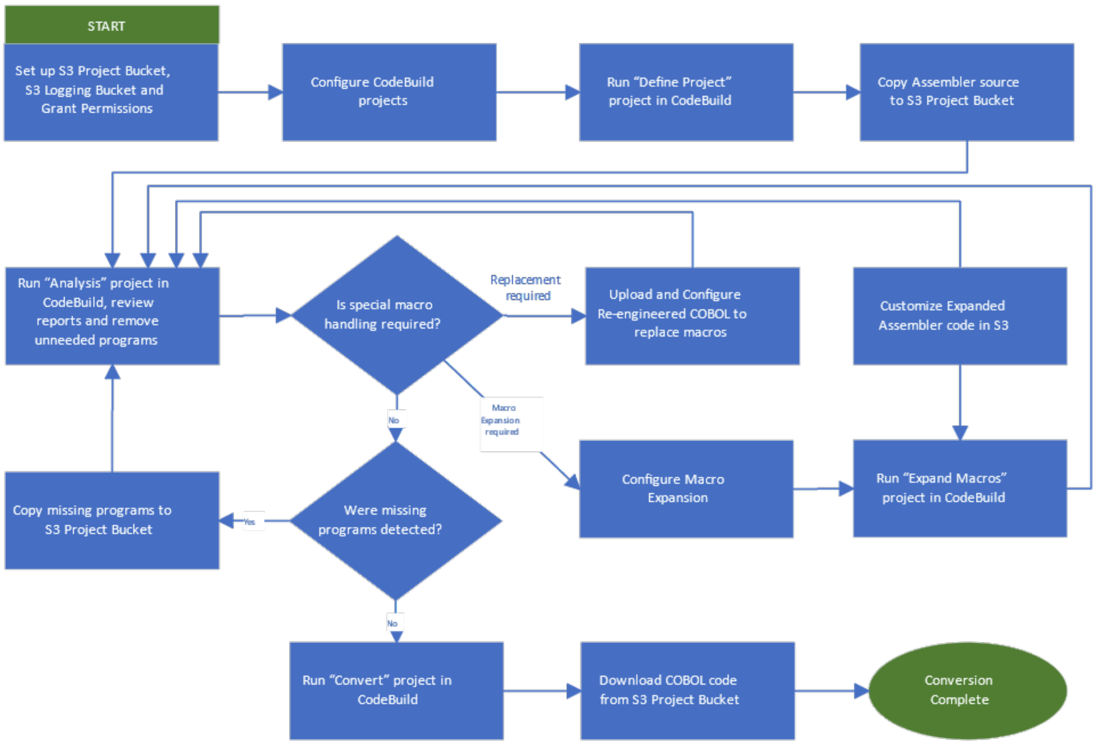
在每次执行 CodeBuild 项目期间，日志文件都写在两个位置：

- 包含每个 CodeBuild 步骤高级结果的日志文件将写入中配置的 Logging 存储桶中的日志文件中 CodeBuild。这些文件以 gzip 存档的形式出现，其文件名由 CodeBuild 框架生成 GUID 类型（例如）。`0c03e183-ab40-4fe0-ba77- bc1d87e73b14.gz`每个档案都包含执行 CodeBuild 项目时生成的日志。如果 CodeBuild项目执行失败，则此日志文件将包含重要的故障排除信息。
- 包含组件级详细执行结果的日志文件将写入到项目的主存储桶路径中，并采用文件名模式 `<Project_Bucket_name>_.log`（例如 `project- bucket_202406131200.log`）。这些日志会提供：
  - 记录输入和输出位置的配置摘要。
  - 根据目标文件名处理的每个汇编器或宏组件的日志。
  - 根据文件位置生成的报告列表。

- 对于转换执行，会提供运行时副本列表。

过程概述

下图说明了将汇编器转换为 COBOL 的过程：



教程：将代码从汇编器转换为 COBOL AWS Mainframe Modernization

您可以使用本文档作为 step-by-step指南，了解如何将大型机现代化汇编程序代码转换为 COBOL。除此之外，您还可以参阅从汇编器到 COBOL 的自动代码转换讲习会，了解有关该转换过程的更多信息。

主题

- 先决条件
- 步骤 1：与共享构建资产 AWS 账户
- 步骤 2：创建 Amazon S3 存储桶
- 步骤 3：创建 IAM 策略

先决条件

阅读本了解汇编器转换的代码转换功能计费节，了解 Assembler 代码转换如何对您产生费用（账单报告）AWS 账户管理，以及计费的工作方式。

步骤 1：与共享构建资产 AWS 账户

在此步骤中，请确保与您共享构建资产 AWS 账户，尤其是在使用资产的区域。

1. 打开 AWS Mainframe Modernization 控制台，网址为https://console.aws.amazon.com/m2/。
2. 在左侧导航中，选择工具。
3. 在使用 mLogica 进行的 AWS 大型机现代化代码转换中，选择与我共享资产。 AWS 账户

> **Important**
>
> 您需要在要进行构建的每个 AWS 区域执行一次此步骤。

步骤 2：创建 Amazon S3 存储桶

在此步骤中，您将创建 Amazon S3 存储桶。第一个存储桶是供 AWS CodeBuild 保存源代码的项目存储桶，然后推送输出存储桶以保存 AWS CodeBuild 输出（转换后的代码）。有关更多信息，请参阅《Amazon S3 用户指南》中的创建、配置和使用 Amazon S3 存储桶。

1. 要创建项目存储桶，请登录 Amazon S3 控制台，然后选择创建存储桶。
2. 在常规配置中，提供存储桶的名称并指定要 AWS 区域 在哪里创建存储桶。一个示例名称为 *codebuild-regionId-accountId-bucket*，其中：
   - `regionId`是最 AWS 区域 重要的。
   - `accountId`是你的 AWS 账户 身份证。

   > **Note**
   >
   > 如果您要在与美国东部（弗吉尼亚北部）不同的 AWS 区域 位置创建存储桶，请指定`LocationConstraint`参数。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的创建存储桶。
3. 保留所有其他设置，并选择创建存储桶。

无论您为这些存储桶选择的名称如何，请务必在本教程中保持统一。

步骤 3：创建 IAM 策略

在此步骤中，您将创建一个 IAM 策略。所提供的 IAM 策略授予与 Amazon S3、Amazon 弹性容器注册表、 CodeBuild 生成的亚马逊 CloudWatch 日志以及用于代码转换的 Amazon Elastic Compute Cloud 资源进行交互的特定权限 AWS CodeBuild 。此策略不是为客户自定义的。该政策授予交互权限，并获取代码转化统计数据以向客户开具相应账单的权限。 AWS Mainframe Modernization

要了解如何创建 IAM 策略，请参阅《IAM 用户指南》中的创建 IAM 策略。

创建策略

1. 登录 IAM 控制台，然后在左侧导航窗格中选择策略。
2. 选择创建策略。
3. 将下面的 JSON 策略复制并粘贴到策略编辑器中。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:GetBucketLocation",
                "s3:ListBucket",
                "s3:PutObjectAcl",
                "s3:GetBucketAcl"
            ],
            "Resource": [
                "arn:aws:s3:::codebuild-regionId-accountId-bucket",
                "arn:aws:s3:::codebuild-regionId-accountId-bucket/*",
                "arn:aws:s3:::aws-m2-repo-*" ],
                    "Effect": "Allow"
          },
          {
              "Action": [
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer",
                "logs:*",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DeleteNetworkInterface",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeDhcpOptions",
                "ec2:DescribeVpcs",
                "ec2:CreateNetworkInterfacePermission"
            ],
            "Resource": "*",
            "Effect": "Allow"
         }
    ]
}
```

4. （可选）您可以向策略添加标签。标签是一些键值对，可以帮助您组织、跟踪或控制对策略的访问。
5. 选择下一步: 审核。
6. 提供策略名称，例如 *CodeBuildAWSM2CCMPolicy*。
7. （可选）您可以输入策略的描述，并检查策略摘要以确保其正确。
8. 选择创建策略。

步骤 4：创建 IAM 角色

在此步骤中，在 CodeBuild 将之前创建的 IAM 策略与该新 IAM 角色关联后，您将创建一个允许您与 AWS 资源交互的新 IAM 角色。

有关创建服务角色的信息，请参阅 IAM 用户指南中的创建角色以向 AWS 服务委派权限。

1. 登录 IAM 控制台，然后在左侧导航窗格中选择角色。
2. 选择创建角色。
3. 在可信实体类型下，选择 AWS 服务。
4. 在其他 AWS 服务的使用案例下 CodeBuild，选择，然后CodeBuild再次选择。
5. 选择下一步。
6. 在添加权限页面上，选择下一步。您稍后给该角色分配策略。
7. 在角色详细信息下，提供角色的名称，例如 `IAMRoleTaskExecutionRoleForCodeBuild`。
8. 在选择可信实体下，验证策略文档是否如下所示：

   JSON

```
{
            "Version":"2012-10-17",
            "Statement": [
              {
                "Effect": "Allow",
                "Principal": {
                  "Service": "codebuild.amazonaws.com"
                },
                "Action": "sts:AssumeRole"
              }
```

```
            ]
}
```

9. 选择创建角色。

步骤 5：将 IAM 策略附加到 IAM 角色

在此步骤中，您需要将之前创建的 IAM 策略附加到 `IAMRoleTaskExecutionRoleForCodeBuild` IAM 角色。

1. 登录 IAM 控制台，然后在左侧导航窗格中选择角色。
2. 在角色中，选择您之前创建的角色，例如 `IAMRoleTaskExecutionRoleForCodeBuild`。
3. 在权限策略中，依次选择添加权限和附加策略。
4. 在其他权限策略中，选择您之前创建的策略，例如 *`CodeBuildAWSM2CCMPolicy`*。
5. 选择附加策略。

步骤 6：创建 CodeBuild 项目

在此步骤中，您将基于上述`buildspec.yml`文件创建三个不同的 CodeBuild 项目。

步骤 6.1：创建定义项目

创建定义项目

1. 登录 CodeBuild 控制台，然后选择 “创建构建项目”。
2. 在项目配置部分中，为项目提供一个名称，例如 `1-awsm2ccm-define-project`。
3. 在源部分中，对于源提供商，保留默认选择。
4. 在环境部分中，选择自定义映像。
5. 在环境类型字段中，选择 Linux。
6. 在映像注册表中，选择其他注册表。
7. 在外部注册表 URL 字段中，按照 the section called “AWS Mainframe Modernization 容器”部分中的说明进行操作。
8. 在服务角色下，选择现有服务角色，然后在角色 ARN 字段中，选择您之前创建的服务角色（例如 `IAMRoleTaskExecutionRoleForCodeBuild`）。

9. 展开其他配置部分，进行如下设置：

   a. VPC：请根据您的设置需要进行配置。

   b. 超时：设置为 60 分钟。

   c. 排队超时：设置为 480 分钟。

   d. 加密：选择适当的加密设置（默认设置即可）。

   e. 在环境变量部分中，逐一添加以下内容：

      - 名称：PROJECT_BUCKET。值：**codebuild-regionId-accountId- bucket**。类型：Plaintext
      - 名称：PROJECT_DIR。值：**prj_codebuild_01**。类型：Plaintext
      - 名称：AWSM2CCM_ ACTION。值：**define_project**。类型：Plaintext
      - 名称：AWSM2CCM_LOGGING_BUCKET。值：**s3:// codebuild-regionId-accountId-bucket**。类型：Plaintext

10. 在 Buildspec 部分中，选择插入构建命令，然后选择切换到编辑器。

11. 将当前值替换为：

```
version: 0.2
phases:
    build:
        commands:
            - . /app/awsm2ccm_prod/bin/setup_env.sh
            - run_awsm2ccm.sh $PROJECT_DIR
artifacts:
    files:
        - '**/*'
    discard-paths: no
    base-directory: $PROJECT_DIR
```

    其中，PROJECT_DIR 是其中可用的环境变量。 CodeBuild有关更多信息，请参阅[构建环境中的环境变量](#)。

12. 在构件部分中，执行以下操作：

    - 在类型下，选择 Amazon S3，然后选择您的输出存储桶，例如 `codebuild-regionId-accountId-bucket`。
    - 对于路径，请将此字段留空。
    - 对于名称，请输入 **prj_codebuild_01**。

- 对于构件打包，请选择无。
- 对于覆盖构件名称，请取消选中此选项。
- 对于加密，请保留默认设置。

13. 对于日志部分，请按如下方式设置：

    - CloudWatch 日志：已禁用
    - S3 日志：已启用

    - 存储桶：**`codebuild-regionId-account-bucket`**
    - 日志路径：**`CODEBUILD-LOGS`**

14. 选择 Create build project（创建构建项目）。

步骤 6.2：创建代码分析项目

创建代码分析项目

1. 登录 CodeBuild 控制台，然后选择 “创建构建项目”。
2. 在项目配置部分中，为项目提供一个名称，例如 `2-awsm2ccm-analysis`。
3. 在源部分中，对于源提供商，请选择 Amazon S3，然后选择您之前创建的输入存储桶（例如 `codebuild-regionId-accountId-bucket`）。
4. 在 S3 对象键或 S3 文件夹字段中，输入 **`prj_codebuild_01`**。
5. 在环境部分中，选择自定义映像。
6. 在环境类型字段中，选择 Linux。
7. 在映像注册表中，选择其他注册表。
8. 在外部注册表 URL 字段中，按照 [the section called “AWS Mainframe Modernization 容器”]部分中的说明进行操作。
9. 在服务角色下，选择现有服务角色，然后在角色 ARN 字段中，选择您之前创建的服务角色（例如 `IAMRoleTaskExecutionRoleForCodeBuild`）。
10. 展开其他配置部分，进行如下设置：

    a. VPC：请根据您的设置需要进行配置。
    b. 超时：设置为 60 分钟。
    c. 排队超时：设置为 480 分钟。

d. 加密：选择适当的加密设置（默认设置即可）。

e. 在环境变量部分中，逐一添加以下内容：

- 名称：PROJECT_BUCKET。值：**codebuild-regionId-accountId-bucket**。类型：Plaintext
- 名称：PROJECT_DIR。值：**prj_codebuild_01**。类型：Plaintext
- 名称：AWSM2CCM_ ACTION。值：**analysis**。类型：Plaintext
- 名称：AWSM2CCM_LOGGING_BUCKET。值：**s3:// codebuild-regionId-accountId-bucket**。类型：Plaintext

11. 在 Buildspec 部分中，选择插入构建命令，然后选择切换到编辑器。

12. 将当前值替换为：

```
version: 0.2
phases:
    build:
        commands:
            - ln -s $CODEBUILD_SRC_DIR $PROJECT_DIR
            - . /app/awsm2ccm_prod/bin/setup_env.sh
            - run_awsm2ccm.sh $PROJECT_DIR
artifacts:
    files:
        - '*.log'
        - '_Converted/*/*'
        - '_Reports/*'
    secondary-artifacts:
        reports:
            files:
                - '_Reports/AWSM2CCM*'
    discard-paths: no
    base-directory: $PROJECT_DIR
```

其中，PROJECT_DIR 是其中可用的环境变量。 CodeBuild有关更多信息，请参阅构建环境中的环境变量。

13. 在构件部分中，执行以下操作：

- 在类型下，选择 Amazon S3，然后选择您的输出存储桶（例如 `codebuild-regionId-accountId-bucket`）。
- 对于路径，请输入 ARTIFACTS。

- 对于名称，请输入 **prj_codebuild_01**。
- 对于构件打包，请选择无。
- 对于覆盖构件名称，请取消选中此选项。
- 对于加密，请保留默认设置。

14. 对于日志部分，请按如下方式设置：

    - CloudWatch 日志：已禁用
    - S3 日志：已启用

    - 存储桶：**codebuild-regionId-account-bucket**
    - 日志路径：**CODEBUILD-LOGS**

15. 选择 Create build project（创建构建项目）。

步骤 6.3：创建代码转换项目

创建代码转换项目

1. 登录 CodeBuild 控制台，然后选择 “创建构建项目”。
2. 在项目配置部分中，为项目提供一个名称（例如 `3-awsm2ccm-convert`）。
3. 在源部分中，对于源提供商，请选择 Amazon S3，然后选择您之前创建的输入存储桶（例如 `codebuild-regionId-accountId-bucket`）。
4. 在 S3 对象键或 S3 文件夹字段中，输入 **prj_codebuild_01**。
5. 在环境部分中，选择自定义映像。
6. 在环境类型字段中，选择 Linux。
7. 在映像注册表中，选择其他注册表。
8. 在外部注册表 URL 字段中，按照 the section called “AWS Mainframe Modernization 容器”部分中的说明进行操作。
9. 在服务角色下，选择现有服务角色，然后在角色 ARN 字段中，选择您之前创建的服务角色，例如 `IAMRoleTaskExecutionRoleForCodeBuild`。
10. 展开其他配置部分，进行如下设置：

    a. VPC：请根据您的设置需要进行配置。

    b. 超时：设置为 60 分钟。

c. 排队超时：设置为 480 分钟。

d. 加密：选择适当的加密设置（默认设置即可）。

e. 在环境变量部分中，逐一添加以下内容：

- 名称：PROJECT_BUCKET。值：**codebuild-regionId-accountId-bucket**。类型：Plaintext
- 名称：PROJECT_DIR。值：**prj_codebuild_01**。类型：Plaintext
- 名称：AWSM2CCM_ ACTION。值：**conversion**。类型：Plaintext
- 名称：AWSM2CCM_LOGGING_BUCKET。值：**s3:// codebuild-regionId-accountId-bucket**。类型：Plaintext

11. 在 Buildspec 部分中，选择插入构建命令，然后选择切换到编辑器。

12. 将当前值替换为：

```
version: 0.2
phases:
    build:
        commands:
            - export AWSM2CCM_PUSH_RUNTIME_COPYBOOKS=y
            - ln -s $CODEBUILD_SRC_DIR $PROJECT_DIR
            - . /app/awsm2ccm_prod/bin/setup_env.sh
            - run_awsm2ccm.sh $PROJECT_DIR
artifacts:
    files:
        - '*.log'
        - '_Converted/*/*'
        - '_Reports/*'
    discard-paths: no
    base-directory: $PROJECT_DIR
```

其中，PROJECT_DIR 是其中可用的环境变量。 CodeBuild有关更多信息，请参阅构建环境中的环境变量。

13. 在构件部分中，执行以下操作：

- 在类型下，选择 Amazon S3，然后选择您的输出存储桶（例如 codebuild-regionId-accountId-bucket）。
- 对于路径，请输入 ARTIFACTS。
- 对于名称，请输入 **prj_codebuild_01**。

- 对于构件打包，请选择无。
- 对于覆盖构件名称，请取消选中此选项。
- 对于加密，请保留默认设置。

14. 对于日志部分，请按如下方式设置：

    - CloudWatch 日志：已禁用
    - S3 日志：已启用

    - 存储桶：**codebuild-regionId-account-bucket**
    - 日志路径：**CODEBUILD-LOGS**

15. 选择 Create build project（创建构建项目）。

步骤 7：定义项目并上传源代码

定义项目会设置项目文件夹和配置文件，并使用默认配置进行初始化。在此步骤中，启动构建。要实现此目的，应按照以下步骤进行：

1. 登录 AWS CodeBuild 控制台。
2. 在左侧导航窗格中，选择构建项目。
3. 选择先前创建的项目（`1-awsm2ccm-define-project`）来进行构建
4. 选择开始构建，然后选择立即开始来定义项目。构建开始后，状态将变为进行中。
5. 选择 “阶段详细信息” 以查看由项目精心策划的每个步骤的进度。 AWS CodeBuild
6. 等待所有步骤的状态均变为已成功。
7. 前往 Amazon S3 控制台。
8. 找到并单击名为 `codebuild-regionId-accountId-bucket` 的 Amazon S3 存储桶

    - **CODEBUILD-LOGS/**文件夹包含正在运行的 AWS CodeBuild 项目的 AWS CodeBuild 日志。
    - **prj_codebuild_01/** 文件夹包含项目结构。该文件夹在分析、扩展宏和转换步骤中使用。您可以选择 `prj_codebuild_01/` 来了解详细信息
    - **cobol_reserved.rsw** 是为转换器保留的配置文件（COBOL 字列表）。该文件在转换步骤中使用。
    - **Macro_Expansion/** 文件夹包含要扩展到汇编器程序中的宏。该文件夹在扩展宏步骤中使用。
    - **macro_settings.json** 配置文件包含自定义宏替换。该文件在扩展宏步骤中使用。

- **macrolib/** 文件夹包含要转换的汇编器宏。该文件夹在分析和转换步骤中使用。
  1. 选择 `macrolib/`。
  2. 默认会提供一个名为 `MACRO1.mac` 的汇编器宏作为示例文件。请删除此文件，因为分析不需要该文件。
  3. 将您的宏上传到此目录中。
- **project_settings_aux.json** 配置文件包含与代码页相关的设置。该文件在转换步骤中使用。
- **project_settings.json** 配置文件包含转换器的设置。该文件在转换步骤中使用。
- **srclib/** 文件夹包含要转换的汇编器程序。该文件夹在分析和转换步骤中使用。
  1. 选择 `srclib/`。
  2. 默认会提供名为 `SQtest01.asm` 和 `SQtest02.asm` 的两个汇编器程序作为示例。请删除这两个文件，因为分析和转换不需要它们。
  3. 将您的汇编器程序上传到此目录中。

9. 确认 `1-awsm2ccm-define-project` 步骤的状态。此步骤应该在最新构建状态选项卡下显示为“已成功”。

您已准备好进行下一步操作：代码分析。

步骤 8：运行分析并理解报告

> **Note**
>
> AWS Mainframe Modernization 代码转换分析步骤是免费的。

在此步骤中，您将启动另一个构建：

1. 在左侧导航窗格中，选择构建项目。
2. 选择您在步骤 6.2 中创建的项目来进行构建：`2-awsm2ccm-analysis`。
3. 选择开始构建，然后选择立即开始以生成分析报告。此时将开始进行构建，状态将变为进行中。
4. 选择阶段详细信息，您将在其中看到项目精心策划的每个步骤的进度。 AWS CodeBuild 等到所有步骤的状态均变为已成功。
5. 从中 AWS 管理控制台，转到 Amazon S3 服务控制台。
6. 找到并单击 Amazon S3 存储桶：`codebuild-regionId-accountId-bucket`

a. **ARTIFACTS/** 文件夹包含分析和转换步骤的输出。

b. 选择 ARTIFACTS/prj_codebuild_01/_Reports/。

c. 将提供以下报告：

- AWSM2CCM-Analysis-Report-<timestamp>.pdf是一份执行报告，提供了 AWS Mainframe Modernization 代码转换计费和范围，改进了转换、转换摘要和详细的转化统计信息。该报告还汇总了项目级别的代码数量和可计费代码数，并提供了每个组件的指标和引用成员列表。在进行实际转换之前，请务必运行并检查此报告。
- Conversion_Detailed_Statistics.txt 提供了每个组件中的每条指令的频率和预期转换结果（显示为“转换状态”）。该报告提供了一种方法来快速识别转换器是否支持指令。可能的转换状态结果有：
  - 完全转换：指令将被准确地转换为 COBOL。
  - 部分转换：支持该指令，但使用了不支持的参数或表达式。转换后可能需要手动调整。
  - 未转换：转换器不支持该指令。
  - 预编译指令以进行验证：这些指令通常包含在宏中，指的是在大型机上可能也称为条件汇编语言（例如 AIF、AGO）指令的指令。这些指令由预编译器处理，预编译器由此类指令驱动，会选择并生成干净/静态的 ASM 代码。这些指令依赖于所编译的宏参数的实际值。因此，同一个宏可以根据所传递的参数值生成不同的 ASM 代码片段。之所以如此，就是因为存在这样的预编译指令。在这种情况下，可以考虑扩展或重新设计宏。
- Conversion_Global_Statistics.txt 提供组件级别的转换状态摘要。
- CrossReference_PgmToCpyMacro.txt 报告汇编器程序对宏的依赖。该报告提供了一种方法来快速确定上传的代码中是否缺少任何宏。
- CrossReference_PgmToPgm.txt 报告汇编器程序对其他汇编器程序的依赖。该报告提供了一种方法来快速确定上传的代码中是否缺少任何汇编器程序。

7. 返回 AWS CodeBuild 服务控制台。
8. 确认 2-awsm2ccm-analysis 步骤的状态。此步骤应该在最新构建状态选项卡下显示为已成功。

您已准备好进行下一步操作：代码转换。

步骤 9：运行代码转换

> **Important**
>
> AWS Mainframe Modernization 代码转换转换步骤将按您的使用量计费。有关定价的更多信息，请参阅 the section called “了解代码转换功能计费”。

在此步骤中，您将配置转换过程，然后开始构建。

1. 从中 AWS 管理控制台，转到 Amazon S3 服务。
2. 找到并单击 Amazon S3 存储桶：`codebuild-regionId-accountId-bucket`。
   a. 转到 `prj_codebuild_01/`。
   b. 选择 `project_settings.json`，然后选择下载。
   c. 打开 `project_settings.json` 文件，您会看到以下 JSON 结构：

```
{
"Source programs directory":"srclib",
"Source copybooks/macros directory":"macrolib",
"Copybook/Macros Conversion":"Called_only",
"Do not regenerate the Copy/Macro if already exists":"false",
"Target Compiler":"IBM",
"Endianess":"Big",
"Converted programs extension":"",
"Converted CICS programs extension":"",
"Converted copies/macros extension":"",
"Trace Level":"STANDARD",
"Trace file open mode":"append",
"Data definition level":5,
"Start picture column":40,
"Generate Sync FILLER with name":"FILL-SYNC",
"Use SYNC clause":"yes",
"Decimal Point Comma":"true",
"Original Source Placement":"RIGHT"
}
```

其中，

- Source program directory：包含转换所需的汇编器程序。

- Source copybooks/macros directory：包含转换所需的汇编器宏和副本。
- Copybooks/Macros Conversion 可为以下二者之一：
    - 全部：此单选按钮表示完全转换将转换目录中所有 copybook/Macros 可用的内容，无论程序是否正在使用该目录。
    - called_ only：此单选按钮表示完全转换只会转换程序实际 copybook/Macros 使用的转换。
- > **Important**
  >
  > Copy/Macro 如果它已经存在，则无需重新生成。

  对于已经转换（存在于输出文件夹中）的副本/宏，该工具将不会再次转换它。
- Target：程序的转换结果（生成的代码）取决于目标 COBOL 编译器。支持以下选项：
    - “IBM”，适用于 IBM 大型机
    - “MF”，适用于 Micro Focus COBOL
    - “VERYANT”，适用于 Veryant isCOBOL
    - “NTT”，适用于 NTT DATA Enterprise COBOL（Unikix）
- Endianess 和 Bitness：程序的转换结果（生成的代码）取决于目标平台（Bitness/Endianess）。此组合支持选择以下受支持的选项：
    - Endianess：Big（对于 Big-Endian）/Little（对于 Little-Endian）。例如，IBM z/OS 的主机是 Big-Endian，Windows 是 Little-Endian，Linux 因发行版而异（例如 Amazon Linux 2 上是 Little-Endian）。 EC2
    - Bitness：32/64（如果未指定，默认为 32）。建议设置为 32 位。
- Converted program extension：这是为了设置所生成的 COBOL 程序的文件扩展名。空（“”）：没有扩展名。对于 Rocket Software（前身为 Micro Focus）CO BOL 目标，建议 CBL 让 Rocket Enterprise Develo
- Converted CICS program extension：这是为了设置所生成的 CICS COBOL 程序的文件扩展名。空（“”）：没有扩展名。对于 Rocket Software CO BOL 目标，建议使用 CBL 让 Rocket Enterprise 开发人员能够正确识别文件。
- Converted copybooks/macros extension：这是为了设置所生成的 COBOL 副本的文件扩展名。空（“”）：没有扩展名。对于 Rocket Software COBOL 目标，建议使用 CPY 来让 Rocket Enterprise Developer

- 跟踪级别：跟踪是转换 CodeBuild 过程中使用的记录信息。用户可以通过选择所提供的任何一个选项来选择详细级别。
  - ERROR = TRACE ERROR：仅显示转换错误。
  - STANDARD = TRACE STANDARD：显示转换错误和标准信息。这是推荐设置。
  - ALL = TRACE ALL：最大跟踪级别
- Trace file open mode：未使用。建议使用默认设置 append。
- Data definition level：这表示在工作存储和链接部分中定义的子字段的初始级别（在级别“01”之后）。必须是一个数字。
- Start picture column：这与生成的 COBOL 代码的格式有关，表示放置 PIC 子句的列（在字段名称之后）。必须是一个数字。
- Original source placement：这表示注释在程序中的放置位置。有两个选项：
  - RIGHT：此选项会将注释或附加信息放在第七十三（73）列之后的右侧位置。在 COBOL 中，代码写在前七十二（1-72）列中，从第七十三列开始（>= 73 列）的任何内容都将被视为注释。
  - ABOVE：此选项会将注释放在已转换内容的上方。
- Generate Sync FILLER with name：此选项与二进制字段（汇编器“H”、“F”、“D”数据类型，这些数据类型会转换为 COBOL“COMP”数据类型）在内存中的对齐有关。为了保证正确的对齐边界，将在转换过程中添加显式填充字段。这是一个基于文本的选项，其值必须是字符串（如 FILL-SYNC）。
- Use SYNC clause：此选项与二进制字段在内存中的对齐有关。Yes = 将所有字段转换为 COBOL。“COMP”将用“SYNC”子句定义（例如，05 WRKFLD PIC S9(09) COMP SYNC）。
- Decimal Point Comma：为 true 时，会将 DECIMAL-POINT IS COMMA 子句添加到“SPECIAL-NAMES”COBOL 段落中。

d. 根据您的需求更改相应的参数，然后保存 `project_settings.json`。

e. 从 Amazon S3 存储桶中的 `prj_codebuild_01/` 中删除现有的 `project_settings.json` 文件，然后上传新版本。

3. 返回 AWS CodeBuild 服务。

4. 选择先前创建的项目（`3-awsm2ccm-convert`）来进行构建

   a. 选择开始构建，然后选择立即开始，将汇编器程序和宏转换为 COBOL 程序和副本。

   b. 等待此项目的构建状态变为已成功。该状态会显示在最新构建状态选项卡下。

步骤 10：验证代码转换

1. 从中 AWS 管理控制台，转到 Amazon S3 服务。
2. 找到并单击 Amazon S3 存储桶：`codebuild-regionId-accountId-bucket`。
3. 导航到**`awsm2ccm-do-not-delete`** 。 AWS Mainframe Modernization 在转换过程中，代码转换会为每个汇编器或宏模块创建编码的二进制文件。这些文件对于防止向客户重复计费以及跟踪所提供的汇编器代码被分析和转换了多少至关重要。这些文件存放在以下位置：`codebuild-regionId-accountId- bucket/awsm2ccm-do-not-delete/<your_AWS_account_id>/Hash`。这些编码的文件不包含任何汇编器代码，也不能从中提取客户代码。

   > Important
   >
   > 请勿手动编辑或删除这些文件。编辑或删除这些文件可能会导致对相同组件进行重复计费。

   请将 **`awsm2ccm-do-not-delete/`** 文件夹视为系统管理的目录。对此目录或其内容进行任何更改 支持 之前，请先进行咨询。
4. 单击 `codebuild-regionId-accountId-bucket` 返回存储桶。
5. 选择 **`ARTIFACTS/prj_codebuild_01/`**。_Converted/ 文件夹包含作为代码转换步骤的结果生成的 COBOL 输出。该文件夹将包含以下子目录：
   - copybooks/ 文件夹，包含生成的 COBOL 副本。
   - programs/ 文件夹，包含生成的 COBOL 程序。
   - runtime_lib/ 文件夹，包含解决方案提供的其他 COBOL 程序和副本。
6. 如果分析报告和其他报告表明转换成功，并且 AWS CodeBuild 项目标记为成功，请从 _ Conver `3-awsm2ccm-convert` ted/ 目录中下载 COBOL 代码和抄本。

步骤 11：下载转换后的代码

在此步骤中，从 _Converted/ 目录中下载 COBOL 代码和副本，然后在目标 COBOL 环境中对其进行编译。

1. 从中 AWS 管理控制台，转到 Amazon S3 服务。
2. 找到并单击 Amazon S3 存储桶：`codebuild-regionId-accountId-bucket`。

3. 导航到以下位置： `ARTIFACTS/prj_codebuild_01/_Converted/`。
4. 从 _Converted/ 下的所有子目录中下载转换后的 COBOL 代码。您也可以使用以下 CLI 命令一次性下载它们：

```
aws s3 cp s3://codebuild-regionId-accountId-
bucket/ARTIFACTS/prj_codebuild_01/_Converted/ . --recursive
```

5. 在目标 COBOL 环境中分析和编译转换后的 COBOL。

清理 资源

如果您不再需要为本教程创建的资源，请将其删除，以免产生额外费用。为此，请完成以下步骤：

- 删除您为本教程创建的 S3 存储桶。有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的删除存储桶。
- 删除您为本教程创建的 IAM 策略。有关更多信息，请参阅《IAM 用户指南》中的删除 IAM 策略。
- 删除您为本教程创建的 IAM 角色。有关更多信息，请参阅《IAM 用户指南》中的删除角色或实例配置文件。
- 删除您为本教程创建的 CodeBuild 项目。有关更多信息，请参阅《AWS CodeBuild 用户指南》CodeBuild中的删除构建项目。

Charon 集成

Charon-SSP 简介

1987 年，Sun Microsystems 发布了 SPARC V7 处理器，这是一款 32 位 RISC 处理器。随后，在 1990 年，又发布了 SPARC V8，对最初的 SPARC V7 进行了修订，其中最引人注目的是包括了硬件除法和乘法指令。SPARC V8 处理器构成了许多服务器和工作站（例如 SPARCstation 5、10 和 20）的基础。继 SPARC V8 之后，1993 年，又推出了 64 位的 SPARC V9 处理器。它也成为了许多服务器和工作站的基础，例如 Enterprise 250 和 450。

由于硬件过时以及缺少备件或翻新部件，为这些基于 SPARC 的旧工作站和服务器开发的软件和系统变得更加难以维护。为了满足对某些 end-of-life基于Sparc的系统的持续需求，Stromasys S.A. 开发了 Charon-SSP系列的SPARC仿真器产品。以下产品是指定本机硬件 SPARC 系统的基于软件的虚拟机替代品。以下是仿真硬件系列的总体概述。

Charon-SSP/4M 仿真以下 SPARC 硬件：

- Sun-4m 系列（由 Sun SPARCstation 20 代表）：最初是多处理器 Sun-4 变体，基于 600MP 系列中推出的 MBus 处理器模块总线。 SPARCServer 后来，Sun-4m架构还包括非MBus 单处理器系统，例如使用SPARC v8架构处理器的SPAR Cstation 5。从 SunOS 4.1.2 开始支持，Solaris 2.1 到 Solaris 9 都支持。 SPARCServer 在 Solaris 2.5.1 之后，6亿万像素的支持被放弃。

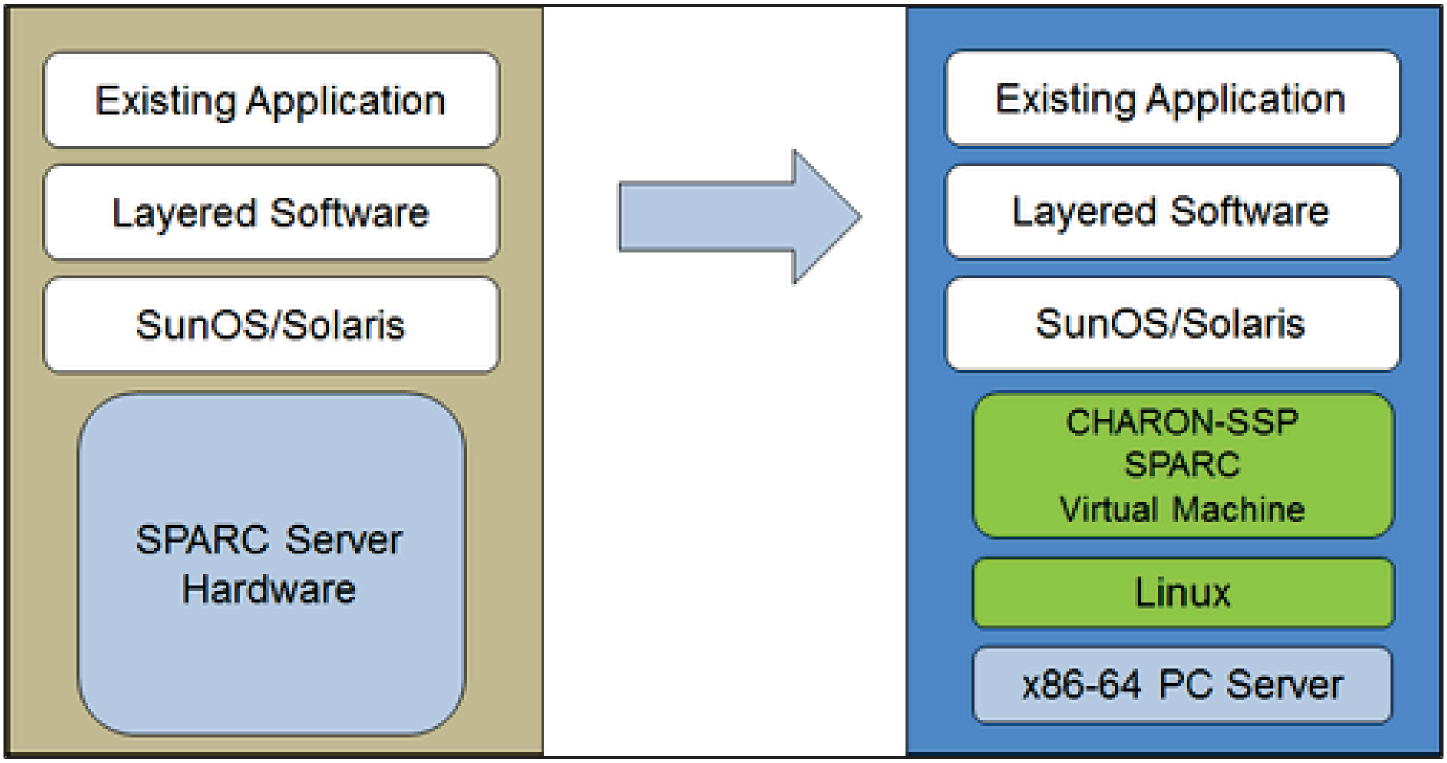
Charon-SSP/4U(+) 仿真以下 SPARC 硬件：

- Sun-4u 系列（代表产品为 Sun Enterprise 450）：（U 代表 UltraSPARC）- 该变体引入了 Sun Ultra 系列中首次使用的 64 位 SPARC V9 处理器架构和 UPA 处理器互连。得到自 2.5.1 版本开始的 32 位版本 Solaris 的支持。Sun-4u 的第一个 64 位 Solaris 版本是 Solaris 7。在 Solaris 9 之后，UltraSPARC I 支持被废弃。Solaris 10 支持从 UltraSP ARC II 到 UltraSPARC IV 的 Sun-4U 实现。

Charon-SSP/4V(+) 仿真以下 SPARC 硬件：

- Sun-4v 系列（代表产品为 SPARC T2 和 T4）：此变体在 Sun-4u 中增加了虚拟机管理程序处理器虚拟化；被引入到 Ultra SPARC T1 多核处理器中。从 3/05 版本开始，Solaris 版本 10 支持部分硬件 HW2 （大多数型号，包括由 Charon-SSP 模拟的硬件）都需要较新版本的 Solaris 10）。还支持多个 Solaris 11 版本。

下图显示了将物理硬件迁移到模拟器的基本概念。



Charon-SSP 虚拟机允许使用基于 Sun 和 Oracle SPARC 的计算机的用户以一种简单的方式来更换其本机硬件，这种方式很少需要甚至不需要更改原始系统配置。这意味着您可以继续运行应用程序和数据，而无需切换或移植到其他平台。Charon-SSP 软件在商用 Intel 64 位系统上运行，确保持续保护您的投资。

Charon-SSP/4U+ 支持与 Charon-SSP/4U 相同的虚拟 SPARC 平台，Charon-SSP/4V+ 则支持与 Charon-SSP/4V 相同的平台。但是，4U+和4V+版本利用了现代英特尔的 VTx /EPT和AMD的AMD-V/NPT硬件辅助虚拟化技术来提供更好的虚拟CPU性能。 CPUs Charon-ssp/4u+ 和 Charon-ssp/4v+ 需要 CPUs 支持 vt-x/EPT 或 AMD-v/NPT，并且必须安装在专用主机系统上。不支持在 VM（例如，on VMware）中运行这些产品变体。

> **Note**
>
> 如果您打算在云环境中运行 Charon-SSP/4U+ 或 4V+ ，请联系 Stromasys 或 Stromasys VAR 来讨论您的要求。

受支持的来宾操作系统

Charon-SSP/4M 虚拟机支持以下来宾操作系统版本：

- SunOS 4.1.3 - 4.1.4
- Solaris 2.3 到 Solaris 9

Charon-SSP/4U(+) 虚拟机支持以下来宾操作系统版本：

- Solaris 2.5.1 到 Solaris 10

Charon-SSP/4V(+) 虚拟机支持以下来宾操作系统版本：

- Solaris 10（从 08/07 的更新 4 开始）和 Solaris 11.1 到 Solaris 11.4

对于 Charon-SSP/4V(+)，请注意以下几点：

- 对于仿真的 SPARC T4，受支持的 Solaris 10 版本为：Oracle Solaris 10 1/13、Oracle Solaris 10 8/11 和 Solaris 10 9/10，或者带有 Oracle Solaris 10 8/11 补丁集的 Solaris 10 10/09。
- 仿真的 SPARC T4 型号是在模拟器中运行 Solaris 11.4 的先决条件。
- 不支持 Solaris 内核区域。

Charon-SSP 云实例的先决条件

通过选择实例类型或形态，您可以选择将用于云中 Charon-SSP 主机实例的虚拟硬件。因此，实例类型或形态的选择决定了 Charon-SSP 虚拟主机硬件的硬件特性（例如，您的虚拟 Charon 主机系统将有多少 CPU 内核和多少内存）。

> **Note**
> 如果使用 Charon-SSP 市场映像来启动实例，将满足所有 Linux 主机操作系统要求。

最低硬件要求如下。

有关规格指南的要点：

- 以下的规格指南（特别是有关主机 CPU 内核数和主机内存量）显示了最低要求。必须审查每种部署情况，并根据需要调整实际的主机规格。例如，如果来宾应用程序产生高 I/O 负载，则必须增加可用于 I/O 的 CPU 内核数量。此外，具有许多仿真功能的系统 CPUs 通常能够产生更高的 I/O 负载，因

此可能必须增加可用于 I/O 的 CPU 内核数量。在超线程环境中，为了获得最佳性能，CPU 内核（即真实/物理 CPUs）的数量必须足以满足活动仿真器的 CPU 要求，从而避免高工作负载线程共享一个物理 CPU 内核。

- 模拟的 CPU 内核 CPUs 和用于 I/O 处理的 CPU 内核分配由配置决定。关于这一点以及用于 I/O 处理的 CPU 内核的默认分配的更多信息，请参阅一般性的《Charon-SSP 用户指南》中的 CPU 配置。

⚠ 重要的一般信息

- 为了便于将模拟器数据从一个云实例快速传输到另一个云实例，强烈建议您将所有相关的模拟器数据存储在单独的磁盘卷上，而该磁盘卷可以轻松地与旧实例分离并连接到新实例。
- 确保从一开始就形成正确的实例大小（检查下面的最低要求）。首次启动实例时，将创建 Charon-SSP AL 的 Charon-SSP 许可证。如果稍后改为其他实例大小/类型，进而改变 CPU 内核的数量，就会使许可证失效，阻止 Charon 实例启动（需要新实例）。如果计划在 AutoVE 模式下使用 Charon-SSP AL 实例，请务必在首次启动之前包含 AutoVE 服务器信息，否则将使用公共许可证服务器。Charon-SSP VE 的许可证是根据在许可证服务器上获取的指纹创建的。如果许可证服务器直接在模拟器主机上运行，而模拟器主机后来要求更改 CPU 内核数量，则该许可证将失效（需要新的许可证，还可能需要新的实例）。

实例先决条件

一般 CPU 要求：Charon-SSP 支持基于现代 x86-64 架构处理器的亚马逊实例。 EC2

Charon-SSP 的最低要求：

- 主机系统 CPU 内核的最小数量：
  - 主机操作系统至少有一个 CPU 内核，以及：
  - 对于每个仿真的 SPARC 系统：
    - 实例的每个仿真 CPU 有一个 CPU 内核，以及：
    - 至少增加一个用于 I/O 处理的 CPU 内核（如果使用服务器 JIT 优化，则至少需要两个 CPU 内核）。有关配置选项，请参阅上面提到的 CPU 配置部分。默认情况下，Charon 会将 Charon 主机 CPUs可见数量的 1/3（最小 1；向下舍入）分配给 I/O 处理。
- 最低内存要求：

  - 对于 Linux 主机操作系统，4GB 或更大 RAM。实际要求可能更高，并将取决于在 Linux 主机上运行的非模拟器服务的要求。之前关于 Linux 主机至少具有 2GB RAM 的建议对许多系统仍然有效，但是随着对 Linux 操作系统和应用程序的要求不断提高，对于新安装的建议也更新了。以及：
  - 对于每个仿真的 SPARC 系统：
    - 仿真实例的配置内存，以及：
    - 2GB 内存（如果使用服务器 JIT，则为 6GB 内存），用于实现 DIT 优化、模拟器要求、运行时缓冲区、SMP 和图形仿真。
- 如果在现代 x86-64 上启用了超线程 CPUs，则两个线程可以在一个物理 CPU 内核上运行，从而为主机操作系统提供两个逻辑 CPUs 线程。如果可能，请在 Charon-SSP 主机上禁用超线程。但是，在云环境中 VMware 通常不可能做到这一点，或者尚不清楚是否使用超线程。Charon-SSP 超线程选项使 Charon-SSP 能够适应此类环境。有关详细的配置信息，请参阅上面提到的一般性《Charon-SSP 用户指南》中的 CPU 配置部分。请注意：为了获得最佳性能，Charon-SSP 线程不应共享物理 CPU 内核 - 主机系统上应有足够的物理核心，以便满足已配置模拟器的要求。
- 一个或多个网络接口，视客户要求而定。
- Charon-SSP/4U+ 和 Charon-SSP/4V+ 必须在支持 Intel VT-x/EPT 或 AMD-v/NPT（裸机实例）的物理硬件上运行，因此无法在所有云环境中运行。请查看您的云提供商的文档，了解此类硬件的可用性。此外，请注意以下几点：
  - Charon-SSP/4U+ 和 Charon-SSP/4V+ 只有在使用 Stromasys 支持的 Linux 内核时才可用。
  - 如果您需要这种类型的仿真 SPARC 硬件，请联系 Stromasys 或您的 Stromasys VAR，详细讨论您的需求。

为 Charon 创建和配置 AWS 云实例（新 GUI）

本节反映了202 AWS 管理控制台 2年春季的情况。如果您仍在使用较旧的主机，请参阅 Charon-SSP AWS 入门指南的附录。

一般先决条

此描述显示了 AWS中的 Linux 实例的基本设置。它没有列出具体的先决条件。但是，根据您的使用案例，可考虑以下先决条件：

- 亚马逊账户和 AWS Marketplace 订阅
  - 要在中设置 Linux 实例 AWS，您需要一个具有管理员访问权限的 AWS 帐户。
  - 确定您计划在哪个 AWS 地区启动实例。确保您计划使用的 AWS 服务在该区域可用。请参阅按区域划分的AWS 服务。

- 确定您计划在其中启动您的实例的 VPC 和子网。
- 如果您的实例需要互联网访问，请确保与您的 VPC 关联的路由表具有互联网网关。如果您的实例需要对本地网络进行 VPN 访问，请确保 VPN 网关可用。VPC 及其子网的确切配置将取决于您的网络设计和应用要求。
- 要订阅特定 AWS Marketplace 服务，请在中选择 AWS Marketplace 订阅， AWS 管理控制台 然后选择管理订阅。
- 搜索您计划使用的服务并订阅它。成功订阅后，您将在管理订阅部分中找到该订阅。从那里您可以直接启动一个新的实例。

- 根据实例的计划用途，实例的硬件和软件先决条件会有所不同：
  - 选项 1：实例将用作 Charon 模拟器主机系统：
    - 请参阅 Charon 产品的《用户指南》和/或《入门》指南中的硬件和软件先决条件部分，以确定 Linux 实例必须满足的确切硬件和软件先决条件。您用于启动实例的映像和您选择的实例类型决定了您的云实例的软件和硬件。
    - 运行模拟遗留系统需要 Charon 产品许可证。请参阅 Charon 产品文档中的许可信息，或者联系您的 Stromasys 代表或 Stromasys VAR 了解更多信息。
  - 选项 2：实例将用作专用 VE 许可证服务器：
    - 有关详细的先决条件，请参阅《VE 许可证服务器指南》。
- 某些可在 Charon 模拟器产品提供的仿真系统中运行的遗留操作系统需要操作系统原始供应商的许可。用户应对与遗留操作系统相关的任何许可义务负责，并且必须提供相应的许可证。

使用 AWS 管理控制台 启动新实例

创建新实例

1. 登录 AWS 管理控制台 并打开亚马逊 EC2 控制台，网址为https://console.aws.amazon.com/ec2/。
2. 选择启动实例。
3. 为实例输入名称。
4. 选择 AMI。AMI 是用于启动云实例的预打包映像。它包括操作系统和适用的应用程序软件。如何选择 AMI 取决于您计划如何使用实例：

   - 如果要将实例用作 Charon 模拟器主机系统，则可以有多个 AMI 选择：
     - 从预打包的 Charon 市场映像安装 Charon 主机系统：它们包含底层操作系统和预装的 Charon 软件。

  - 请咨询您的 Stromasys 代表，了解您的云提供商市场中目前有哪些选项可用。
  - 根据云提供商和 Stromasys 产品发布计划，可能有两种变体：
    - 自动许可（AL）可用于 Stromasys 运营的公共许可证服务器，或用于客户运营的私有 AutoVE 许可证服务器
    - 虚拟环境（VE）可用于客户运营的私有 VE 许可证服务器
- 使用传统的 Charon 模拟器安装（包含适用于 Linux 的 Charon 模拟器安装 RPM 软件包）来安装 Charon 主机系统：
  - 选择由您所选的 Charon 产品和版本提供支持的发行版的 Linux AMI。请参阅 Stromasys 文档网站上的产品用户指南。
- 如果要将实例用作专用 VE 许可证服务器，请参阅许可文档中的《VE 许可证服务器指南》，了解 Linux 实例的要求。

在决定需要哪个 AMI 之后，请选择匹配的 Linux 或 Charon 产品 AMI。如果您没有看到所需的 AMI，请选择浏览更多 AMIs。选择与您计划使用实例的方式相匹配的 Linux AMI。它可以是下列项之一：

- 预打包的 Charon VE 市场映像。AMI 的名称将包含字符串“ve”。
- 自动许可或 AutoVE 的预打包的 Charon AL 市场映像。
- RPM 产品安装支持的 Linux 版本。
- VE 许可证服务器支持的 Linux 版本。

5. 选择一个实例类型。Amazon EC2 提供的实例类型具有不同的 CPU、内存、存储和网络容量组合。选择符合您要使用的 Charon 产品要求的实例类型。一些市场映像的实例类型选择有限。
6. 选择现有密钥对或创建新密钥对并保存。如果您选择现有密钥对，请确保您拥有匹配的私钥。否则，您将无法连接到您的实例。

> **Note**
>
> 如果你的管理系统支持，那么对于 RHEL 9.x、Rocky Linux 9.x 和 Oracle Linux 9.x，请使用 SSH 密钥类型 ECDSA 或。 ED25519这些类型允许您使用 SSH 隧道连接到这些 Charon 主机 Linux 系统，而无需将 Charon 主机上的默认加密策略设置更改为不太安全的设置。例如，这对于 Charon-SSP Manager 很重要。请参阅 Red Hat 文档中的使用系统级加密策略。

7. 在网络设置部分中，选择编辑。选择与您的环境相对应的设置。

- 指定 VPC。
- 指定现有子网或创建新的子网。
- 启用或禁用为主接口自动分配公有 IP 地址的功能。只有当实例只有一个网络接口时，才能自动分配。
- 分配现有或新的自定义安全组。安全组必须至少允许 SSH 访问实例。还必须允许您计划在实例上运行的应用程序所需的任何端口。创建实例后，您可以随时修改安全组。

8. 在存储部分，为根卷（系统磁盘）选择适合您的环境的大小。建议 Linux 系统的最小系统磁盘大小为 30 GiB。要为虚拟磁盘容器和其他存储需求提供空间，您可以立即或在启动实例之后添加更多存储。但是系统磁盘大小必须满足 Linux 系统要求，包括您计划安装的所有应用程序和实用程序。

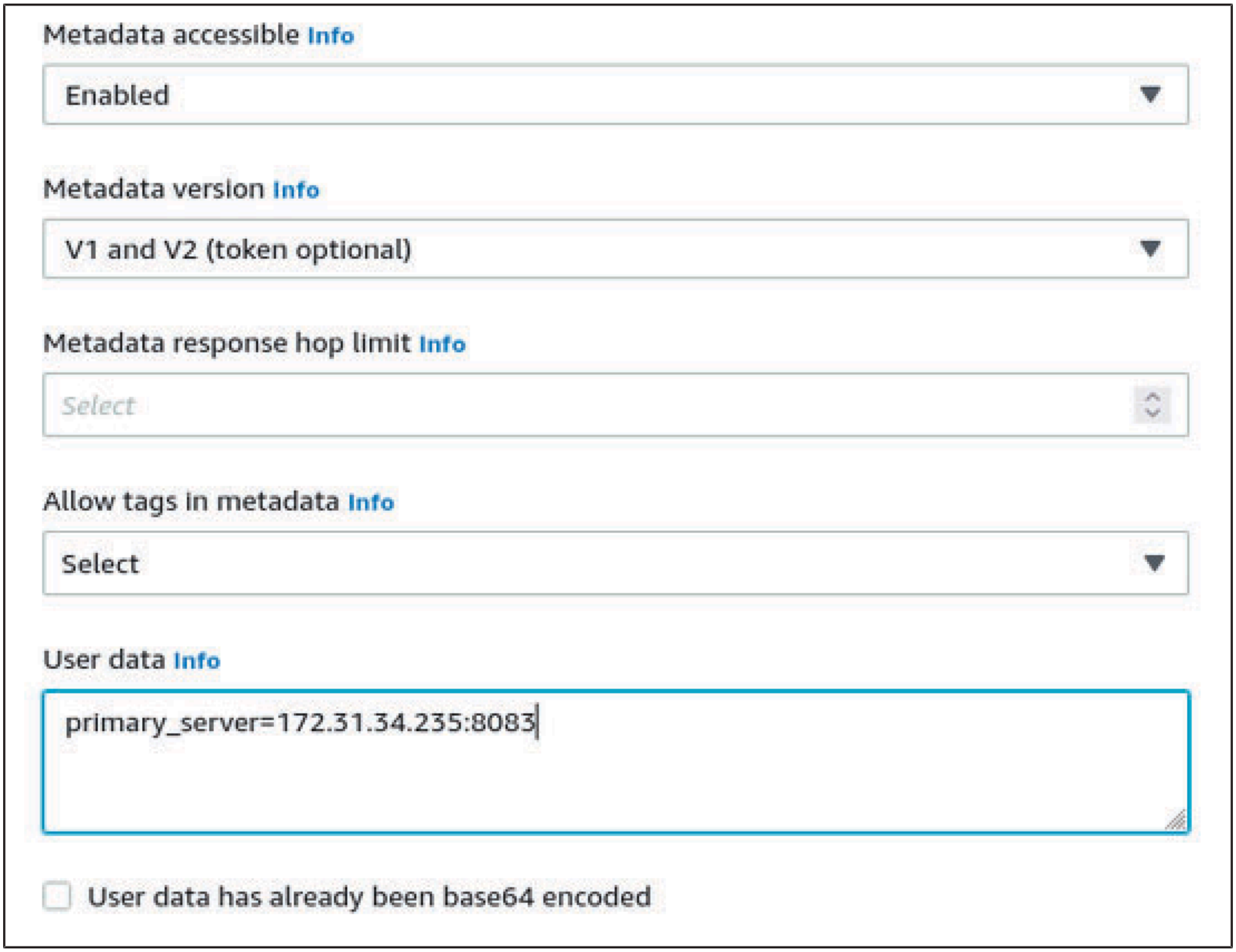
> **Note**
>
> 建议您为 Charon 应用程序数据（例如磁盘映像）创建单独的存储卷。如有必要，可以稍后将此类卷迁移到另一个实例。

9. 展开高级详细信息部分，向下滚动并选择指定 CPU 选项。作为示例，下图显示了三个更可能对 Charon 模拟器环境有用的选项。

10. 对于版本低于 1.1.23 的 VE 许可证服务器系统，您必须为实例分配所需的 IAM 角色。它必须是一个允许执行 `ListUsers` 操作的角色。要分配角色，请在展开的高级详细信息部分，在 IAM 实例配置文件下选择一个角色，或者选择创建新的 IAM 配置文件。有关更多信息，请参阅 Amazon 的 IAM 角色 EC2。

11. 如果您的实例基于 Charon AL AWS Marketplace 映像，并且您计划使用 Stromasys 运营的公共许可证服务器，则必须在启动实例之前将相应信息添加到实例配置中。

    输入 AutoVE 许可证服务器的信息，如下图中所示。

    以下是有效的用户数据配置选项：

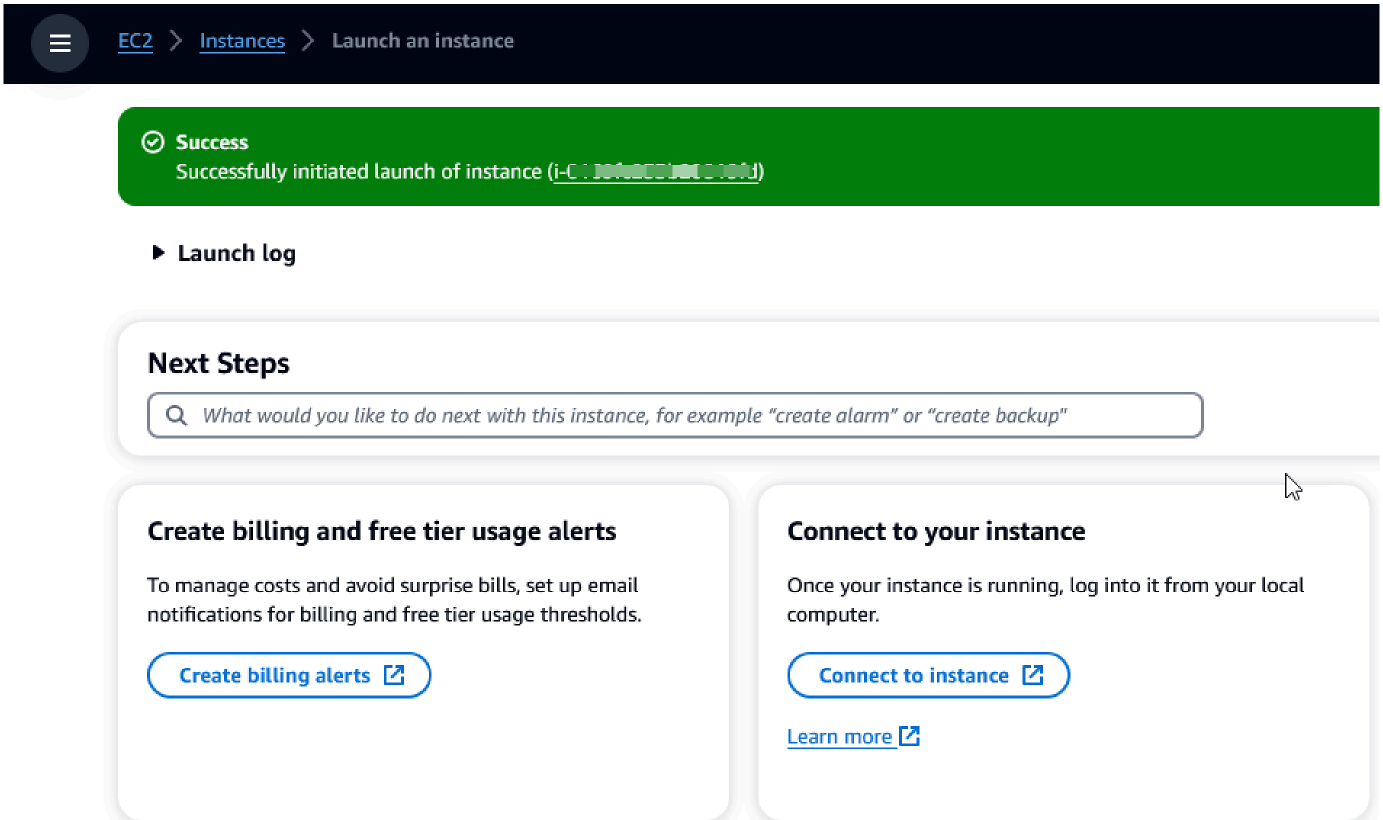
    - **primary_server=***<ip-address>*[**:***<port>*]
    - **backup_server=***<ip-address>*[**:***<port>*]

    位置

    - <ip-address> 代表主服务器和备份服务器（如适用）的 IP 地址。
    - <port> 代表用于与许可证服务器通信的非默认 TCP 端口（默认：TCP/8083）。

> **Note**
>
> 初次启动时必须至少配置一个许可证服务器才能启用 AutoVE 模式。否则，该实例将绑定到 Stromasys 运营的公共许可证服务器之一。

12. 在摘要部分，选择启动实例。过一会儿，您将看到以下成功消息：



13. 在屏幕的右下角，选择查看所有实例。
14. 要查看您的实例的详细信息，请在实例表中选中代表该实例的行左侧的复选框。您的实例详细信息将显示在屏幕的下半部分。有关如何连接到您的实例的信息，请参阅 Amazon EC2 用户指南中的 Connect。

AWS 使用 NTT DATA 实现大型机现代化改造平台

AWS 大型机现代化提供各种 Amazon 机器映像 (AMIs)。它们 AMIs 便于快速配置 Amazon EC2 实例，为使用 NTT Data 重新托管和重构大型机应用程序创建量身定制 AWS 的环境。本指南提供了访问和使用这些内容所需的步骤 AMIs。

先决条件

- 确保您拥有可以创建 Amazon EC2 实例的 AWS 账户的管理员访问权限。
- 确认您计划创建 Amazon EC2 实例的地区是否提供 AWS 大型机现代化服务。查看按区域划分的可用 AWS 服务列表。
- 确定您要在其中创建亚马逊 EC2 实例的 Amazon VPC。

订阅亚马逊机器映像

当您订阅 AWS Marketplace 产品后，可以从该产品的 AMI 启动实例。

1. 登录 AWS 管理控制台 并在 https://console.aws.amazon.com/marketplac AWS Marketplace e 上打开控制台。
2. 选择管理订阅。
3. 将以下链接复制并粘贴到浏览器地址栏中：https://aws.amazon.com/marketplace/pp /prodview-eg227ymldsnx2
4. 选择继续订阅。
5. 如果可以接受条款和条件，请选择接受条款。处理此订阅可能需要几分钟时间。
6. 等待感谢消息出现。该消息确认您是否成功订阅了产品。
7. 在左侧导航窗格中，选择管理订阅。此视图显示您的所有订阅。

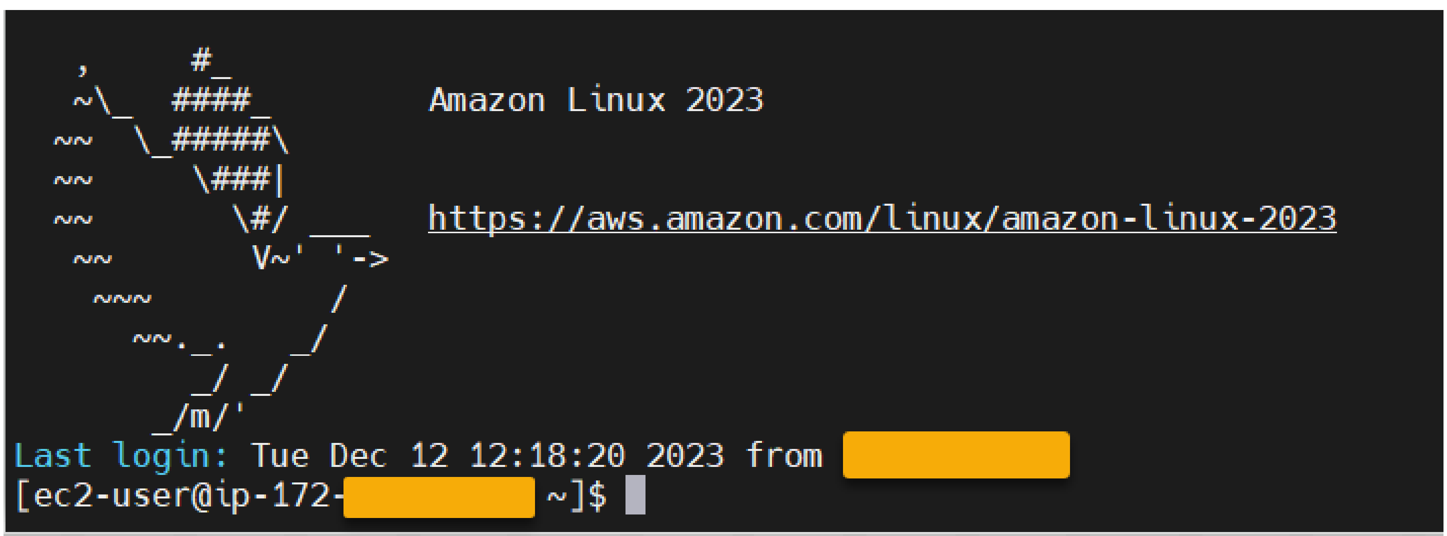
使用 NTT DATA 实例启动 AWS 大型机现代化改造平台

1. 在 https://console.aws.amazon.com/marketplac AWS Marketplace e 上打开控制台。
2. 在左侧导航窗格中，选择管理订阅。
3. 找到您要启动的 AMI，选择启动新实例。

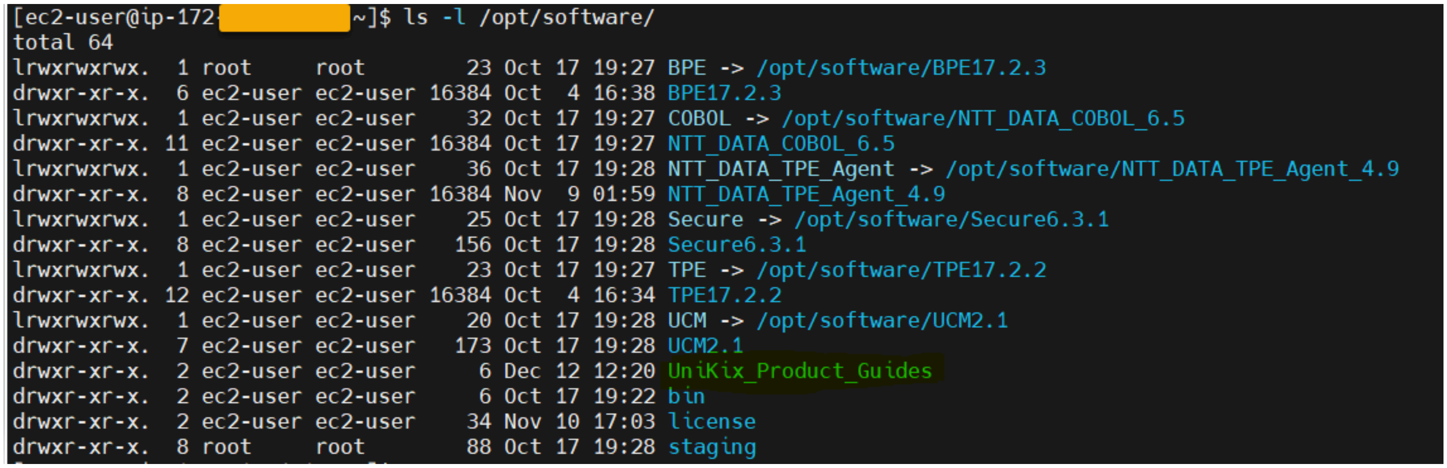
4. 在区域下，选择列为允许的区域。
5. 选择 “继续” 即可启动 EC2。此操作会将您带到 Amazon EC2 控制台。
6. 输入服务器的名称。
7. 选择与您的项目性能和成本要求相匹配的实例类型。建议的实例大小的起点是 c5.2xLarge。
8. 选择现有密钥对或创建新的密钥对并保存。有关密钥对的信息，请参阅亚马逊 EC2 用户指南中的亚马逊 EC2 密钥对和 Linux 实例。
9. 编辑网络设置并选择列为允许的 VPC 和相应的子网。
10. 选择现有安全组或者创建新的安全组。如果这是企业服务器 Amazon EC2 实例，则通常允许流向端口 86 和 10086 的 TCP 流量来管理 Rocket Software（前身为 Micro Focus）配置。
11. 为 Amazon EC2 实例配置存储。
12. 审核摘要并选择启动实例。要成功启动，实例类型必须有效。如果启动失败，请选择编辑实例配置并选择其他实例类型。
13. 看到成功消息后，选择连接到实例。
14. 打开亚马逊 EC2 控制台，网址为https://console.aws.amazon.com/ec2/。
15. 在左侧导航窗格中的实例菜单下，选择实例。
16. 在主窗格中，检查实例的状态。

开始使用 NTT Data

在您配置 Amazon EC2 实例后，使用用户名通过 SSH 进入该实例ec2-user。屏幕显示与下图类似。



在 /opt/software/ 文件夹下，有一个名为 UniKix_Product_Guides 的文件夹，如下图所示。



该UniKix_Product_Guides文件夹包含安装在此 Amazon EC2 实例上的以下组件的文档：

- NTT DATA TPE
- NTT DATA BPE
- NTT DATA Enterprise COBOL
- NTT 数据安全 UniKix
- NTT 数据 UniKix 中心管理器

上图中显示的 software 文件夹包含上面列出的组件的二进制文件。

成功验证 Amazon EC2 实例后，请按照 NTT Data 文档开始使用带有 NTT 数据 AWS 的大型机现代化改造平台。

教程：在 NTT DATA 上部署 CardDemo 应用程序

本页将指导您完成使用 NTT DATA Unikix 运行时在 AWS 大型机现代化改造平台上部署 CardDemo 示例应用程序的 step-by-step过程。

该 CardDemo 示例应用程序是一个简化的大型机应用程序，旨在测试和展示大型机迁移 AWS 和现代化用例的合作伙伴技术。

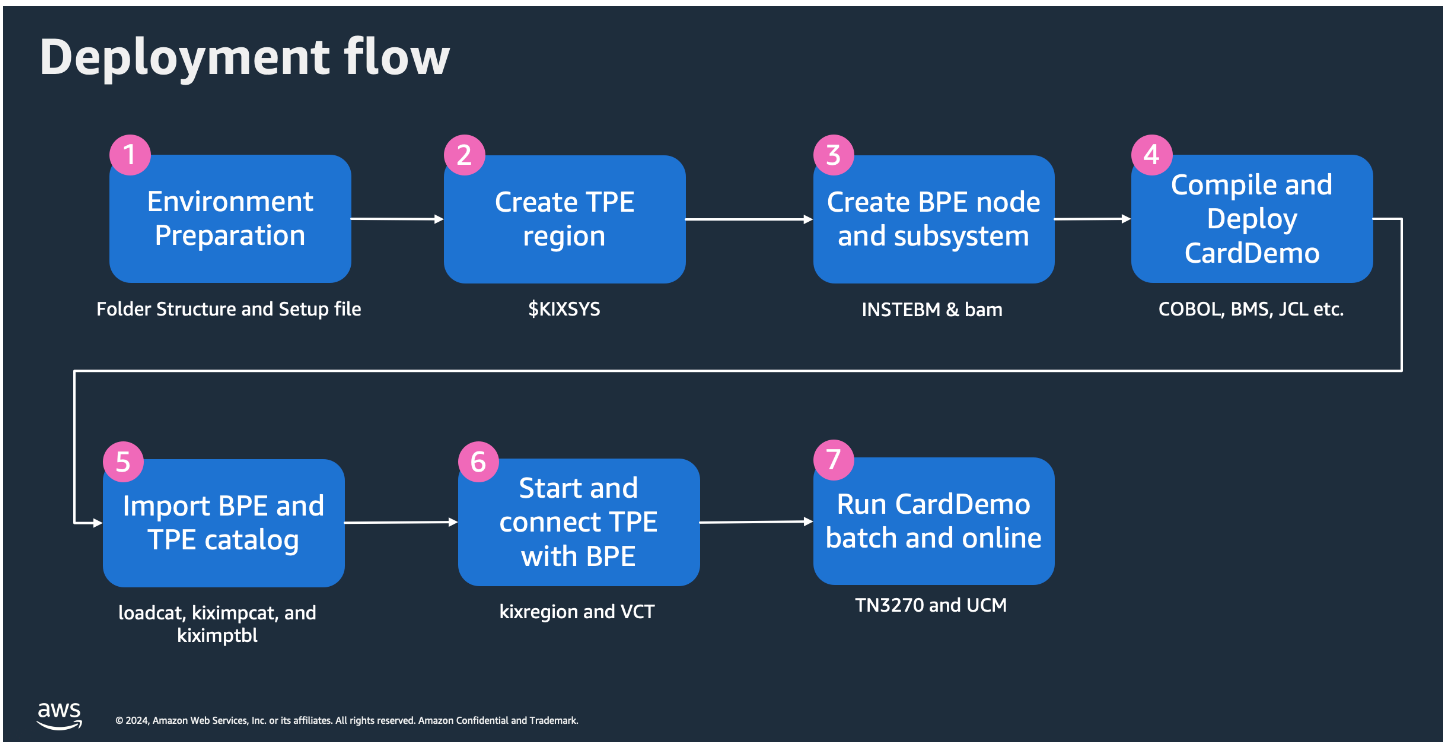
有关此应用程序的更多信息，请参阅，的GitHub存储库 CardDemo。

主题

- 部署流程图
- 先决条件

部署流程图

下图显示了在 NTT DATA Unikix 运行时上部署应用程序的工作流程中的每个步骤。



先决条件

- 按照使用 NTT DATA 进行平台更换使用 NTT DATA Market UniKix place AMI 中提供的说明进行操作。
- 将实例元数据选项修改IMDSv2为可选，如亚马逊 EC2 用户指南 IMDSv1中恢复使用中所述。
- UniKix 从下载 NTT 数据的 CardDemo 运行时组件

GitHub 存储库。

- 以身份登录 UniKix 运行时 EC2 实例ec2-user。
- 使用以下链接提取下载的 CardDemo 运行时组件：UniKix_ CardDemo _runtime_v1.zip。
  - 提取的目录应包含 bin 和 migrated_app 目录。
  - 将 bin 和 migrated_app 目录都移到您的 $HOME 目录下。该路径看起来如下：/home/ec2-user。
  - 您的 $HOME 中应具有以下目录：
    - /home/ec2-user/bin
    - /home/ec2-user/migrated_app
  - 使用以下命令移动 $HOME/bin 目录中的所有文件：
  - • chmod +x $HOME/bin/*

步骤 1：准备环境

完成先决条件后，第一步是准备要在其中部署 CardDemo 应用程序的环境。

1. 以身份登录 UniKix 运行时 EC2 实例ec2-user。
2. 在您的实例中使用以下命令，查看 AMI 中预先打包的 UniKix 软件（例如 TPE、BPE 和 COBOL）以及来自 NTT DATA UniKix 产品位置的其他软件列表： EC2

```
ls -l /opt/software/
```

3. 检查迁移的 CardDemo 应用程序。您将看到所有源代码，包括 BMS 地图、COBOL 程序、COBOL Copybooks 和。 JCLs您还可以通过运行以下命令，找到 BPE 和 TPE 目录、CICS 资源定义以及迁移数据（如顺序文件和 VSAM 文件）的导出：

```
ls  $HOME/migrated_app/*/*
```

4. 通过以下命令运行 create_project 脚本来创建文件夹结构：

```
sh $HOME/bin/create_project
```

5. 使用以下方法获取carddemo.env安装文件来激活 CardDemo 环境：

```
source $HOME/bin/carddemo.env
```

步骤 2：创建 TPE 区域

激活要部署应用程序的环境后，需要创建一个 TPE 区域。

1. 使用 `kixregion createRegion` 命令创建 TPE 区域，该命令需要 `$KIXSYS`、`$JAVA_HOME` 和 `$KIXLICDIR` 等输入。这些环境变量已在 `carddemo.env` 安装文件中设置。

    ```
    kixregion createRegion $KIXSYS $JAVA_HOME $KIXLICDIR
    ```

2. 使用 `kixregion setAttr` 命令配置 TPE 区域。

    ```
    kixregion setAttr $KIXSYS server.tx.languages.cobol.enabled true
    kixregion setAttr $KIXSYS server.tx.languages.cobol.flavor vcobol
    kixregion setAttr $KIXSYS server.tx.languages.cobol.home $VCOBOL
    kixregion setAttr $KIXSYS maps.location $PROJECT_ROOT/maps
    kixregion setAttr $KIXSYS programs.location $PROJECT_ROOT/loadlib
    kixregion setAttr $KIXSYS environment.KIXDATA $KIXDATA
    kixregion setAttr $KIXSYS td.jobq.submission.node $EBMHOME
    kixregion setAttr $KIXSYS td.jobq.submission.subsys $EBMSYS
    ```

3. 通过执行 `kixregion createScript` 命令生成特定于此 TPE 区域的用户环境文件。该命令根据 TPE 区域配置创建或更新 `$KIXSYS/bin/userenv`。

    ```
    kixregion createScript $KIXSYS
    ```

4. 通过获取用户环境文件（`$KIXSYS/bin/userenv`）来激活 TPE 区域。

    ```
    source $KIXSYS/bin/userenv
    ```

5. 通过运行 `kixinstall2` 命令构建 TPE 区域。

    ```
    kixinstall2
    ```

步骤 3：创建 BPE 节点和子系统

创建 TPE 区域后，您需要按照以下步骤创建 BPE 节点和子系统。

1. 更改 `INSTEBM` 的所有权和权限。

    ```
    sudo chown root $INSTEBM
    ```

```
sudo chmod 4755 $INSTEBM
```

2. 使用 `INSTEBM` 命令创建一个 BPE 节点。BPE 节点目录作为输入参数提供。

```
$INSTEBM $EBMHOME
```

3. 通过从新创建的 BPE 节点获取 `batchenv` 文件来激活批处理环境。

```
source $EBMHOME/batchenv
```

4. 使用 Batch Administration Manager（bam）在此节点内创建 BPE 子系统。`bam` 命令将打开 Batch Administration Manager 界面。

```
bam
```

   a. 使用 BAM 界面启动 BPE 节点。从主菜单中选择选项 2 System Environments。

```
                 Batch Administration Manager              2024/09/17 21:25:56




             1   Software License Management
             2   System Environments
             3   Applications & Subsystems
             4   Security & Users
             5   Classes & Activities
             6   Problem Determination



             H - Help
             Q - Quit

--------------------------------------------------------------------------------
Type an option and press Return: 2
```

   b. 选择选项 2 Start/(Stop) Batch Node 以启动 BPE 节点。

```
                 System Environments                         2024/09/17 21:27:03




             1   Report System Status
             2   Start/(Stop) Batch Node
             3   Assign the Console
             4   Change the Date
             5   Redirect Job Output
             6   Inter-Node Communications
             7   Job Accounting
             8   Assign the Initial Job Number
             9   Enable/Disable Duplicate Name Execution Delay : Enabled

             H - Help
             R - Return to Main Menu

--------------------------------------------------------------------------------
Type an option and press Return: 2
```

c. 启动后，按两次 Return 键返回 BAM 主菜单。

```
                 System Environments: Start Batch Node       2024/09/17 21:28:28




              Batch Node Startup Completed.




--------------------------------------------------------------------------------
Press Return to Continue
```

d. 要创建 BPE 子系统，请选择选项 3 Applications & Subsystems。

```
                  Batch Administration Manager                  2024/09/17 21:29:03





              1   Software License Management
              2   System Environments
              3   Applications & Subsystems
              4   Security & Users
              5   Classes & Activities
              6   Problem Determination



              H - Help
              Q - Quit

--------------------------------------------------------------------------------
Type an option and press Return: 3
```

e. 然后选择选项 3 Create a Subsystem。

```
                  Applications & Subsystems                     2024/09/17 21:29:57





              1   List All Subsystems
              2   Query a Subsystem
              3   Create a Subsystem
              4   Update a Subsystem
              5   Delete a Subsystem
              6   Change/Show Default Subsystem
              7   Import a Subsystem (from another batch node)
              8   Create a BPESUB Project
              9   Export Subsystem bldsub Config File

              H - Help
              R - Return to Main Menu

--------------------------------------------------------------------------------
Type an option and press Return: 3
```

f. 输入 `sys1` 作为子系统名称。

```
                 Applications & Subsystems: Create          2024/09/17 21:30:22








         No Subsystems are currently defined.






             R - Return to Previous Screen

--------------------------------------------------------------------------------
Enter the Subsystem's name you want to create: sys1
```

g. 选择选项 3 Data Management。

```
                 Applications & Subsystems: Create          2024/09/17 21:30:53



             D   Display Current sys1 Subsystem's Configuration
             S   Set to Default Subsystem Configuration

             1   Application Languages
             2   Database Management System (DBMS)
             3   Data Management
             4   Optional Packages
             5   Date/Time Management
             6   User-Specific Objects
             7   Configuration Options

             C - Create sys1 Subsystem
             H - Help
             R - Return to Main Menu

--------------------------------------------------------------------------------
Type an option and press Return: 3
```

h. 选择选项 5，因为该 CardDemo 应用程序同时涉及顺序文件和 VSAM 文件。

```
                 Applications & Subsystems: Data Management  2024/09/17 21:31:49


               1   No TPE VSAM Data Management        < Default >

               2   TPE VSAM "RELATIVE/INDEXED" Files

               3   TPE VSAM "RELATIVE", COBOL "INDEXED" Files

               4   COBOL "RELATIVE", TPE VSAM "INDEXED" Files

         ->    5   COBOL "RELATIVE/INDEXED/SEQUENTIAL" Files
                                 and
                         TPE VSAM "RRDS/KSDS/ESDS" Files


               R - Return to Create Menu


--------------------------------------------------------------------------------
Type an option and press Return:
```

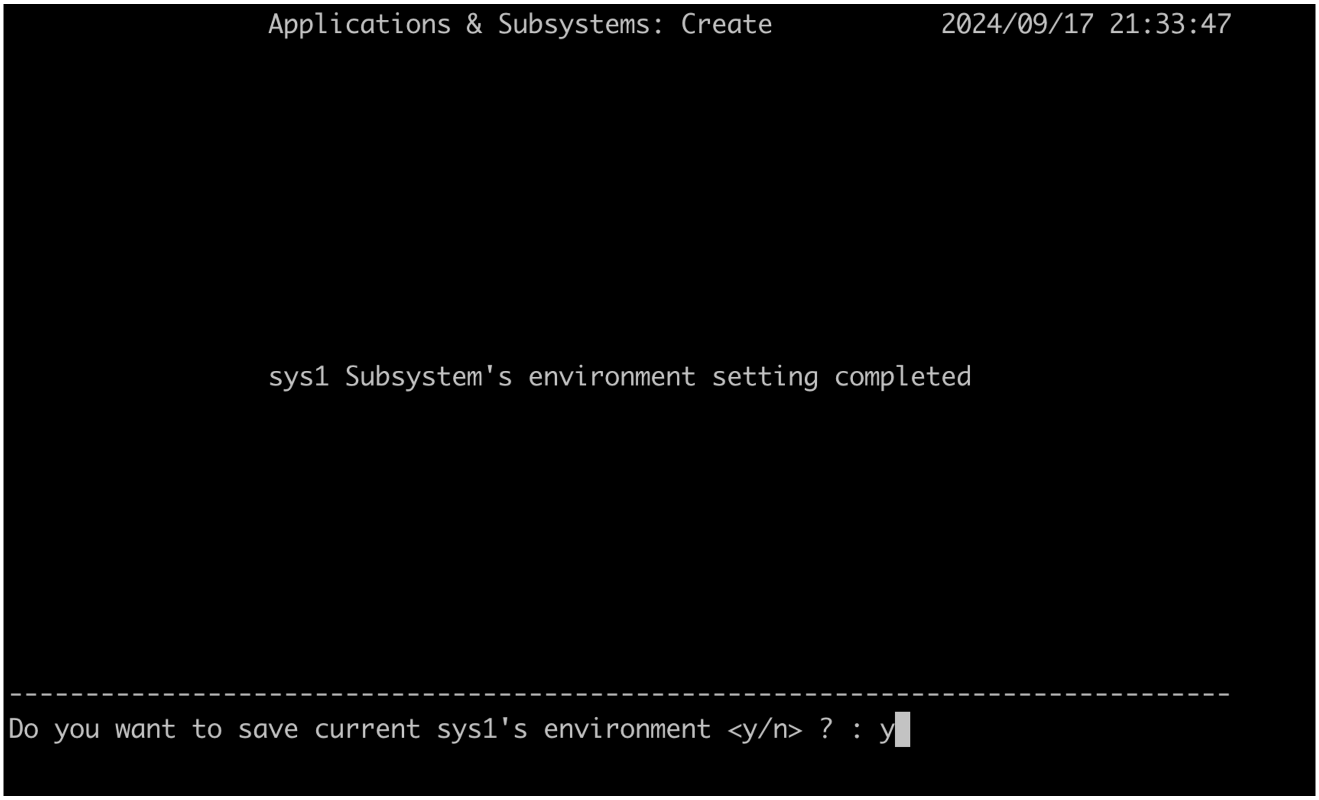
i. （可选）。按“R”返回到 Create Menu 页面，查看可用的不同配置选项。

j. 在 Create 页面上，输入“C”以创建子系统 sys1。

```
                 Applications & Subsystems: Create           2024/09/17 21:32:37



             D   Display Current sys1 Subsystem's Configuration
             S   Set to Default Subsystem Configuration

             1   Application Languages
             2   Database Management System (DBMS)
             3   Data Management
             4   Optional Packages
             5   Date/Time Management
             6   User-Specific Objects
             7   Configuration Options

             C - Create sys1 Subsystem
             H - Help
             R - Return to Main Menu

--------------------------------------------------------------------------------
Type an option and press Return: C
```

k. 查看设置，然后输入“C”，继续执行其余的环境设置。由于 `carddemo.env` 安装文件中已经定义了必要的环境变量并且已存在建议的文件夹结构，这些环境设置已预先填充。

l. 输入“y”以确认并保存当前的环境设置。

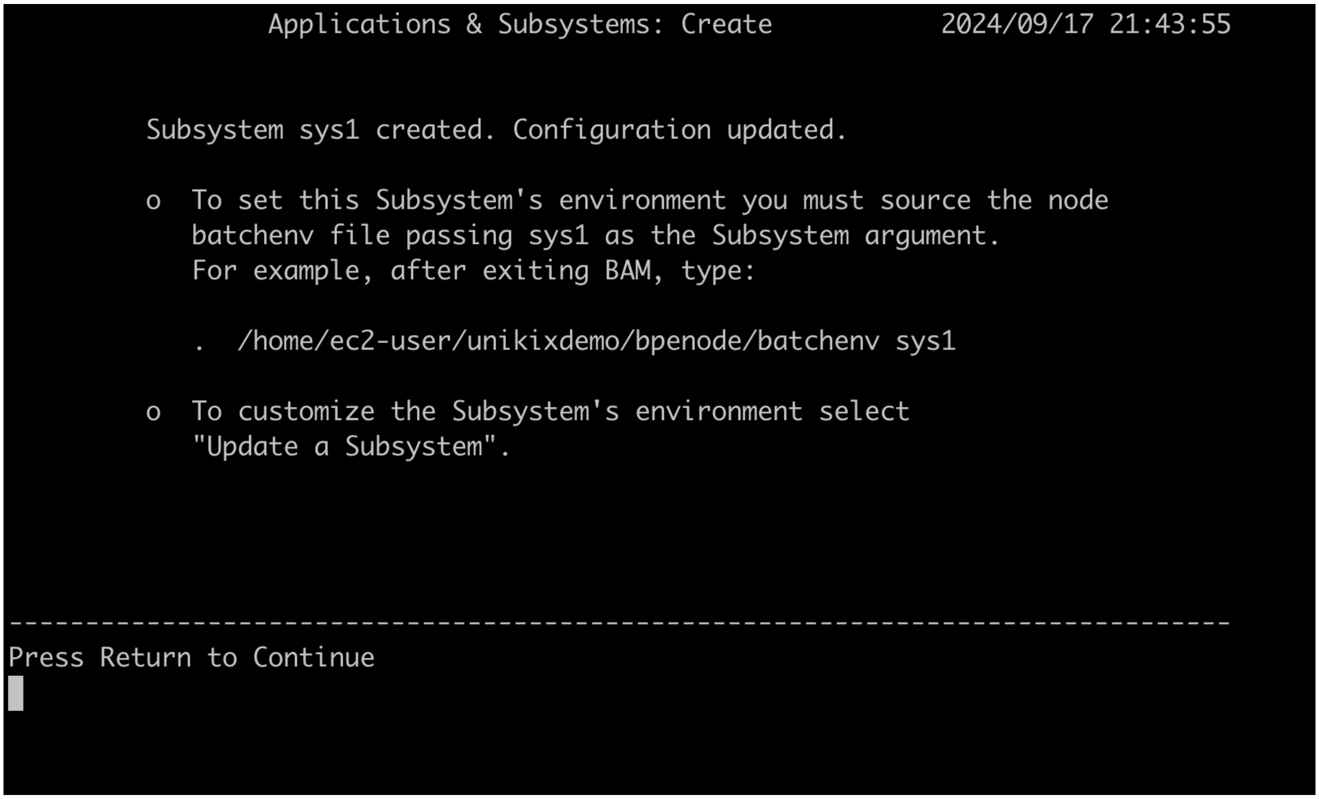


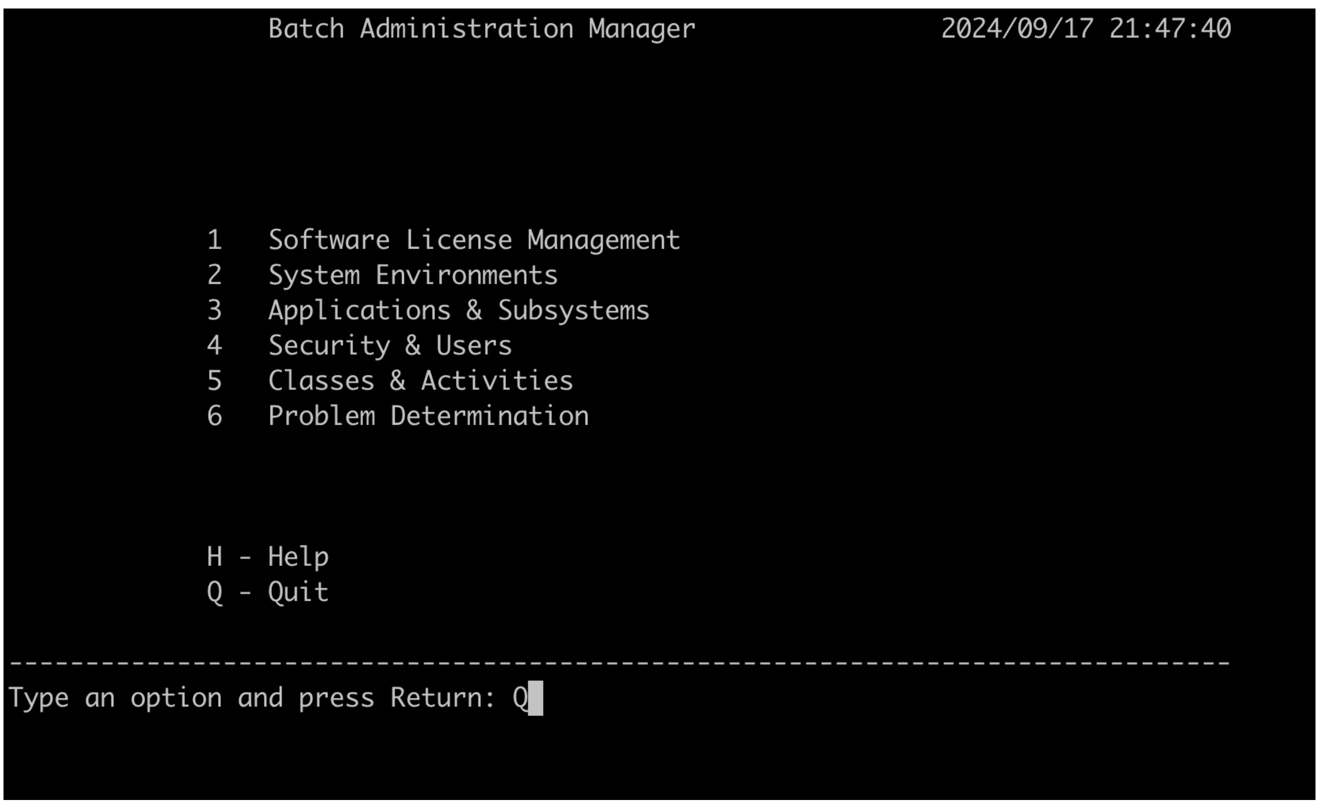
m. 输入“y”以在构建子系统时显示日志。

```
                 Applications & Subsystems: Create          2024/09/17 21:34:17




                 Building sys1's NTT DATA COBOL runtime system




--------------------------------------------------------------------------------
Show log information while building the runtime system ?  <y/n> y
```

n. 按 Return 键直到返回主菜单，然后选择 Quit 选项退出 BAM 界面。

```
                 Applications & Subsystems: Create          2024/09/17 21:43:12




                 COBOL runtime system created




--------------------------------------------------------------------------------
Press Return to Continue
```





5. 通过获取带有子系统名称 sys1 的 batchenv 来激活 BPE 子系统。

```
source $EBMHOME/batchenv sys1
```

步骤 4：编译和部署 CardDemo应用程序

在此步骤中，您将编译 COBOL 程序并部署应用程序构件，例如 JCL、过程、数据文件和 CICS 资源定义。

1. 通过获取`carddemo.env`安装文件再次激活 CardDemo 环境。

```
source $HOME/bin/carddemo.env
```

2. 导航到 COBOL 源目录。

```
cd $MIGAPP_DIR/cbl
```

3. 使用 `compile` 脚本编译 Cobol 程序 `CBACT01C.cbl`。

```
compile CBACT01C.cbl
```

4. 使用 `compile.all` 脚本编译所有 Cobol 程序。

```
compile.all
```

5. 导航到 BMS 映射源目录。

```
cd $MIGAPP_DIR/bms
```

6. 使用 compbms 脚本编译 BMS 映射 `COACTUP.bms`。

```
compbms COACTUP.bms
```

7. 使用 `compbms.all` 脚本编译所有 BMS 映射。

```
compbms.all
```

8. 验证 COBOL 和 BMS 映射的已编译二进制文件。

```
ls $PROJECT_ROOT/loadlib
ls $PROJECT_ROOT/maps
```

9. 使用 deploy_app 脚本部署其他应用程序构件，例如 JCL、过程、数据文件和 CICS 资源定义。

```
deploy_app
```

10. 导航到 JCL 项目目录。

```
cd $PROJECT_ROOT/jcl
```

11. 将 JCL ACCTFILE 转换为 BPE JCL 宏。使用 mvstrans 命令，结合“-v”选项进行 JCL 验证，结合“-f”选项创建宏。

```
mvstrans ACCTFILE -v
mvstrans ACCTFILE -f
```

12. 将 JCL 过程 REPROC 转换为 BPE JCL 过程宏。使用 mvstrans 命令，结合“-p”选项和“-v”选项进行验证，结合“-p”选项和“-f”选项创建宏。

```
mvstrans REPROC -v -p
mvstrans REPROC -f -p
```

13. Translate JCLs all 和 JCL 程序。

```
for file in "./jmvs/*"; do mvstrans $file -f; done > jmvs.out
for file in "./mvsp/*"; do mvstrans $file -p -f; done > mvsp.out
```

步骤 5：导入 BPE 和 TPE 目录

在此步骤中，您将使用不同的命令导入 BPE 和 TPE 目录。

1. 使用 loadcat 命令导入 BPE 目录。

```
loadcat $MIGAPP_DIR/catlg/bpe/BPECAT*
```

2. 导航到 $KIXSYS 目录。

```
cd $KIXSYS
```

3. 使用 kiximpcat 命令导入 TPE 目录。

```
kiximpcat -c CATALOG -l CATALOG.lst
```

4. 使用 kiximptbl 命令导入 CICS 资源定义。

```
kiximptbl
```

步骤 6：启动并连接 TPE 和 BPE

在此步骤中，您需要启动先前创建的 TPE 区域和 BPE 管理器，并将它们连接起来，以便能够运行示例 CardDemo 应用程序。

1. 对所有 VSAM 文件运行 `kixverify` 命令，确保它们已被重置，并且之前打开的文件已关闭。

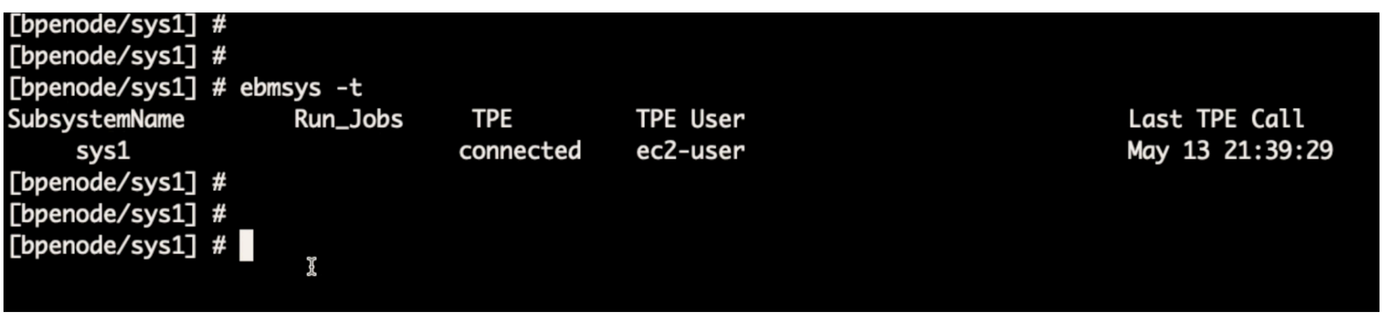
```
kixverify -r ALL
```

2. 启动 TPE 区域。

```
kixregion start $KIXSYS
```

3. 确保 BPE 和 TPE 都已连接。这一点至关重要，因为 VSAM 文件归 TPE 所有，任何访问 VSAM 的批处理操作都需要连接到 TPE。

```
ebmsys -t
```



步骤 7：运行 CardDemo应用程序

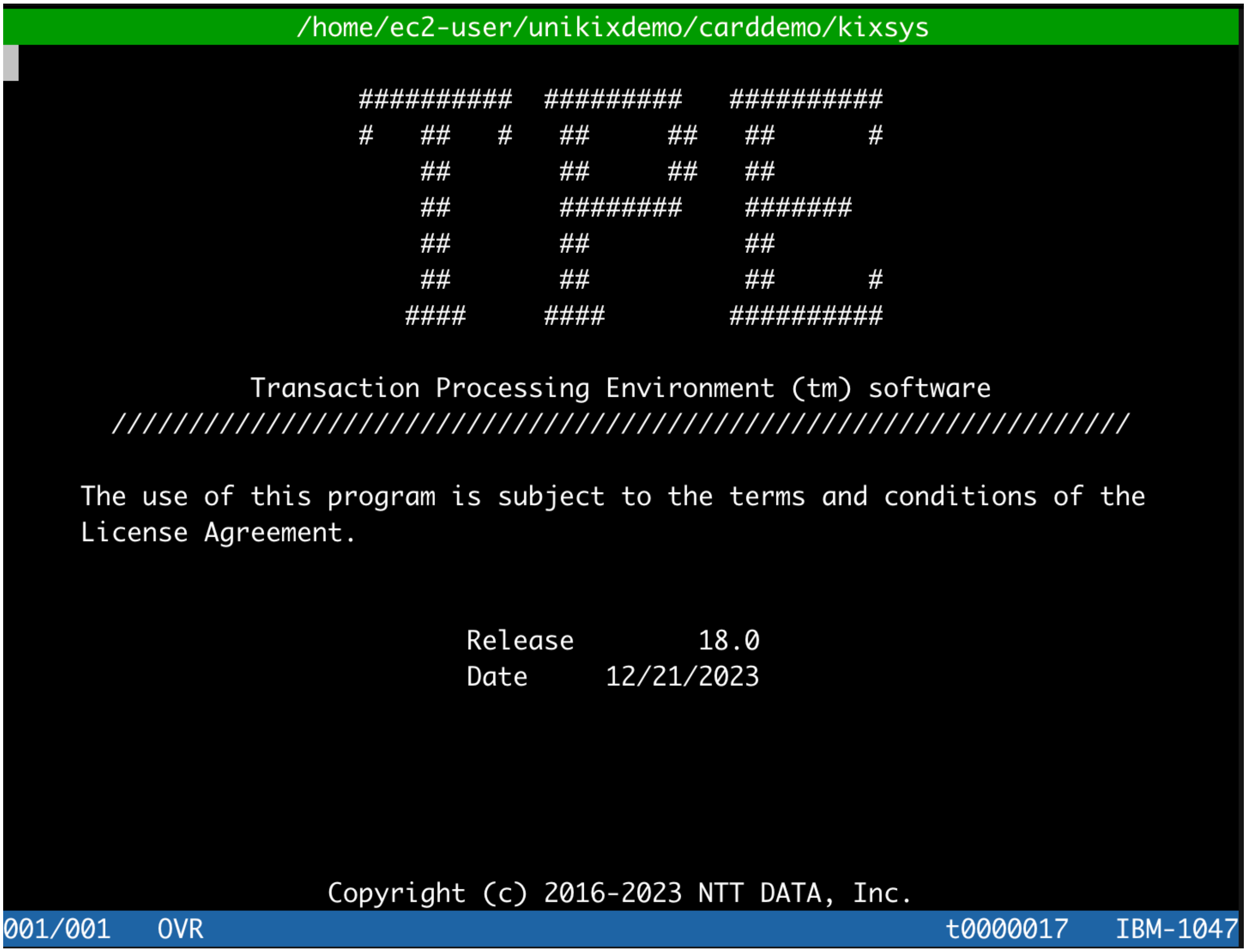
在此步骤中，您将在 TN327 0 终端仿真器中运行 CardDemo 应用程序。

UniKix 运行时 AMI 附带 TN327 0 终端仿真器，您可以直接从 UniKix EC2 实例启动该仿真器。

使用 TN327 0 终端仿真器连接到 TPE

- 使用`kixterm`命令启动 TN327 0 终端。

```
kixterm
```



（可选）。如果您希望使用自己的终端模拟器：

1. 从 Amazon EC2 控制台获取 UniKix 运行时实例的 IP 地址。
2. 使用 TN327 0 终端仿真器获取连接到 TPE 区域的端口号。你可以 TNServer ListenPort 从 unikixrc.cfg 文件中找到这个。

```
cat $KIXSYS/unikixrc.cfg
```

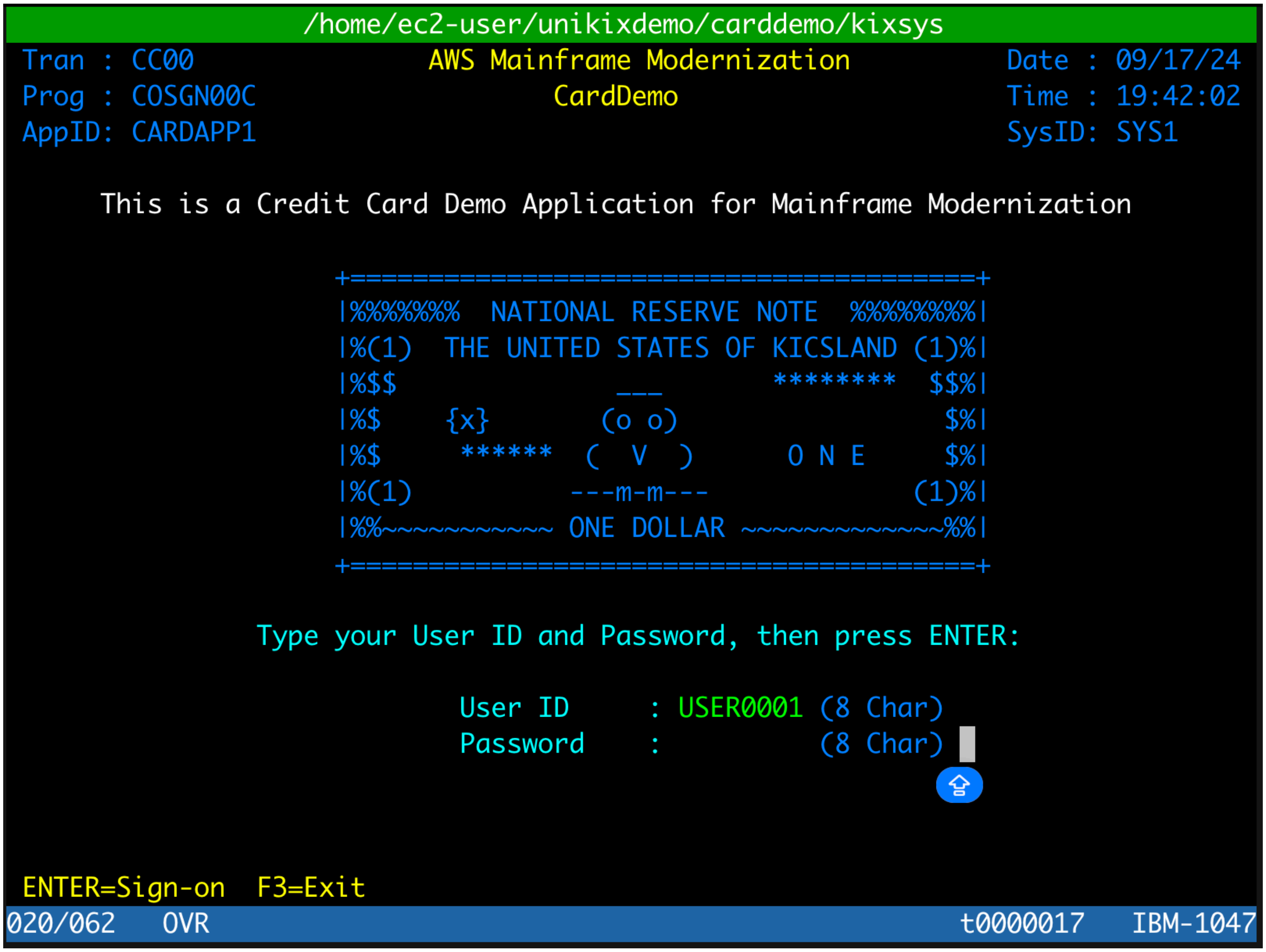
```
! UniKix unikixrc.cfg
TNServer*Active:                            true
TNServer*EndPoints:                         200
TNServer*KeepAlive:                         true
TNServer*ListenPort:                        15440
TNServer*Processes:                         1
TNServer*UserLogin:                         false
```

3. 将 TN327 0 终端仿真器配置为使用 UniKix 运行时实例的 IP 地址和端口号 15440。

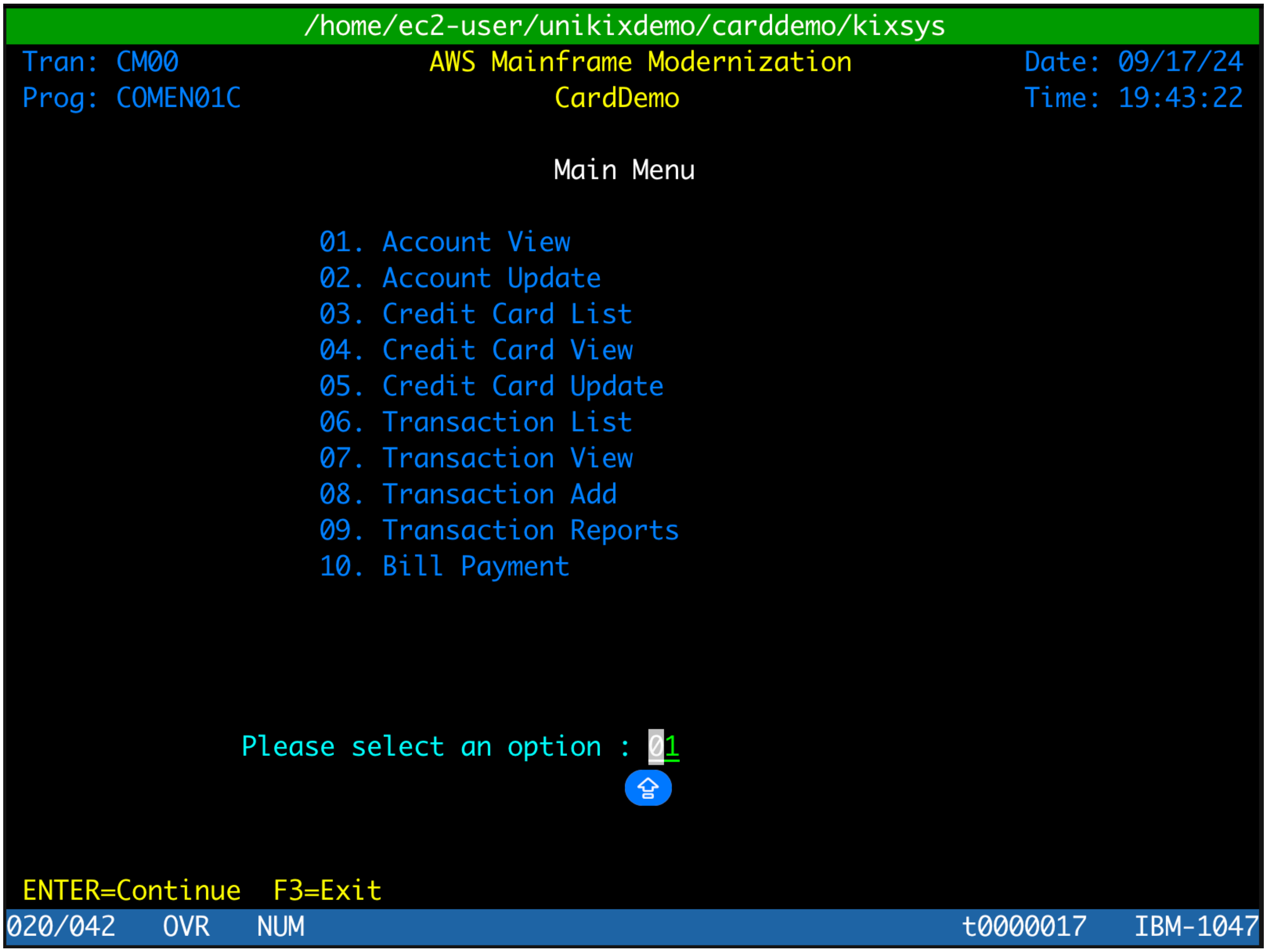
在线事务

本节假设您已使用`kixterm`命令连接到 TN327 0 终端仿真器。

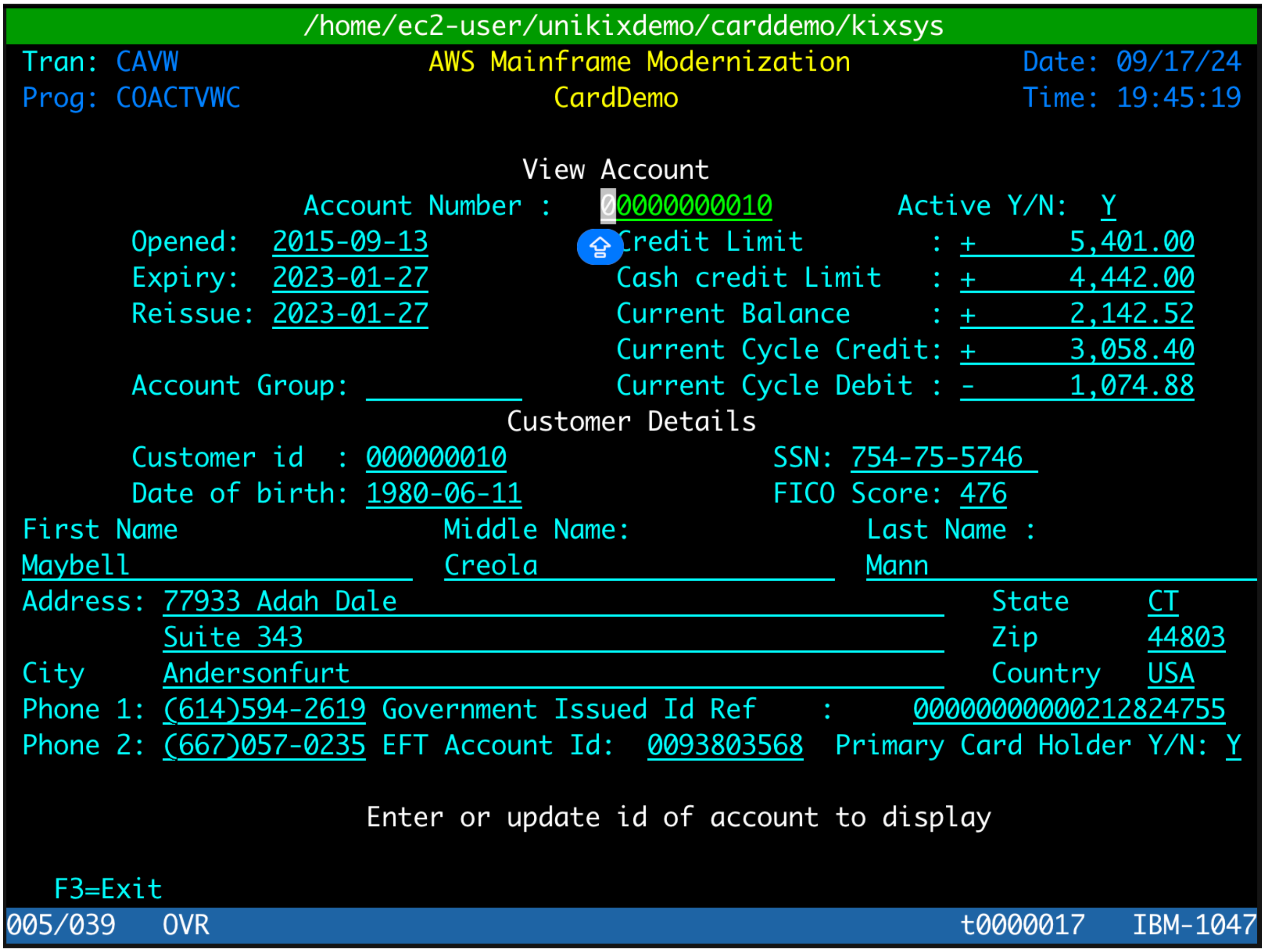
1. 从 TN327 0 终端仿真器连接后，按 “Enter” 键清除 TPE 屏幕并进入初始交易。
2. 在初始事务 CC00（登录屏幕）上，输入用户名 `USER001` 和密码 `PASSWORD`。



3. 从 Main Menu 中选择“01”选项以查看账户。



4. 在 View Account 屏幕中，输入账号（例如 00000000010）。您应该会看到从迁移的数据填充的账户信息。



5. 按两次 “PF03” 键返回登录屏幕，然后按 “Ctrl+C” (Windows) 或 “Cmd+C” (Macbook) 退出 TN3270 终端。

批处理作业

1. 导航到 JCL 目录。

```
cd $MBMSUB
```

2. 提交 `MFCATGL1` 作业并查看作业日志输出。

```
BPESUB READCARD
```

3. 或者，也可以从 `$SUBSYS_OUTDIR` 目录中查看作业日志。

```
ls -lrt $SUBSYS_OUTDIR/*
```

现在，您已成功将 CardDemo 应用程序部署到 NTT DATA UniKix 运行时，并通过浏览几个 CICS 在线屏幕和批处理作业来验证正在运行的应用程序。

问题排查

以下是您在设置 CardDemo 应用程序时可能会发现的一些常见错误。

错误：许可错误

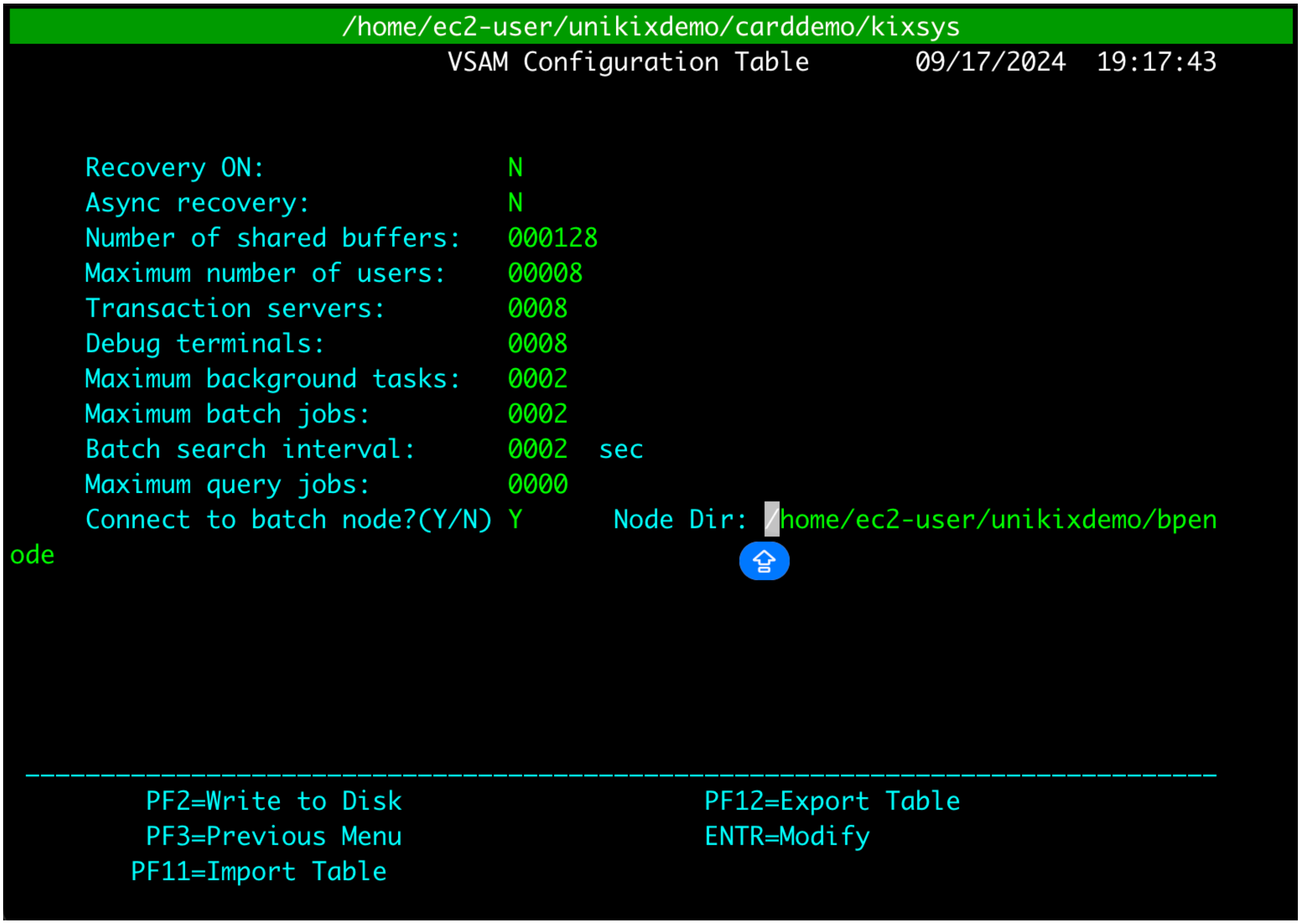
如果您在学习本教程时收到许可证失败错误，则可能是您的已启用 EC2。IMDSv2您可以通过将实例元数据选项修改为可选IMDSv2来解决此问题，如亚马逊 EC2 用户指南 IMDSv1中的恢复使用中所述。

错误：TPE 未连接到 BPE

如果 TPE 未连接到 BPE，请确保使用 BPE 节点目录正确配置 VSAM Configuration Table。要访问 VSAM 配置表，请使用以下命令启动 TN327 0 终端仿真器：

```
kixterm
```

1. 将事务名称输入为 CTBL。
2. 在 Table Manager 菜单中，选择 Standard Tables 选项。
3. 在“Standard Tables”屏幕上，选择 VSAM Configuration Table 选项。
4. 检查 Connect to batch node? 是否设置为 Y 且节点目录正确。


/home/ec2-user/unikixdemo/carddemo/kixsys
VSAM Configuration Table 09/17/2024 19:17:43
Recovery ON: N
Async recovery: N
Number of shared buffers: 000128
Maximum number of users: 00008
Transaction servers: 0008
Debug terminals: 0008
Maximum background tasks: 0002
Maximum batch jobs: 0002
Batch search interval: 0002 sec
Maximum query jobs: 0000
Connect to batch node?(Y/N) Y Node Dir: /home/ec2-user/unikixdemo/bpenode
PF2=Write to Disk
PF12=Export Table
PF3=Previous Menu
ENTR=Modify
PF11=Import Table

AWS 大型机现代化中的安全性

云安全 AWS 是重中之重。作为 AWS 客户，您可以受益于专为满足大多数安全敏感型组织的要求而构建的数据中心和网络架构。

安全是双方共同承担 AWS 的责任。责任共担模式将此描述为云的安全性和云中的安全性：

- 云安全 — AWS 负责保护在 AWS 云中运行 AWS 服务的基础架构。 AWS 还为您提供可以安全使用的服务。作为AWS 合规计划合规计划合规计划合的一部分，第三方审计师定期测试和验证我们安全的有效性。要了解适用于 AWS 大型机现代化的合规计划，请参阅按合规计划划分的 AWS 范围内的服务 AWS 按合规计划。
- 云端安全-您的责任由您使用的 AWS 服务决定。您还需要对其他因素负责，包括您的数据的敏感性、您公司的要求以及适用的法律法规。

本文档可帮助您了解在使用 AWS 大型机现代化时如何应用分担责任模型。它向您展示了如何配置 AWS 大型机现代化以满足您的安全性和合规性目标。您还将学习如何使用其他 AWS 服务来帮助您监控和保护您的 AWS 大型机现代化资源。

AWS 大型机现代化提供自己的 IAM 保护资源（应用程序、环境、部署等），这些资源是 AWS 大型机现代化管理资源，IAM 策略必须允许对其进行任何操作。

AWS 用于平台改造的大型机现代化也受到 IAM 的保护。IAM 还通过标准 IAM 策略授予或拒绝主体对源自原始大型机环境的已定义资源执行特定操作的权限。当应用程序尝试对受保护的资源执行此类操作时， AWS 大型机现代化平台改造运行时会调用 IAM 授权服务。IAM 将根据标准 IAM 策略评估机制返回“允许”或“拒绝”。

内容

AWS 大型机现代化中的数据保护

分 AWS 担责任模型适用于 AWS 大型机现代化中的数据保护。如本模型所述 AWS ，负责保护运行所有内容的全球基础架构 AWS Cloud。您负责维护对托管在此基础结构上的内容的控制。您还负责您所使用的 AWS 服务 的安全配置和管理任务。有关数据隐私的更多信息，请参阅数据隐私常见问题。有关欧洲数据保护的信息，请参阅 AWS Security Blog 上的 AWS Shared Responsibility Model and GDPR 博客文章。

出于数据保护目的，我们建议您保护 AWS 账户 凭证并使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 设置个人用户。这样，每个用户只获得履行其工作职责所需的权限。还建议您通过以下方式保护数据：

- 对每个账户使用多重身份验证（MFA）。
- 用于 SSL/TLS 与 AWS 资源通信。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 使用设置 API 和用户活动日志 AWS CloudTrail。有关使用 CloudTrail 跟踪捕获 AWS 活动的信息，请参阅《AWS CloudTrail 用户指南》中的使用跟 CloudTrail 踪。
- 使用 AWS 加密解决方案以及其中的所有默认安全控件 AWS 服务。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的敏感数据。
- 如果您在 AWS 通过命令行界面或 API 进行访问时需要经过 FIPS 140-3 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准（FIPS）第 140-3 版》https://aws.amazon.com/compliance/fips/。

强烈建议您切勿将机密信息或敏感信息（如您客户的电子邮件地址）放入标签或自由格式文本字段（如名称字段）。这包括您 AWS 服务 使用控制台、API 或进行 AWS 大型机现代化或其他操作时。AWS CLI AWS SDKs在用于名称的标签或自由格式文本字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供 URL，强烈建议您不要在网址中包含凭证信息来验证对该服务器的请求。

AWS 大型机现代化收集的数据

AWS 大型机现代化会从您那里收集几种类型的数据：

- `Application configuration`：此处是指您为配置应用程序而创建的 JSON 文件。它包含您对 AWS 大型机现代化提供的不同选项的选择。该文件还包含相关 AWS 资源的信息，例如存储应用

程序项目的亚马逊简单存储服务路径或存储数据库凭证的 AWS Secrets Manager 亚马逊资源名称 (ARN)。

- `Application executable (binary)`：这是您编译的二进制文件，打算在 AWS 大型机现代化上部署。
- `Application JCL or scripts`：此源代码代表您的应用程序管理批处理作业或其他处理。
- `User application data`：导入数据集时， AWS 大型机现代化将它们存储在关系数据库中，以便您的应用程序可以访问它们。
- `Application source code`：通过 Amazon WorkSpaces 应用程序， AWS 大型机现代化为您提供了编写和编译代码的开发环境。

AWS 大型机现代化以本地方式存储这些数据。 AWS我们收集的您的数据存储在 AWS Mainframe Modernization 托管的 Amazon S3 存储桶中。部署应用程序时， AWS 大型机现代化会将数据下载到由亚马逊弹性区块存储支持的亚马逊弹性计算云实例上。触发清理后，数据将从 Amazon EBS 卷和 Amazon S3 中删除。Amazon EBS 卷是单租户的，这表示一个实例用于一个客户。实例不共享。当您删除运行时环境时，Amazon EBS 卷也会被删除。当您删除应用程序时，构件和配置将从 Amazon S3 中删除。

应用程序日志存储在 Amazon 中 CloudWatch。客户应用程序日志消息也会导 CloudWatch 出到。CloudWatch 日志可能包含客户敏感数据，例如业务数据或调试消息中的安全信息）。有关更多信息，请参阅 使用 A AWS mazon 监控大型机现代化 CloudWatch。

此外，如果您选择将一个或多个 Amazon Elastic File System 或 Amazon FSx 文件系统附加到运行时环境，则这些系统中的数据将存储在中 AWS。如果您决定停止使用这些文件系统，则需要清理相应数据。

当您将数据放入 AWS 大型机现代化用于应用程序部署和数据集导入的 Amazon S3 存储桶时，您可以使用所有可用的 Amazon S3 加密选项来保护数据。此外，如果您将其中一个或多个文件系统附加到运行时环境，则可以使用 Amazon EFS 和 Amazon FSx 加密选项。

AWS 大型机现代化服务的静态数据加密

AWS 大型机现代化与集成， AWS Key Management Service 为所有永久存储数据的依赖资源（即亚马逊简单存储服务、Amazon DynamoDB 和亚马逊弹性区块存储）提供透明的服务器端加密 (SSE)。AWS 大型机现代化在中为您创建和管理对称加密 AWS KMS 密钥。 AWS KMS

默认情况下，静态数据加密有助于降低保护敏感数据的操作开销和复杂性。同时，它还支持迁移需要满足严格加密合规性和监管要求的应用程序。

在创建运行时环境和应用程序时，无法禁用此加密层或选择其他加密类型。

您可以为 AWS 大型机现代化应用程序和运行时环境使用自己的客户托管密钥来加密 Amazon S3 和 Amazon EBS 资源。

对于您的 AWS 大型机现代化应用程序，您可以使用此密钥来加密您的应用程序定义以及其他应用程序资源，例如 JCL 文件，这些资源保存在服务账户中创建的 Amazon S3 存储桶中。有关更多信息，请参阅 创建 应用程序。

对于您的 AWS 大型机现代化运行时环境， AWS 大型机现代化使用您的客户托管密钥加密其创建并附加到您的 AWS 大型机现代化 Amazon 实例（也位于该服务的账户中）的 EC2 Amazon EBS 卷。有关更多信息，请参阅 创建运行时环境。

> **Note**
>
> DynamoDB 资源始终使用大型机现代化服务 AWS 托管式密钥 账户进行 AWS 加密。您无法使用客户托管密钥加密 DynamoDB 资源。

AWS 大型机现代化使用您的客户托管密钥完成以下任务：

- 重新部署应用程序。
- 更换 AWS 大型机现代化 Amazon EC2 实例。

AWS 大型机现代化不会使用您的客户托管密钥来加密亚马逊关系数据库服务或 Amazon Aurora 数据库、亚马逊简单队列服务队列以及为支持 AWS 大型机现代化应用程序而创建的亚马逊 ElastiCache 缓存，因为它们都不包含客户数据。

有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的客户托管密钥。

下表汇总了 AWS 大型机现代化如何加密您的敏感数据。

数据类型	AWS 托管式密钥 加密	客户托管密钥加密
`Definition` 包含特定应用程序的定义。	已启用	已启用
`EnvironmentSummary`	已启用	已启用

数据类型	AWS 托管式密钥 加密	客户托管密钥加密
包含有关运行时环境的信息。		
`ApplicationSummary` 包含有关 AWS 大型机现代化应用程序的信息。	已启用	已启用
`DeploymentSummary` 包含有关部署 AWS 大型机现代化应用程序的信息。	已启用	已启用

> **Note**
>
> AWS 大型机现代化可自动启用静态加密 AWS 托管式密钥 ，从而免费保护您的敏感数据。但是，使用客户管理的密钥需要 AWS KMS 付费。有关定价的更多信息，请参阅 AWS Key Management Service 定价。

有关的更多信息 AWS KMS，请参阅 AWS Key Management Service。

AWS 大型机现代化如何使用补助金 AWS KMS

AWS 大型机现代化需要获得授权才能使用您的客户托管密钥。

当您创建应用程序或运行时环境，或者在使用客户托管密钥加密 AWS 的大型机现代化中部署应用程序时， AWS 大型机现代化将通过向发送CreateGrant请求来代表您创建补助金。 AWS KMS中的赠款 AWS KMS 用于授予 AWS 大型机现代化访问客户账户中的 KMS 密钥的权限。

AWS 大型机现代化需要获得授权，才能使用您的客户托管密钥进行以下内部操作：

- 向发送DescribeKey请求， AWS KMS 以验证在创建应用程序、运行时环境或应用程序部署时输入的对称客户托管密钥 ID 是否有效。
- 向发送GenerateDataKey请求 AWS KMS 以加密托管 AWS 大型机现代化运行时环境的亚马逊 EC2 实例上的 Amazon EBS 卷。
- 向发送解密请求 AWS KMS 以解密 Amazon EBS 上的加密内容。

AWS 在创建运行时环境、创建或重新部署应用程序以及创建部署时，大型机现代化使用 AWS KMS 授权来解密存储在 Secrets Manager 中的密钥。 AWS 大型机现代化所产生的补助金支持以下操作：

- 创建或更新运行时环境授权：
  - Decrypt
  - Encrypt
  - ReEncryptFrom
  - ReEncryptTo
  - GenerateDataKey
  - DescribeKey
  - CreateGrant
- 创建或重新部署应用程序授权：
  - GenerateDataKey
- 创建部署授权：
  - Decrypt

您可以随时撤销授予访问权限，或删除服务对客户托管密钥的访问权限。如果这样做， AWS 大型机现代化将无法访问由客户托管密钥加密的任何数据，这会影响依赖这些数据的操作。例如，如果 AWS 大型机现代化尝试访问由客户托管密钥加密的应用程序定义，但未授予该密钥，则应用程序创建操作将失败。

AWS 大型机现代化收集用户应用程序配置（JSON 文件）和工件（二进制文件和可执行文件）。AWS Mainframe Modernization 还会创建元数据，用于跟踪用于其操作的各种实体，并创建日志和指标。客户可见的日志和指标包括：

- CloudWatch 反映应用程序和运行时引擎（ AWS Blu Age 或 Rocket Software（以前是 Micro Focus））的日志
- CloudWatch 操作仪表板的指标。

此外， AWS 大型机现代化还会收集有关服务的使用数据和指标，用于计量、活动报告等。客户无法看到这些数据。

AWS 大型机现代化根据数据类型将这些数据存储在不同的位置。您上传的客户数据将存储在一个 Amazon S3 存储桶中。服务数据存储在 Amazon S3 和 DynamoDB 中。部署应用程序时，您的数据和

服务数据都会下载到 Amazon EBS 卷上。如果您选择将 Amazon EFS 或 Amazon FSx 存储连接到运行时环境，则存储在这些文件系统中的数据也会下载到 Amazon EBS 卷中。

默认情况下会配置静态加密。您无法将其禁用或更改。目前，您也无法更改其配置。

创建客户托管密钥

您可以使用 AWS 管理控制台 或创建对称的客户托管密钥。 AWS KMS APIs

创建对称的客户托管密钥

按照《AWS Key Management Service 开发人员指南》中创建对称的客户托管密钥的步骤进行操作。

密钥策略

密钥策略控制对客户自主管理型密钥的访问。每个客户托管式密钥必须只有一个密钥策略，其中包含确定谁可以使用密钥以及如何使用密钥的声明。创建客户托管式密钥时，可以指定密钥策略。有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的管理对客户托管密钥的访问。

要将客户托管密钥用于 AWS 大型机现代化资源，密钥策略中必须允许以下 API 操作：

- `kms:CreateGrant`：向客户托管密钥添加授权。授予对指定 KMS 密钥的控制访问权限，该密钥允许访问 AWS 大型机现代化所需的操作。有关使用授权的更多信息，请参阅《AWS Key Management Service 开发人员指南》。

  这样， AWS 大型机现代化就可以做到以下几点：

  - 调用 `GenerateDataKey` 生成加密的数据密钥并将其存储，因为数据密钥不会立即用于加密。
  - 调用 `Decrypt` 使用存储的加密数据密钥访问加密数据。
  - 设置停用主体，以允许服务 `RetireGrant`。
- `kms:DescribeKey`— 提供客户管理的密钥详细信息，以便 AWS 大型机现代化能够验证密钥。

AWS 大型机现代化需要客户密钥策略中的`kms:DescribeKey`权限`kms:CreateGrant`和权限。AWS 大型机现代化使用此政策为自己创建补助金。

JSON

```
{
    "Version":"2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
            },
            "Action": [
                "kms:CreateGrant",
                "kms:DescribeKey"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 前面示例Principal中显示的角色是您用于 AWS 大型机现代化操作的角色，例如CreateApplication和CreateEnvironment。

有关在策略中指定权限的更多信息，请参阅《AWS Key Management Service 开发人员指南》。

有关密钥访问故障排除的信更多息，请参阅《AWS Key Management Service 开发人员指南》。

为 AWS Mainframe Modernization 指定客户托管密钥

您可以为以下资源指定客户托管密钥：

- 应用程序
- 环境

创建资源时，您可以通过输入 KMS ID 来指定密钥， AWS 大型机现代化使用该ID来加密资源存储的敏感数据。

- KMS ID – 客户托管密钥的密钥标识符。输入密钥 ID、密钥 ARN、别名名称或别名 ARN。

您可以使用 AWS 管理控制台 或指定客户托管密钥 AWS CLI。

要在中创建运行时环境时指定您的客户托管密钥 AWS 管理控制台，请参阅创建 AWS 大型机现代化运行时环境。要在中创建应用程序时指定您的客户托管密钥 AWS 管理控制台，请参阅创建 AWS Mainframe Modernization 应用程序。

要在使用创建运行时环境时添加客户托管密钥 AWS CLI，请按如下方式指定`kms-key-id`参数：

```
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
```

要在使用创建应用程序时添加客户托管密钥 AWS CLI，请按如下方式指定`kms-key-id`参数：

```
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 大型机现代化加密环境

加密上下文是一组可选的键值对，包含有关数据的其他上下文信息。

AWS KMS 使用加密上下文作为其他经过身份验证的数据来支持经过身份验证的加密。当您在加密数据的请求中包含加密上下文时，会将加密上下文 AWS KMS 绑定到加密数据。要解密数据，您必须在请求中包含相同的加密上下文。

AWS 大型机现代化加密环境

AWS 大型机现代化在与应用程序相关的所有 AWS KMS 加密操作（创建应用程序和创建部署）中使用相同的加密上下文，其中密钥是`aws:m2:app`，值是应用程序的唯一标识符。

Example

```
"encryptionContextSubset": {
        "aws:m2:app": "a1bc2defabc3defabc4defabcd"
}
```

使用加密上下文进行监控

使用对称客户托管密钥加密应用程序或运行时环境时，您还可以使用审计记录和日志中的加密上下文来识别客户托管密钥的使用情况。

使用加密上下文控制对客户托管式密钥的访问

您可以使用密钥策略和 IAM 策略中的加密上下文作为 `conditions` 来控制对您的对称客户托管密钥的访问。您还可以在授权中使用加密上下文约束。

AWS 大型机现代化在授权中使用加密上下文限制来控制对您账户或区域中客户托管密钥的访问权限。授权约束要求授权允许的操作使用指定的加密上下文。以下示例是 AWS 大型机现代化在创建应用程序时用来加密应用程序工件的赠款。

```
//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”
   }
}
```

监控用于 AWS Mainframe Modernization 的加密密钥

当您在 AWS 大型机现代化资源中使用 AWS KMS 客户托管密钥时，您可以使用AWS CloudTrail或Amazon CloudWatch Logs 来跟踪 AWS 大型机现代化向其发送的请求。 AWS KMS

针对运行时环境的示例

以下示例是`DescribeKey`、`CreateGrant`、和`Decrypt`监控 KMS 操作 AWS CloudTrail 的事件`GenerateDataKey`，这些操作由 AWS 大型机现代化调用，以访问由您的客户托管密钥加密的数据：

DescribeKey

AWS 大型机现代化使用该`DescribeKey`操作来验证 AWS KMS 客户和区域中是否存在与您的运行时环境相关的客户托管密钥。

以下示例事件记录了 `DescribeKey` 操作：

```
{
    "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

当您使用 AWS KMS 客户托管密钥加密运行时环境时， AWS 大型机现代化会代表您发送多个`CreateGrant`请求以执行必要的 KMS 操作。 AWS 大型机现代化创建的某些补助金在使用后会立即停用。其他授权会在您删除运行时环境时停用。

以下示例事件记录了与创建环境工作流程关联的 Lambda 执行角色的 `CreateGrant` 操作。

```
{
    "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"
}
```

以下示例事件记录了自动扩缩组服务相关角色的 `CreateGrant` 操作。与创建环境工作流程关联的 Lambda 执行角色调用 `CreateGrant` 操作。它允许执行角色针对自动扩缩组的服务相关角色创建子授权。


```
{
    "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

当您为运行时环境资源启用 AWS KMS 客户托管密钥时，Auto Scaling 会创建一个用于加密与运行时环境关联的 Amazon EBS 卷的唯一密钥。它向发送`GenerateDataKey`请求 AWS KMS ，指定资源的 AWS KMS 客户托管密钥。

以下示例事件记录了 `GenerateDataKey` 操作：


```
{
    "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

当您访问加密的运行时环境时，Amazon EBS 会调用 `Decrypt` 操作，以便使用存储的加密数据密钥访问加密数据。

以下示例事件记录了 `Decrypt` 操作：


```
{
    "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"
}
```

针对应用程序的示例

以下示例是 AWS 大型机现代化为访问由客户托管密钥加密的数据而调用的 KMS 操作`CreateGrant`和`GenerateDataKey`监控 KMS 操作 AWS CloudTrail 的事件：

CreateGrant

当您使用 AWS KMS 客户托管密钥加密应用程序资源时，Lambda 执行角色会代表您发送访问您账户中的 KMS 密钥的`CreateGrant`请求。 AWS 该授权允许 Lambda 执行角色使用您的客户托管密钥将客户应用程序资源上传到 Amazon S3。该授权将在应用程序创建后立即停用。

以下示例事件记录了 `CreateGrant` 操作：

```
{
    "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

当您为应用程序资源启用 AWS KMS 客户托管密钥时，Lambda 执行角色会创建一个用于加密客户数据并将其上传到亚马逊简单存储服务的密钥。Lambda 执行角色向发送`GenerateDataKey`请求 AWS KMS ，指定资源的 AWS KMS 客户托管密钥。

以下示例事件记录了 `GenerateDataKey` 操作：

```
{
    "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"
```

```
}
```

针对部署的示例

以下示例是 AWS 大型机现代化为访问由客户托管密钥加密的数据而调用的 KMS 操作`CreateGrant`和`Decrypt`监控 KMS 操作 AWS CloudTrail 的事件：

CreateGrant

当您使用 AWS KMS 客户托管密钥加密部署资源时， AWS 大型机现代化将代表您发送两个`CreateGrant`请求。第一个授权针对当前的 Lambda 执行角色进行调用 ListBatchJobScriptFiles，部署完成后立即停用。第二笔拨款针对亚马逊缩小 EC2 范围的实例角色，以便亚马逊 EC2 可以从 Amazon S3 下载客户应用程序资源。该授权在应用程序从运行时环境中删除时即会停用。

以下示例事件记录了 `CreateGrant` 操作：


```
{
    "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

当您访问部署时，Amazon 会 EC2 调用该`Decrypt`操作以使用存储的加密数据密钥来解密并从 Amazon S3 下载加密的客户数据。

以下示例事件记录了 `Decrypt` 操作：


```
{
    "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"
}
```

了解详情

以下资源提供有关静态数据加密的更多信息。

- 有关 AWS Key Management Service 基本概念的更多信息，请参阅 《AWS Key Management Service 开发人员指南》。
- 有关 AWS Key Management Service的安全最佳实操的更多信息，请参阅 《AWS Key Management Service 开发人员指南》。

传输中加密

对于属于事务性工作负载的交互式应用程序，终端仿真器与 TN327 0 协议 AWS 的大型机现代化服务端点之间的数据交换在传输过程中未加密。如果应用程序要求在传输过程中加密，则您可能需要实现一些额外的隧道机制。

AWS 大型机现代化使用 HTTPS 对服务 APIs进行加密。 AWS 大型机现代化中的所有其他通信均受服务 VPC 或安全组以及 HTTPS 保护。 AWS 大型机现代化传输应用程序工件、配置和应用程序数据。应用程序构件是从您的 Amazon S3 存储桶中复制的，应用程序数据也是如此。您可以使用指向 Amazon S3 的链接或通过在本地上传文件来提供应用程序配置。

传输中的基本加密是默认配置的，但不适用于 TN327 0 协议。 AWS 大型机现代化对 API 端点使用 HTTPS，这些端点也是默认配置的。

AWS 大型机现代化的 Identity and Access 管理

AWS Identity and Access Management (IAM) AWS 服务 可帮助管理员安全地控制对 AWS 资源的访问权限。IAM 管理员控制谁可以进行身份验证（登录）和授权（有权限）使用 AWS 大型机现代化资源。您可以使用 IAM AWS 服务 ，无需支付额外费用。

主题

受众

您的使用方式 AWS Identity and Access Management (IAM) 因您的角色而异：

- 服务用户：如果您无法访问功能，请从管理员处请求权限（请参阅AWS 大型机现代化身份和访问权限疑难解答）
- 服务管理员：确定用户访问权限并提交权限请求（请参阅AWS 大型机现代化如何与 IAM 配合使用）
- IAM 管理员：编写用于管理访问权限的策略（请参阅大型机现代化的基于身份的 AWS 策略示例）

使用身份进行身份验证

身份验证是您 AWS 使用身份凭证登录的方式。您必须以 IAM 用户身份进行身份验证 AWS 账户根用户，或者通过担任 IAM 角色进行身份验证。

您可以使用来自身份源的证书 AWS IAM Identity Center （例如（IAM Identity Center）、单点登录身份验证或 Google/Facebook 证书，以联合身份登录。有关登录的更多信息，请参阅《AWS 登录 用户指南》中的如何登录您的 AWS 账户。

对于编程访问， AWS 提供 SDK 和 CLI 来对请求进行加密签名。有关更多信息，请参阅《IAM 用户指南》中的适用于 API 请求的AWS 签名版本 4。

AWS 账户 root 用户

创建时 AWS 账户，首先会有一个名为 AWS 账户 root 用户的登录身份，该身份可以完全访问所有资源 AWS 服务 和资源。我们强烈建议不要使用根用户进行日常任务。有关需要根用户凭证的任务，请参阅《IAM 用户指南》中的需要根用户凭证的任务。

联合身份

作为最佳实践，要求人类用户使用与身份提供商的联合身份验证才能 AWS 服务 使用临时证书进行访问。

联合身份是指来自您的企业目录、Web 身份提供商的用户 Directory Service ，或者 AWS 服务 使用来自身份源的凭据进行访问的用户。联合身份代入可提供临时凭证的角色。

要集中管理访问权限，建议使用。 AWS IAM Identity Center有关更多信息，请参阅《AWS IAM Identity Center 用户指南》中的什么是 IAM Identity Center？。

IAM 用户和群组

IAM 用户是对某个人员或应用程序具有特定权限的一个身份。建议使用临时凭证，而非具有长期凭证的 IAM 用户。有关更多信息，请参阅 IAM 用户指南中的要求人类用户使用身份提供商的联合身份验证才能 AWS 使用临时证书进行访问。

IAM 组指定一组 IAM 用户，便于更轻松地对大量用户进行权限管理。有关更多信息，请参阅《IAM 用户指南》中的 IAM 用户使用案例。

IAM 角色

IAM 角色是具有特定权限的身份，可提供临时凭证。您可以通过从用户切换到 IAM 角色（控制台）或调用 AWS CLI 或 AWS API 操作来代入角色。有关更多信息，请参阅《IAM 用户指南》中的担任角色的方法。

IAM 角色对于联合用户访问、临时 IAM 用户权限、跨账户访问、跨服务访问以及在 Amazon 上运行的应用程序非常有用。 EC2有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

使用策略管理访问

您可以 AWS 通过创建策略并将其附加到 AWS 身份或资源来控制中的访问权限。策略定义了与身份或资源关联时的权限。 AWS 在委托人提出请求时评估这些政策。大多数策略都以 JSON 文档的 AWS 形式存储在中。有关 JSON 策略文档的更多信息，请参阅《IAM 用户指南》中的 JSON 策略概述。

管理员使用策略，通过定义哪个主体可以在什么条件下对哪些资源执行哪些操作来指定谁有权访问什么。

默认情况下，用户和角色没有权限。IAM 管理员创建 IAM 策略并将其添加到角色中，然后用户可以担任这些角色。IAM 策略定义权限，与执行操作所用的方法无关。

基于身份的策略

基于身份的策略是您附加到身份（用户、组或角色）的 JSON 权限策略文档。这些策略控制身份可以执行什么操作、对哪些资源执行以及在什么条件下执行。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的使用客户管理型策略定义自定义 IAM 权限。

基于身份的策略可以是内联策略（直接嵌入到单个身份中）或托管策略（附加到多个身份的独立策略）。要了解如何在托管策略和内联策略之间进行选择，请参阅《IAM 用户指南》中的在托管策略与内联策略之间进行选择。

基于资源的策略

基于资源的策略是附加到资源的 JSON 策略文档。示例包括 IAM 角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。您必须在基于资源的策略中指定主体。

基于资源的策略是位于该服务中的内联策略。您不能在基于资源的策略中使用 IAM 中的 AWS 托管策略。

其他策略类型

AWS 支持其他策略类型，这些策略类型可以设置更常见的策略类型授予的最大权限：

- 权限边界 – 设置基于身份的策略可以授予 IAM 实体的最大权限。有关更多信息，请参阅《 IAM 用户指南》中的 IAM 实体的权限边界。
- 服务控制策略 (SCPs)-在中指定组织或组织单位的最大权限 AWS Organizations。有关更多信息，请参阅《AWS Organizations 用户指南》中的服务控制策略。

- 资源控制策略 (RCPs)-设置账户中资源的最大可用权限。有关更多信息，请参阅《AWS Organizations 用户指南》中的资源控制策略 (RCPs)。
- 会话策略 – 在为角色或联合用户创建临时会话时，作为参数传递的高级策略。有关更多信息，请参阅《IAM 用户指南》中的会话策略。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解在涉及多种策略类型时如何 AWS 确定是否允许请求，请参阅 IAM 用户指南中的策略评估逻辑。

AWS 大型机现代化如何与 IAM 配合使用

在使用 IAM 管理 AWS 大型机现代化的访问权限之前，请先了解有哪些 IAM 功能可用于 AWS 大型机现代化。

可在 AWS 大型机现代化中使用的 IAM 功能

IAM 功能	AWS 大型机现代化支持
基于身份的策略	是
基于资源的策略	否
策略操作	是
策略资源	是
策略条件键	是
ACLs	否
ABAC（策略中的标签）	是
临时凭证	是
转发访问会话（FAS）	是
服务角色	是
服务关联角色	是

要全面了解 AWS 大型机现代化和其他 AWS 服务如何与大多数 IAM 功能配合使用，请参阅 IAM 用户指南中与 IAM 配合使用的AWS 服务。

大型机现代化的基于身份的 AWS 策略

支持基于身份的策略：是

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的使用客户管理型策略定义自定义 IAM 权限。

通过使用 IAM 基于身份的策略，您可以指定允许或拒绝的操作和资源以及允许或拒绝操作的条件。要了解可在 JSON 策略中使用的所有元素，请参阅《IAM 用户指南》中的 IAM JSON 策略元素引用。

大型机现代化的基于身份的 AWS 策略示例

要查看 AWS 大型机现代化基于身份的策略的示例，请参阅。大型机现代化的基于身份的 AWS 策略示例

AWS 大型机现代化中的基于资源的策略

支持基于资源的策略：否

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。委托人可以包括账户、用户、角色、联合用户或 AWS 服务。

要启用跨账户访问，您可以将整个账户或其他账户中的 IAM 实体指定为基于资源的策略中的主体。有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

AWS 大型机现代化的政策行动

支持策略操作：是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

JSON 策略的 `Action` 元素描述可用于在策略中允许或拒绝访问的操作。在策略中包含操作以授予执行关联操作的权限。

要查看 AWS 大型机现代化操作列表，请参阅《服务授权参考》中的 “ AWS 大型机现代化定义的操作”。

AWS 大型机现代化中的策略操作在操作前使用以下前缀：

```
m2
```

要在单个语句中指定多项操作，请使用逗号将它们隔开。

```
"Action": [
      "m2:StartApplication",
      "m2:StopApplication"
         ]
```

您也可以使用通配符（*）指定多个操作。例如，要指定以单词 List 开头的所有操作，包括以下操作：

```
"Action": "m2:List*"
```

要查看 AWS 大型机现代化基于身份的策略的示例，请参阅。大型机现代化的基于身份的 AWS 策略示例

AWS 大型机现代化的政策资源

支持策略资源：是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

Resource JSON 策略元素指定要向其应用操作的一个或多个对象。作为最佳实践，请使用其 Amazon 资源名称（ARN）指定资源。对于不支持资源级权限的操作，请使用通配符 (*) 指示语句应用于所有资源。

```
"Resource": "*"
```

您可以使用特定 AWS 大型机现代化资源 ARNs 来识别 IAM 策略适用的资源，从而限制对这些资源的访问。有关格式的更多信息 ARNs，请参阅中的 Amazon 资源名称 (ARNs) AWS 一般参考。

例如， AWS 大型机现代化环境具有以下 ARN。

```
"Resource": "arn:aws:m2:regionId:accountId:env/service-generated-unique-identifier"
```

AWS 大型机现代化应用程序具有以下 ARN。

```
"Resource": "arn:aws:m2:regionId:accountId:app/service-generated-unique-identifier"
```

并非所有 AWS 大型机现代化操作都支持资源级权限。对于不支持资源级权限的操作，必须使用通配符（*）。

以下 AWS 大型机现代化操作不支持资源级权限。

```
ListApplications
            ListApplicationVersions
            ListBatchJobDefinitions
            ListBatchJobExecutions
            ListDataSetImportHistory
            ListDataSets
            ListDeployments
            ListEngineVersions
            ListEnvironments
            ListTagsForResource
```

要查看 AWS 大型机现代化资源类型及其列表 ARNs，请参阅《服务授权参考》中的 “AWS 大型机现代化定义的资源”。要了解您可以使用哪些操作来指定每种资源的 ARN，请参阅AWS 大型机现代化定义的操作。

要查看 AWS 大型机现代化基于身份的策略的示例，请参阅。大型机现代化的基于身份的 AWS 策略示例

AWS 大型机现代化 API 权限：操作、资源和条件参考

在编写您可附加到 IAM 身份的权限策略（基于身份的策略）时，可以使用下表作为参考。此表中包括以下内容：

- 每个 AWS 大型机现代化 API 操作。
- 您可授予执行该操作的权限的对应操作。
- 您可以为其授予权限的 AWS 资源。

您在策略的 `Action` 字段中指定操作，并在策略的 `Resource` 字段中指定资源值。

您可以在 AWS 大型机现代化策略中使用 AWS 全局条件键来表达条件。有关 AWS 密钥的完整列表，请参阅 IAM 用户指南中的可用全局条件密钥。

> **Note**
> 要指定操作，请在 API 操作名称之前使用 `m2:` 前缀（例如，`m2:CreateApplication`）。

AWS 大型机现代化 API 和操作所需的权限

AWS 大型机现代化 API 操作	所需权限（API 操作）	资源
CancelBatchJobExecution		应用程序

CreateApplication	`iam:PassRole`
	`kms:DescribeKey`
	`kms:CreateGrant`
	`s3:GetObject`
	`s3:ListBucket`
		应用程序

CreateDataSetImportTask	`s3:GetObject`	应用程序
CreateDataSetExportTask	`kms:DescribeKey` `s3:PutObject`	应用程序

CreateDeployment	`elasticloadbalancing:AddTags`
	`elasticloadbalancing:CreateListener`
	`elasticloadbalancing:CreateTargetGroup`
		应用程序

AWS 大型机现代化 API 操作	所需权限（API 操作）	资源
	elasticloadbalancing:RegisterTargets elasticloadbalancing:DeleteListener elasticloadbalancing:DeleteTargetGroup elasticloadbalancing:DeregisterTargets elasticloadbalancing:DeleteLoadBalancer logs:CreateLogDelivery logs:GetLogDelivery logs:UpdateLogDelivery logs:DeleteLogDelivery logs:ListLogDeliveries logs:PutResourcePolicy logs:DescribeResourcePolicies logs:DescribeLogGroups	

AWS 大型机现代化 API 操作	所需权限（API 操作）	资源
CreateEnvironment	ec2:CreateNetworkInterface ec2:CreateNetworkInterfacePermission ec2:DescribeNetworkInterfaces ec2:DescribeSecurityGroups ec2:DescribeSubnets ec2:DescribeVpcAttribute ec2:DescribeVpcs ec2:ModifyNetworkInterfaceAttribute elasticfilesystem:DescribeMountTargets elasticloadbalancing:AddTags elasticloadbalancing:CreateLoadBalancer elasticloadbalancing:DeleteLoadBalancer kms:DescribeKey kms:CreateGrant fsx:DescribeFileSystems	环境

AWS 大型机现代化 API 操作	所需权限（API 操作）	资源
	iam:CreateServiceLinkedRole	
DeleteApplication	elasticloadbalancing:DeleteListener elasticloadbalancing:DeleteTargetGroup logs:DeleteLogDelivery	应用程序
DeleteApplicationFromEnvironment	elasticloadbalancing:DeleteListener elasticloadbalancing:DeleteTargetGroup	应用程序 环境
DeleteEnvironment	elasticloadbalancing:DeleteLoadBalancer	环境
GetApplication		应用程序
GetApplicationVersion		应用程序
GetBatchJobExecution		应用程序
GetDataSetDetails		应用程序
GetDataSetImportTask		应用程序
GetDataSetExportTask		应用程序
GetDeployment		应用程序
GetEnvironment		环境

AWS 大型机现代化 API 操作	所需权限（API 操作）	资源
ListApplications		*
ListApplicationVersions		*
ListBatchJobDefinitions		*
ListBatchJobExecutions		*
ListDataSetImportHistory		*
ListDataSetExportHistory		*
ListDataSets		*
ListDeployments		*
ListEngineVersions		*
ListEnvironments		*
ListTagsForResource		*
StartApplication		应用程序
StartBatchJob		应用程序
StopApplication		应用程序
TagResource		*
UntagResource		*

AWS 大型机现代化 API 操作	所需权限（API 操作）	资源

UpdateApplication	`s3:GetObject`
	
	`s3:ListBucket`
		应用程序

UpdateEnvironment	`kms:DescribeKey`	环境

AWS 大型机现代化的策略条件密钥

支持特定于服务的策略条件键：是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

`Condition` 元素根据定义的条件指定语句何时执行。您可以创建使用条件运算符（例如，等于或小于）的条件表达式，以使策略中的条件与请求中的值相匹配。要查看所有 AWS 全局条件键，请参阅 IAM 用户指南中的AWS 全局条件上下文密钥。

以下条件密钥特定于 AWS 大型机现代化

```
m2:EngineType
            m2:InstanceType
```

要查看 AWS 大型机现代化条件密钥列表，请参阅《服务授权参考》中的 “AWS 大型机现代化条件密钥”。要了解您可以使用哪些操作和资源使用条件键，请参阅AWS 大型机现代化定义的操作。

要查看 AWS 大型机现代化基于身份的策略的示例，请参阅。大型机现代化的基于身份的 AWS 策略示例

AWS 大型机现代化中的访问控制列表 (ACLs)

支持 ACLs：否

访问控制列表 (ACLs) 控制哪些委托人（账户成员、用户或角色）有权访问资源。 ACLs 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

基于属性的访问控制 (ABAC) 和大型机现代化 AWS

支持 ABAC（策略中的标签）：是

基于属性的访问权限控制（ABAC）是一种授权策略，该策略基于称为标签的属性来定义权限。您可以将标签附加到 IAM 实体和 AWS 资源，然后设计 ABAC 策略以允许在委托人的标签与资源上的标签匹配时进行操作。

要基于标签控制访问，您需要使用 `aws:ResourceTag/key-name` `aws:RequestTag/key-name` 或 `aws:TagKeys` 条件键在策略的条件元素中提供标签信息。

如果某个服务对于每种资源类型都支持所有这三个条件键，则对于该服务，该值为是。如果某个服务仅对于部分资源类型支持所有这三个条件键，则该值为部分。

有关 ABAC 的更多信息，请参阅《IAM 用户指南》中的使用 ABAC 授权定义权限。要查看设置 ABAC 步骤的教程，请参阅《IAM 用户指南》中的使用基于属性的访问权限控制（ABAC）。

在 AWS 大型机现代化中使用临时证书

支持临时凭证：是

临时证书提供对 AWS 资源的短期访问权限，并且是在您使用联合身份或切换角色时自动创建的。AWS 建议您动态生成临时证书，而不是使用长期访问密钥。有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的临时安全凭证和使用 IAM 的。AWS 服务

AWS 大型机现代化的转发访问会话

支持转发访问会话（FAS）：是

转发访问会话 (FAS) 使用调用主体的权限 AWS 服务，再加上 AWS 服务 向下游服务发出请求的请求。有关发出 FAS 请求时的策略详情，请参阅转发访问会话。

> **Important**
>
> 这些令牌允许 AWS 大型机现代化在未经您明确同意的情况下访问客户数据；例如， AWS 大型机现代化在未获得客户明确许可的情况下部署应用程序工件以及来自 Amazon S3 存储桶的关联业务数据。您可能需要相应地更新所有合规性文档。

AWS Mainframe Modernization 的服务角色

支持服务角色：是

服务角色是由一项服务担任、代表您执行操作的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 AWS 服务委派权限的角色。

AWS 大型机现代化支持活动挂钩的服务角色（交易/作业中止或完成等）。

> **Warning**
> 更改服务角色的权限可能会破坏 AWS 大型机现代化功能。只有当 AWS 大型机现代化提供相关指导时，才能编辑服务角色。

在 AWS 大型机现代化中选择 IAM 角色

如果您之前创建了在 Amazon 上运行的应用程序 EC2 可以代入的 IAM 角色，则可以在创建启动模板或启动配置时选择此角色。 AWS 大型机现代化为您提供了可供选择的角色列表。创建这些角色时，请关联最低权限 IAM 策略以限制对应用程序所需的特定 API 调用的访问权限，这一点非常重要。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的在亚马逊 EC2 实例上运行的应用程序的 IAM 角色。

AWS 大型机现代化的服务相关角色

支持服务关联角色：是

服务相关角色是一种与服务相关联的 AWS 服务服务角色。服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 AWS 账户 ，并且归服务所有。IAM 管理员可以查看但不能编辑服务关联角色的权限。

有关创建或管理 AWS 大型机现代化服务相关角色的详细信息，请参阅。将服务关联角色用于 AWS Mainframe Modernization

有关创建或管理服务相关角色的详细信息，请参阅能够与 IAM 搭配使用的AWS 服务。在表中查找服务相关角色列中包含 Yes 的表。选择是链接以查看该服务的服务相关角色文档。

大型机现代化的基于身份的 AWS 策略示例

默认情况下，用户和角色无权创建或修改 AWS 大型机现代化资源。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM 策略。

要了解如何使用这些示例 JSON 策略文档创建基于 IAM 身份的策略，请参阅《IAM 用户指南》中的创建 IAM 策略（控制台）。

有关 AWS 大型机现代化定义的操作和资源类型（包括每种资源类型的格式）的详细信息，请参阅《服务授权参考》中的 “AWS 大型机现代化的操作、资源和条件密钥”。 ARNs

主题

策略最佳实践

基于身份的策略决定是否有人可以在您的账户中创建、访问或删除 AWS 大型机现代化资源。这些操作可能会使 AWS 账户产生成本。创建或编辑基于身份的策略时，请遵循以下指南和建议：

- 开始使用 AWS 托管策略并转向最低权限权限 — 要开始向用户和工作负载授予权限，请使用为许多常见用例授予权限的AWS 托管策略。它们在你的版本中可用 AWS 账户。我们建议您通过定义针对您的用例的 AWS 客户托管策略来进一步减少权限。有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管策略或工作职能的AWS 托管策略。
- 应用最低权限：在使用 IAM 策略设置权限时，请仅授予执行任务所需的权限。为此，您可以定义在特定条件下可以对特定资源执行的操作，也称为最低权限许可。有关使用 IAM 应用权限的更多信息，请参阅《IAM 用户指南》中的 IAM 中的策略和权限。
- 使用 IAM 策略中的条件进一步限制访问权限：您可以向策略添加条件来限制对操作和资源的访问。例如，您可以编写策略条件来指定必须使用 SSL 发送所有请求。如果服务操作是通过特定的方式使用的，则也可以使用条件来授予对服务操作的访问权限 AWS 服务，例如 CloudFormation。有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素：条件。
- 使用 IAM Access Analyzer 验证您的 IAM 策略，以确保权限的安全性和功能性：IAM Access Analyzer 会验证新策略和现有策略，以确保策略符合 IAM 策略语言（JSON）和 IAM 最佳实践。IAM Access Analyzer 提供 100 多项策略检查和可操作的建议，以帮助您制定安全且功能性强的策略。有关更多信息，请参阅《IAM 用户指南》中的使用 IAM Access Analyzer 验证策略。
- 需要多重身份验证 (MFA)-如果 AWS 账户您的场景需要 IAM 用户或根用户，请启用 MFA 以提高安全性。若要在调用 API 操作时需要 MFA，请将 MFA 条件添加到您的策略中。有关更多信息，请参阅《IAM 用户指南》中的使用 MFA 保护 API 访问。

有关 IAM 中的最佳实操的更多信息，请参阅《IAM 用户指南》中的 IAM 中的安全最佳实践。

使用 AWS 大型机现代化控制台

要访问 AWS 大型机现代化控制台，您必须拥有一组最低权限。这些权限必须允许您在中列出和查看有关 AWS 大型机现代化资源的详细信息。 AWS 账户如果创建比必需的最低权限更为严格的基于身份的策略，对于附加了该策略的实体（用户或角色），控制台将无法按预期正常运行。

对于仅调用 AWS CLI 或 AWS API 的用户，您无需为其设置最低控制台权限。相反，只允许访问与其尝试执行的 API 操作相匹配的操作。

为确保用户和角色仍然可以使用 AWS 大型机现代化控制台，还要将 AWS 大型机现代化`ConsoleAccess`或`ReadOnly` AWS 托管策略附加到实体。有关更多信息，请参阅《IAM 用户指南》中的为用户添加权限。

允许用户查看他们自己的权限

该示例说明了您如何创建策略，以允许 IAM 用户查看附加到其用户身份的内联和托管式策略。此策略包括在控制台上或使用 AWS CLI 或 AWS API 以编程方式完成此操作的权限。

```
{
    "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": "*"
        }
    ]
```

```
}
```

AWS 大型机现代化身份和访问权限疑难解答

使用以下信息来帮助您诊断和修复在使用 AWS 大型机现代化和 IAM 时可能遇到的常见问题。

主题

我无权执行 iam：PassRole

如果您收到一条错误消息，说您无权执行该`iam:PassRole`操作，则必须更新您的策略以允许您将角色移交给 AWS 大型机现代化。

有些 AWS 服务 允许您将现有角色传递给该服务，而不是创建新的服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

当名为 `marymajor` 的 IAM 用户尝试使用控制台在 AWS Mainframe Modernization 中执行操作时，会发生以下示例错误。但是，服务必须具有服务角色所授予的权限才可执行此操作。Mary 不具有将角色传递到服务的权限。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在这种情况下，必须更新 Mary 的策略以允许她执行 `iam:PassRole` 操作。

如果您需要帮助，请联系您的 AWS 管理员。您的管理员是提供登录凭证的人。

我想允许我以外的人 AWS 账户 访问我的 AWS 大型机现代化资源

您可以创建一个角色，以便其他账户中的用户或您组织外的人员可以使用该角色来访问您的资源。您可以指定谁值得信赖，可以代入角色。对于支持基于资源的策略或访问控制列表 (ACLs) 的服务，您可以使用这些策略向人们授予访问您的资源的权限。

要了解更多信息，请参阅以下内容：

- 要了解 AWS 大型机现代化是否支持这些功能，请参阅AWS 大型机现代化如何与 IAM 配合使用。

- 要了解如何提供对您拥有的资源的访问权限 AWS 账户 ，请参阅 IAM 用户指南中的向您拥有 AWS 账户 的另一个 IAM 用户提供访问权限。
- 要了解如何向第三方提供对您的资源的访问权限 AWS 账户，请参阅 IAM 用户指南中的向第三方提供访问权限。 AWS 账户
- 要了解如何通过身份联合验证提供访问权限，请参阅《IAM 用户指南》中的为经过外部身份验证的用户（身份联合验证）提供访问权限。
- 要了解使用角色和基于资源的策略进行跨账户访问之间的差别，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

将服务关联角色用于 AWS Mainframe Modernization

AWS Mainframe Modernization 使用 AWS Identity and Access Management (IAM) 服务相关角色。服务相关角色是一种与之直接关联的 IAM 角色的独特类型。 AWS Mainframe Modernization服务相关角色由服务预定义 AWS Mainframe Modernization ，包括该服务代表您调用其他 AWS 服务所需的所有权限。

服务相关角色使设置变得 AWS Mainframe Modernization 更加容易，因为您不必手动添加必要的权限。 AWS Mainframe Modernization 定义其服务相关角色的权限，除非另有定义，否则 AWS Mainframe Modernization 只能担任其角色。定义的权限包括信任策略和权限策略，以及不能附加到任何其他 IAM 实体的权限策略。

只有在首先删除相关资源后，您才能删除服务关联角色。这样可以保护您的 AWS Mainframe Modernization 资源，因为您不会无意中删除访问资源的权限。

有关支持服务相关角色的其他服务的信息，请参阅与 IAM 配合使用的AWS 服务，并在服务相关角色列中查找标有 “是” 的服务。请选择是与查看该服务的服务相关角色文档的链接。

的服务关联角色权限 AWS Mainframe Modernization

AWS Mainframe Modernization 使用名为的服务相关角色 AWSServiceRoleForAWSM2— 将网络配置为连接到您的 VPC 并访问文件系统等资源。

AWSServiceRoleForAWSM2 服务相关角色信任以下服务来代入该角色：

- `m2.amazonaws.com`

名为的角色权限策略 AWSM2ServicePolicy AWS Mainframe Modernization 允许对指定资源完成以下操作：

- 创建、删除、描述和附加权限到该 AWS Mainframe Modernization 环境的 Amazon EC2 网络接口，以建立与客户 VPC 的连接。
- 从 Elastic Load Balancing 中注册或注销条目，这是客户连接到 AWS Mainframe Modernization 环境的方式。
- 描述 Amazon EFS 或亚马逊 FSx 文件系统（如果使用）。
- CloudWatch 从运行时环境向客户发送指标。

JSON

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

您必须配置权限，允许 IAM 实体（如用户、组或角色）创建、编辑或删除服务关联角色。有关更多信息，请参阅《IAM 用户指南》中的服务相关角色权限。

为 创建服务关联角色 AWS Mainframe Modernization

您无需手动创建服务关联角色。当您在 AWS 管理控制台、或 AWS API 中创建运行时环境时， AWS Mainframe Modernization 会为您创建服务相关角色。 AWS CLI

如果您删除该服务关联角色，然后需要再次创建，您可以使用相同流程在账户中重新创建此角色。当您创建运行时环境时， AWS Mainframe Modernization 会再次为您创建服务相关角色。

为 AWS Mainframe Modernization编辑服务关联角色

AWS Mainframe Modernization 不允许您编辑 AWSServiceRoleForAWSM2 服务相关角色。创建服务关联角色后，您将无法更改角色的名称，因为可能有多种实体引用该角色。但是可以使用 IAM 编辑角色描述。有关更多信息，请参阅《IAM 用户指南》中的编辑服务相关角色。

删除 AWS Mainframe Modernization的服务关联角色

如果不再需要使用某个需要服务关联角色的功能或服务，我们建议您删除该角色。这样就没有未被主动监控或维护的未使用实体。但是，必须先清除服务相关角色的资源，然后才能手动删除它。

> **Note**
>
> 如果您尝试删除资源时 AWS Mainframe Modernization 服务正在使用该角色，则删除可能会失败。如果发生这种情况，请等待几分钟后重试。

删除使用的 AWS Mainframe Modernization 资源 AWSService RoleFor AWSM2

- 删除中的运行时环境 AWS Mainframe Modernization。在删除环境本身之前，请务必从环境中删除应用程序。

使用 IAM 手动删除服务关联角色

使用 IAM 控制台 AWS CLI、或 AWS API 删除 AWSServiceRoleForAWSM2服务相关角色。有关更多信息，请参阅《IAM 用户指南》中的删除服务相关角色。

AWS Mainframe Modernization 服务相关角色的受支持区域

AWS Mainframe Modernization 支持在提供服务的所有地区使用服务相关角色。有关更多信息，请参阅 AWS 区域和端点。

AWS 大型机现代化的合规性验证

作为多项合规计划的一部分，第三方审计师评估 AWS 大型机现代化的安全 AWS 性和合规性。其中包括 SOC、PCI、FedRAMP、HIPAA 及其他。

有关特定合规计划范围内的 AWS 服务列表，请参阅按合规计划划分的 AWS 范围内的服务 AWS 按合规计划。有关常规信息，请参阅 AWS 合规性计划、、。

您可以使用下载第三方审计报告 AWS Artifact。有关更多信息，请参阅中的 “下载报告” 中的 “ AWS Artifact。

在使用 AWS 大型机现代化时，您的合规责任取决于数据的敏感性、公司的合规目标以及适用的法律和法规。 AWS 提供了以下资源来帮助实现合规性：

- 安全性与合规性快速入门指南 - 这些部署指南讨论了架构注意事项，并提供了在 AWS上部署基于安全性和合规性的基准环境的步骤。
- HIPAA 安全与合规架构白皮书 — 本白皮书描述了公司如何使用来 AWS 创建符合 HIPAA 标准的应用程序。
- AWS 合规资源AWS — 此工作簿和指南集可能适用于您所在的行业和所在地区。
- 使用AWS Config 开发人员指南中的规则评估资源 — AWS Config; 评估您的资源配置在多大程度上符合内部实践、行业准则和法规。
- AWS Security Hub CSPM— 此 AWS 服务可全面了解您的安全状态 AWS ，帮助您检查是否符合安全行业标准和最佳实践。

AWS 大型机现代化的弹性

AWS 全球基础设施是围绕 AWS 区域和可用区构建的。各区域提供多个在物理上独立且隔离的可用区，这些可用区通过延迟低、吞吐量高且冗余性高的网络连接在一起。利用可用区，您可以设计和操作在可用区之间无中断地自动实现故障转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错能力和可扩展性。

有关 AWS 区域和可用区的更多信息，请参阅AWS 全球基础设施。

中的基础设施安全 AWS Mainframe Modernization

作为一项托管服务 AWS Mainframe Modernization ，受 AWS 全球网络安全的保护。有关 AWS 安全服务以及如何 AWS 保护基础设施的信息，请参阅AWS 云安全。要使用基础设施安全的最佳实践来设计您的 AWS 环境，请参阅 S AWS ecurity Pillar Well-Architected Fram ework 中的基础设施保护。

您可以使用 AWS 已发布的 API 调用 AWS Mainframe Modernization 通过网络进行访问。客户端必须支持以下内容：

- 传输层安全性协议（TLS）。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 具有完全向前保密（PFS）的密码套件，例如 DHE（临时 Diffie-Hellman）或 ECDHE（临时椭圆曲线 Diffie-Hellman）。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

AWS Mainframe Modernization 使用 AWS PrivateLink 接口端点进行访问

您可以使用 AWS PrivateLink 在您的 VPC 和之间创建私有连接 AWS Mainframe Modernization。您可以像在 VPC 中 AWS Mainframe Modernization 一样进行访问，无需使用互联网网关、NAT 设备、VPN 连接或 Direct Connect 连接。VPC 中的实例不需要公有 IP 地址即可访问 AWS Mainframe Modernization。

您可以通过创建由 AWS PrivateLink提供支持的接口端点来建立此私有连接。我们将在您为接口端点启用的每个子网中创建一个端点网络接口。这些是请求者托管的网络接口，用作发往 AWS Mainframe Modernization的流量的入口点。

有关更多信息，请参阅AWS PrivateLink 指南 AWS PrivateLink中的AWS 服务 通过访问。

的注意事项 AWS Mainframe Modernization

在为设置接口端点之前 AWS Mainframe Modernization，请查看AWS PrivateLink 指南中的注意事项。

AWS Mainframe Modernization 支持通过接口端点调用其所有 API 操作。

为创建接口终端节点 AWS Mainframe Modernization

您可以创建用于 AWS Mainframe Modernization 使用 Amazon VPC 控制台或 AWS Command Line Interface (AWS CLI) 的接口终端节点。有关更多信息，请参阅《AWS PrivateLink 指南》中的创建接口端点。

AWS Mainframe Modernization 使用以下服务名称创建接口终端节点：

```
com.amazonaws.region.m2
```

如果为接口端点启用私有 DNS，则可使用其默认区域 DNS 名称向 AWS Mainframe Modernization 发出 API 请求。例如 `m2.us-east-1.amazonaws.com`。

为 VPC 端点创建端点策略

端点策略是一种 IAM 资源，您可以将其附加到接口端点。默认终端节点策略允许 AWS Mainframe Modernization 通过接口终端节点进行完全访问。要控制允许 AWS Mainframe Modernization 从您的 VPC 访问权限，请将自定义终端节点策略附加到接口终端节点。

端点策略指定以下信息：

- 可执行操作的主体（AWS 账户、用户和 IAM 角色）。
- 可执行的操作。
- 可对其执行操作的资源。

有关更多信息，请参阅《AWS PrivateLink 指南》中的使用端点策略控制对服务的访问权限。

示例：用于 AWS Mainframe Modernization 操作的 VPC 终端节点策略

以下是自定义端点策略的一个示例。将此策略附加到接口端点时，其会向所有资源上的所有主体授予对列出的 AWS Mainframe Modernization 操作的访问权限。

```
//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":"*"
      }
   ]
}
```

监控 AWS 大型机现代化

监控是维护 AWS 大型机现代化和您的其他 AWS 解决方案的可靠性、可用性和性能的重要组成部分。AWS 提供以下监控工具，用于监控 AWS 大型机现代化情况、报告问题并在适当时自动采取行动：

- Amazon 会实时 CloudWatch监控您的 AWS 资源和您运行 AWS 的应用程序。您可以收集和跟踪指标，创建自定义的控制平面，以及设置警报以在指定的指标达到您指定的阈值时通知您或采取措施。例如，您可以 CloudWatch 跟踪您的 Amazon EC2 实例的 CPU 使用率或其他指标，并在需要时自动启动新实例。有关更多信息，请参阅 Amazon CloudWatch 用户指南。
- Amazon Lo CloudWatch gs 使您能够监控、存储和访问来自亚马逊 EC2 实例和其他来源的日志文件。 CloudTrail CloudWatch 日志可以监视日志文件中的信息，并在达到特定阈值时通知您。您还可以在高持久性存储中检索您的日志数据。有关更多信息，请参阅 Amazon CloudWatch 日志用户指南。
- AWS CloudTrail捕获由您的账户或代表您的 AWS 账户进行的 API 调用和相关事件，并将日志文件传输到您指定的 Amazon S3 存储桶。您可以识别哪些用户和账户拨打了电话 AWS、发出呼叫的源 IP 地址以及呼叫发生的时间。有关更多信息，请参阅 AWS CloudTrail 《用户指南》。

使用 A AWS mazon 监控大型机现代化 CloudWatch

您可以使用监控 AWS 大型机现代化 CloudWatch，它会收集原始数据并将其处理为可读的近乎实时的指标。这些统计数据会保存 15 个月，从而使您能够访问历史信息，并能够更好地了解您的 Web 应用程序或服务的执行情况。此外，可以设置用于监测特定阈值的警报，并在达到相应阈值时发送通知或执行操作。有关更多信息，请参阅 Amazon CloudWatch 用户指南。

下表列出了 AWS 大型机现代化的指标和维度。这些指标的命名空间为 `AWS/M2`。

运行时环境指标

指标	描述
CPUUtilization	环境中实例的 CPU 利用率。 维度：environmentId 单位：百分比

指标	描述
	有效统计数据：Average、Minimum、Maximum
InboundNetworkThroughput	环境中实例的入站网络吞吐量。 维度：environmentId 单位：字节/秒 有效统计数据：Average、Minimum、Maximum
MemoryUtilization	环境中实例的内存利用率。 维度：environmentId 单位：百分比 有效统计数据：Average、Minimum、Maximum
OutboundNetworkThroughput	环境中实例的出站网络吞吐量。 维度：environmentId 单位：字节/秒 有效统计数据：Average、Minimum、Maximum

应用程序指标

指标	描述
BatchJobCompletedCount	在时间间隔内完成的作业数。 该指标适用于 Rocket Software（前身为 Micro Focus）和 AWS Blu Age 3.7.0 及更高版本。 维度：applicationId 单位：计数

指标	描述
	有效统计数据：Sum
BatchJobFailedCount	在时间间隔内失败的作业数。 该指标适用于 Rocket Software 和 AWS Blu Age 3.7.0 及更高版本。 维度：applicationId 单位：计数 有效统计数据：Sum
JvmMemoryFree	Java 虚拟机当前未使用的可用内存。 此指标仅适用于 AWS Blu Age 运行时引擎。适用于 AWS Blu Age 3.7.0 及更高版本。 维度：applicationId 单位：字节 有效统计数据：Average、Minimum、Maximum
JvmMemoryMax	Java 虚拟机允许的最大内存量。 此指标仅适用于 AWS Blu Age 运行时引擎。适用于 AWS Blu Age 3.7.0 及更高版本。 维度：applicationId 单位：字节 有效统计数据：Average、Minimum、Maximum

指标	描述
JvmMemoryUsed	Java 虚拟机已经使用的内存量。 此指标仅适用于 AWS Blu Age 运行时引擎。适用于 AWS Blu Age 3.7.0 及更高版本。 维度：applicationId 单位：字节 有效统计数据：Average、Minimum、Maximum
ProcessesActiveCount	正在处理请求的活跃并发服务执行进程的数量。 此指标仅适用于 Rocket Software 运行时引擎。 维度：applicationId 单位：计数 有效统计数据：Sum
SessionCount	应用程序的 HTTP 会话数。 此指标仅适用于 AWS Blu Age 运行时引擎。适用于 AWS Blu Age 3.7.0 及更高版本。 维度：applicationId 单位：计数 有效统计数据：Average、Minimum、Maximum

指标	描述
SharedMemoryFree	Enterprise Server 可用的内存，用于存储运行事务和作业所需的所有信息。 此指标仅适用于 Rocket Software 运行时引擎。 维度：applicationId 单位：千字节 有效统计数据：Average、Minimum、Maximum
SharedMemoryTotal	分配给企业服务器的共享内存总量，用于存储运行事务和作业所需的所有信息。 此指标仅适用于 Rocket Software 运行时引擎。 维度：applicationId 单位：千字节 有效统计数据：Average、Minimum、Maximum
ThreadActiveCount	正在处理请求的引擎线程数。 此指标仅适用于 AWS Blu Age 运行时引擎。适用于 AWS Blu Age 3.7.0 及更高版本。 维度：applicationId 单位：计数 有效统计数据：Average、Minimum、Maximum

指标	描述
TransactionCompletedCount	在时间间隔内提交的事务数。 该指标适用于 Rocket Software 和 AWS Blu Age 3.7.0 及更高版本。 维度：applicationId 单位：计数 有效统计数据：Sum
TransactionFailedCount	在时间间隔内失败的事务数。 该指标适用于 Rocket Software 和 AWS Blu Age 3.7.0 及更高版本。 维度：applicationId 单位：计数 有效统计数据：Sum
TransactionResponseTime	从用户发送请求时起，到应用程序指示请求已完成的时间。 该指标适用于 Rocket Software 和 AWS Blu Age 3.7.0 及更高版本。 维度：applicationId 单位：毫秒 有效统计数据：Average、Minimum、Maximum

Dimensions

维度	描述
applicationId	此维度筛选指标，应用到按 ID 识别的应用程序。
environmentId	此维度筛选指标，应用到按 ID 识别的环境。

使用记录 AWS 大型机现代化 API 调用 AWS CloudTrail

AWS 大型机现代化与 AWS CloudTrail一项服务集成，该服务提供用户、角色或 AWS 服务在 AWS 大型机现代化中采取的操作的记录。 CloudTrail 将 AWS 大型机现代化的所有 API 调用捕获为事件。捕获的调用包括来自 AWS 大型机现代化控制台的调用和对 AWS 大型机现代化 API 操作的代码调用。如果您创建跟踪，则可以将 CloudTrail 事件持续传输到 Amazon S3 存储桶，包括 AWS 大型机现代化事件。如果您未配置跟踪，您仍然可以在 CloudTrail 控制台的 “事件历史记录” 中查看最新的事件。通过收集的信息 CloudTrail，您可以确定向 AWS 大型机现代化提出的请求、发出请求的 IP 地址、谁提出了请求、何时提出请求以及其他详细信息。

要了解更多信息 CloudTrail，请参阅《AWS CloudTrail 用户指南》。

AWS 大型机现代化信息位于 CloudTrail

CloudTrail 在您创建 AWS 账户时已在您的账户上启用。当 AWS 大型机现代化中发生活动时，该活动会与其他 AWS 服务 CloudTrail 事件一起记录在事件历史记录中。您可以在自己的 AWS 账户中查看、搜索和下载最近发生的事件。有关更多信息，请参阅使用事件历史记录查看 CloudTrail 事件。

要持续记录 AWS 账户中的事件，包括 AWS 大型机现代化事件，请创建跟踪。跟踪允许 CloudTrail 将日志文件传输到 Amazon S3 存储桶。默认情况下，在控制台中创建跟踪记录时，此跟踪记录应用于所有 AWS 区域。跟踪记录 AWS 分区中所有区域的事件，并将日志文件传送到您指定的 Amazon S3 存储桶。此外，您可以配置其他 AWS 服务，以进一步分析和处理 CloudTrail 日志中收集的事件数据。有关更多信息，请参阅下列内容：

- 创建跟踪记录概述
- CloudTrail 支持的服务和集成
- 配置 Amazon SNS 通知 CloudTrail
- 接收来自多个区域的 CloudTrail 日志文件

- 接收来自多个账户的 CloudTrail 日志文件

所有 AWS 大型机现代化操作均由《大型机现代化 API 参考》记录 CloudTrail 并记录在《AWS 大型机现代化 API 参考》中。例如，调用`CreateEnvironment`和`CreateDeployment`操作会在 CloudTrail 日志文件中生成条目。`CreateApplication`

每个事件或日志条目都包含有关生成请求的人员信息。身份信息有助于您确定以下内容：

- 请求是使用根用户凭证还是用户凭证发出的。
- 请求是使用角色还是联合用户的临时安全凭证发出的。
- 请求是否由其他 AWS 服务发出。

有关更多信息，请参阅 CloudTrail userIdentity 元素。

了解 AWS 大型机现代化日志文件条目

跟踪是一种配置，允许将事件作为日志文件传输到您指定的 Amazon S3 存储桶。 CloudTrail 日志文件包含一个或多个日志条目。事件代表来自任何来源的单个请求，包括有关请求的操作、操作的日期和时间、请求参数等的信息。 CloudTrail 日志文件不是公共 API 调用的有序堆栈跟踪，因此它们不会按任何特定的顺序出现。

以下示例显示了演示该`CreateApplication`操作的 CloudTrail 日志条目。

```
{
    "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"
}
```

AWS 大型机现代化中的故障排除

使用本节中的信息来帮助您解决同时使用 AWS Blu Age 和 Rocket Software 引擎 AWS 的大型机现代化应用程序和运行时环境中的常见错误。

主题

错误故障排除：等待解锁数据集名称时超时

本页介绍当您在环境中看到另一个应用程序锁定共享数据集时，如何解决错误。

- 引擎： AWS Blu Age
- 组件：Blusam

如果您在使用 AWS Blu Age 引擎并在具有高可用性模式的环境中运行 AWS 的大型机现代化应用程序的 Amazon CloudWatch 日志中看到此错误，则表示另一个应用程序锁定了共享数据集。通常，如果其他应用程序崩溃或以其他方式失败并且没有释放锁定，就会发生这种情况。

查找失败的应用程序，并检查其是否在使用错误消息中提及的相同数据集。检查应用程序是否在具有高可用性模式的运行时环境中运行。引发超时异常的应用程序无法继续运行，并将显示 `Failed` 状态。

常见原因

应用程序 `example-app-1` 尝试锁定写入操作的 `example-record-1` 记录。此操作同时创建了对数据集 `example-dataset-1`（包含 `example-record-1`）的锁定和对 `example-record-1` 自身的锁定。现在，另一个应用程序 `example-app-2` 尝试锁定相同的记录 `example-record-1`。数据集和记录已被锁定，因此 `example-app-2` 等待锁定释放。如果 `example-app-1` 崩溃，则对数据集 `example-dataset-1` 的锁定仍然存在，这会导致 `example-app-2` 取消其写入尝试并引发超时异常。这种死锁情况使所有应用程序都无法访问 `example-dataset-1`。

解决方案

要立即解决此问题，您可以强制解除锁定。为了防止将来发生类似的情况，您可以配置两个控制 Blusam自动修复机制的参数。

强制释放锁定

Blusam锁管理器使用亚马逊 ElastiCache （Redis OSS）在应用程序之间提供共享锁。要解除锁定 ElastiCache，请使用 Redis CLI 实用工具。您无法删除单个记录锁定，而是必须从拥有的数据集中移除所有锁定。完成以下步骤：

1. ElastiCache 使用以下命令连接到您的：

   ```
   redis-cli -h hostname -p port
   ```

   您可以在 ElastiCache 控制台 ElastiCache 中找到您的详细信息，网址为 https://console.aws.amazon.com/elasticache/。
2. 输入您的密码。
3. 输入要运行的命令，如下所示：

命令	用途
KEYS *	获取所有现有密钥。
钥匙 * *YOUR_DATASET_NAME*	获取数据集锁定密钥。
DEL *THE_RETURNED_KEY*	删除数据集锁
FLUSHDB	清理整个 Redis。 Warning Redis 缓存中的所有数据都将丢失。如果 Redis 用于其他目的，例如处理 http 会话，您可能不希望使用 FLUSHDB。

配置 a Blusam uto 修复机制

Blusam锁管理器包括自动修复机制，可防止数据集或记录出现死锁。您可以调整应用程序定义（`application-main.yml`）中的以下参数来配置自动修复机制：

- `locksDeadTime`：指应用程序可以持有锁定的最长时间。经过此时间后，该锁定声明为过期并立即释放。`locksDeadTime` 值以毫秒为单位，默认值为 1000。
- `locksCheck`: 定义了用于检查Blusam锁的锁管理器策略。所有Blusam锁定 ElastiCache 都有时间戳并有到期时间。`locksCheck` 参数值决定是否移除过期的锁。
  - `off`：任何时候均不会执行任何检查。可能发生死锁。（不推荐使用）
  - `reboot`：在 AWS 大型机现代化运行时环境中运行的大型 AWS 机现代化应用程序实例启动或重启时，将执行检查。所有过期的锁将立即释放。（默认值）
  - `timeout`：在 AWS 大型机现代化运行时环境中运行的大型 AWS 机现代化应用程序实例启动或重新启动时，或者在尝试锁定数据集期间超时到期时，将执行检查。过期的锁将立即释放。

有关 AWS Blu Age 应用程序的应用程序定义的更多信息，请参阅AWS Blu Age 应用程序定义示例。

Blusam锁管理器

在使用高可用性模式 AWS 的大型机现代化运行时环境中，可能会多次部署 AWS Blu Age 应用程序。对于那些处理Blusam数据集的应用程序，可能会出现并发访问问题。Blusam锁管理器通过使用在应用程序之间提供共享锁来确保数据完整性并管理对记录和数据集的读写访问权限 ElastiCache。这种机制支持多个应用程序同时读取记录，并确保一次只有一个应用程序写入记录。

写入锁定

要更新或删除特定记录，应用程序必须先锁定拥有该记录的数据集，然后锁定记录本身。当记录被锁定时，数据集的锁定就会被释放，同一数据集中的其他记录可供使用。更新或删除操作完成后，保持的记录锁定即会被释放。如果定义的应用程序策略允许等待释放，则一次只有一个应用程序可以更新记录，这将阻止其他应用程序在锁定被释放之前读取或写入。

读取锁定

只要记录或数据集没有写入锁定，多个应用程序即可以同时读取相同的记录。要锁定写入操作的记录，必须释放所有读取锁定。

> **Note**
>
> Blusam锁管理器使用相同的锁定机制处理来自给定应用程序中多个线程的访问。

错误故障排除：无法访问应用程序 URL

本页介绍当无法访问正在运行的 AWS Mainframe Modernization 应用程序的 URL 时，如何解决错误。

- 引擎： AWS Blu Age 和 Rocket Software（以前是
- 组件：应用程序

如果您无法访问您创建并部署到 AWS 大型机现代化运行时环境的正在运行 AWS 的大型机现代化应用程序的 URL，则可能需要在与运行时环境关联的安全组上配置入站规则。

常见原因

创建运行时环境时，您提供的安全组（包括默认安全组）必须具有配置为允许流量从 VPC 外部到已部署应用程序的入站规则（如果您想允许此类访问）。

解决方案

检查与运行时环境关联的 Amazon VPC 安全组是否允许流量通过相应的应用程序端口进入该环境。要查看安全组规则，请完成以下步骤：

1. 打开 AWS 大型机现代化控制台，网址为https://console.aws.amazon.com/m2/。
2. 在左侧导航中，选择环境。
3. 选择托管您要连接到的应用程序的运行时环境。
4. 选择配置。
5. 在安全与网络下面，选择安全组。单击链接可在 Amazon VPC 控制台中打开安全组的详细信息。
6. 如有必要，请选择编辑入站规则并添加以下规则（如果尚不存在）：

   Type

      自定义 TCP

端口：

8196 或与应用程序定义中指定的侦听器属性相匹配的端口。有关更多信息，请参阅 步骤 2：创建应用程序定义。

来源

您调用应用程序的 IP 地址。您可以从下拉列表中选择 myIP。如果仍有超时问题，请尝试选择 “任何地方” IPV4 或 “任何地方” IPV6。在安全组上添加入站规则后，请务必停止应用程序并将其重新启动。

有关更多信息，请参阅《Amazon VPC 用户指南》中的使用安全组规则。

故障排除： AWS Blu Insights 无法从主机打开

本页介绍如何解决无法通过 AWS 大型机现代化控制台打开 Blu Insights 页面的问题。

- 引擎： AWS Blu Age
- 组件：Blu Insights

当你尝试从 AWS 大型机现代化控制台访问 Blu Insights 时，它不会打开，新选项卡会立即关闭。

常见原因

您用于访问 Blu Insights 的角色没有足够的权限。

解决方案

将一个 IAM 策略附加到该角色以允许其访问 Blu Insights。请确保该策略至少包含以下权限。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                    "m2:GetSignedBluinsightsUrl"
                ],
                "Resource": "*"
                }
        ]
}
```

请务必用正确的`region`和`account`替换 AWS 区域 和 AWS 账户。

错误故障排除：环境运行不正常

本页介绍当你收到一个 AWS 大型机现代化环境运行状况不佳的通知时，如何解决错误。

- 引擎： AWS Blu Age 和 Rocket Software（以前是
- 组件：环境

如果您收到一条通知，说您的一个 AWS 大型机现代化环境已变得不健康，则这适用于您。您会通过以下来源之一收到通知：

- 不健康的环境状态显示在您的 AWS 大型机现代化控制台中。
- 来自 AWS Health的有关不健康环境状态的电子邮件通知。
- 您可以在 AWS Health 控制面板的 “您的账户运行状况” 下方看到 AWS 大型机现代化的相关事件。

常见原因

当您的 AWS 账户中与 AWS 大型机现代化环境关联的资源无法访问时，就会发生错误。出现此问题的一个常见原因是与该环境相关的资源正在被修改或删除。

解决方案

要获得具体指导，请使用 AWS Health 发送的电子邮件中提供的错误代码或通过您的 AWS Mainframe Modernization 控制台提供的错误代码。

错误代码：

- Storage unreachable

此错误表示环境的附加存储（Amazon Elastic File System 或 Amazon FSx 文件系统）无法正确安装。要查看运行不正常的环境的详细信息，请完成以下步骤：

1. 打开 AWS 大型机现代化控制台，网址为https://console.aws.amazon.com/m2/。
2. 选择运行不正常的环境，然后选择配置。
3. 选择附加的存储，以查看与此环境关联的存储资源。
4. 检查与网络相关的配置，例如与该存储关联的安全组、子网和 Amazon VPC。如果这些配置不正确，请尝试通过恢复这些配置来解决此问题。

> Note
>
> 如果存储已删除，则无法恢复该环境。在此情况下，您应考虑删除运行不正常的环境。

Rocket Software（前身为 Micro Focu

本页介绍如何使用 Rocket Software 运行时引擎解决许可证问题

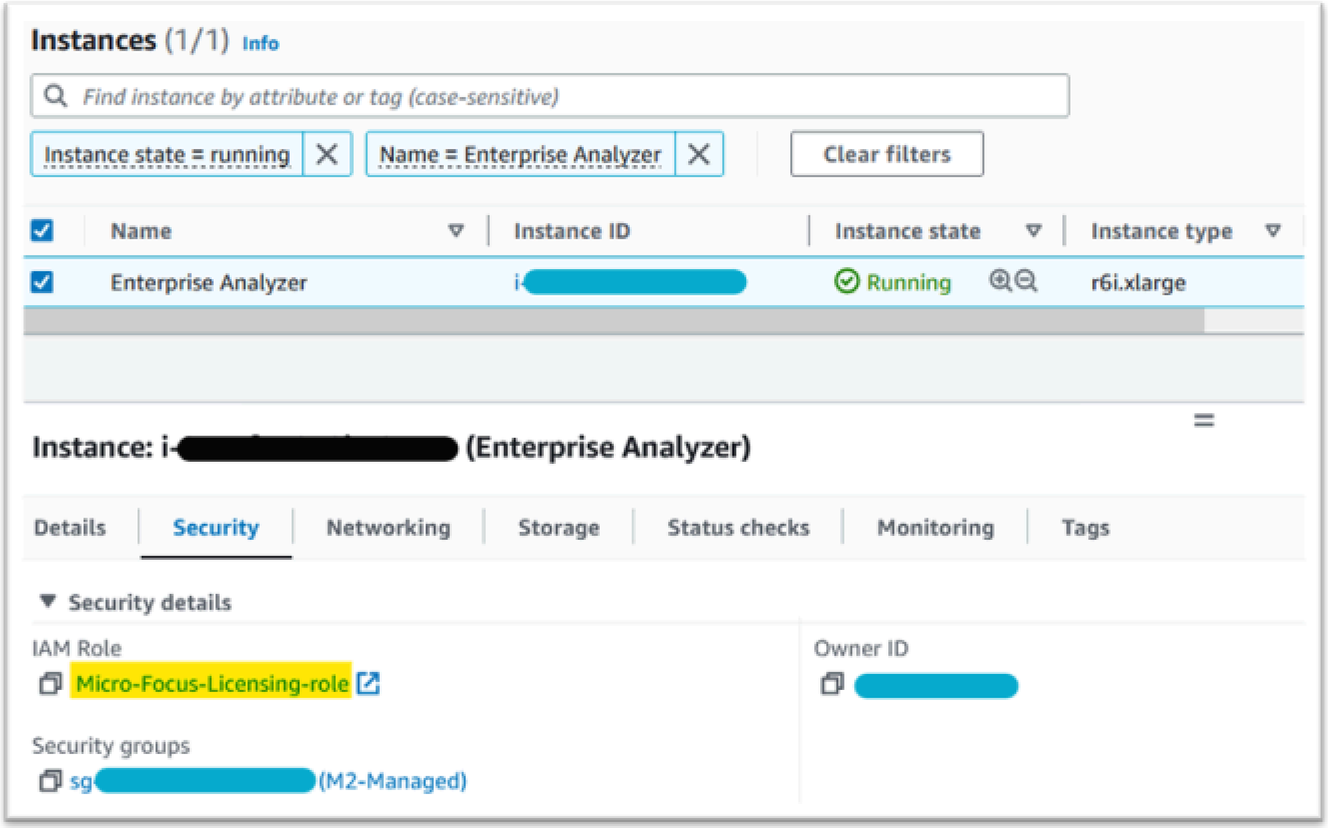
- 引擎：火箭软件
- 组件：亚马逊 EC2

如果您在访问或使用时遇到问题 AMIs，以下信息可能会对您有所帮助。

主题

验证 Amazon EC2 实例是否具有 IAM 许可角色

可以在 Amazon EC2 实例详情的 “安全” 选项卡上进行检查。可以使用操作下拉菜单的“安全选项”进行更改。



使用 Reachability Analyzer

在控制台页面上找到 Reachability Analyzer。 AWS Network Manager

创建并分析从 AMI 创建的亚马逊 EC2 实例和 Amazon S3 VPC 终端节点之间的路径。

如果 Amazon EC2 实例无法访问互联网，请对所有 4 个终端节点重复路径分析。

有关 Reachability Analyzer 的更多信息，请参阅《Reachability Analyzer 指南》中的开始使用 Reachability Analyzer。

运行 license-daemon

在 Windows Enterprise Developer 中，从命令提示符下使用以下命令：

```
“C:\Program Files (x86)\Micro Focus\Enterprise Developer\AdoptOpenJDK\bin\java” -jar
 "C:\Program Files (x86)\Micro Focus\Licensing\aws-license-daemon.jar"
```

然后，查看输出。忽略 SLF4 J 消息并查找第一个异常。

在 Enterprise Analyzer 中，从命令提示符下使用以下命令：

```
"C:\Program Files (x86)\Micro Focus\AdoptOpenJDK\bin\java" -jar "C:\Program Files
 (x86)\Micro Focus\Licensing\aws-license-daemon.jar"
```

然后，查看输出。忽略 SLF4 J 消息并查找第一个异常。

在 Linux 上，运行：

```
java -jar /var/microfocuslicensing/bin/aws-license-daemon.jar
```

忽略 SLF4 J 消息并查找第一个异常。

例如，如果 Amazon S3 资源不可用，则异常如下所示：

```

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
```

异常消息会指示哪个资源不可用。将配置值与本主题中显示的值进行比较。

执行操作系统修补后，Linux 上的 Enterprise Server 或企业构建工具出现许可证问题

如果执行操作系统修补后 Linux 上的 Enterprise Server 或企业构建工具出现许可证问题，请下载并运行补丁脚本来更新许可证进程守护程序。要执行此操作，请在命令提示下运行以下命令：

```
sudo curl https://d148y999krizvm.cloudfront.net/patch/v8/linux/patch.sh -o /var/
microfocuslicensing/bin/patch.sh
sudo chmod +x /var/microfocuslicensing/bin/patch.sh
sudo /var/microfocuslicensing/bin/patch.sh
sudo ./startmfcesd.sh
```

> **Note**
>
> 即使下载路径是版本 8 的路径，此补丁脚本也适用于版本 9。

《AWS 大型机现代化用户指南》的文档历史记录

下表描述了 AWS 大型机现代化的文档版本。

变更	说明	日期
AWS 从 2025 年 11 月 7 日起，大型机现代化将不再向新客户开放。	AWS 从 2025 年 11 月 7 日起，大型机现代化将不再向新客户开放。如果您想使用 AWS 大型机现代化服务，请在该日期之前注册。现有客户可以继续照常使用该服务。有关更多信息，请参阅AWS 大型机现代化可用性变更。	2025年10月7日
AWS Blu Age 发行说明 4.7.0 和 4.8.0	4.7.0 版本的 Blu Age Runtime 和 AWS Blu Age Transform ation Engines 包括对 Spring AWS Boot 3.4.2 和 Angular 19 的核心依赖项升级，在实现前端应用程序框架现代化的同时增强了安全性和性能。4.8.0 版本包括关键功能，例如 IBM IMS 数据库转换支持和一项新的内存缓存功能，该功能可提高数据密集型程序的性能，以实现 00 现代化。 AS4	2025年5月18日
AWS Blu Age 发行说明 4.6.0	此版本的 AWS Blu Age Runtime 和 AWS Blu Age 转换引擎侧重于 zoS 和 AS4 00 的新功能和改进。	2025 年 1 月 24 日
AWS Blu Age 发行说明 4.5.0	此版本的 AWS Blu Age Runtime 和 AWS Blu Age 转换引擎侧重于 JCL 支持的主要功能、对 AS/400 现代化应用程	2024 年 12 月 20 日

	序的绑定目录和激活组支持以及更新的依赖关系。	
AWS 蓝光时代常见问题解答	通过这份全面的列表，了解 AWS Blu Age 的重构功能。FAQs	2024 年 12 月 20 日
Amazon Q 开发者为大型机进行转型	您可以了解 Amazon Q Developer 的大型机转换功能，该功能使您能够将传统 COBOL 大型机应用程序现代化为 Java 应用程序。	2024 年 12 月 2 日
AWS Blu Age 发行说明 4.4.0	此版本的 AWS Blu Age Runtime 和转换引擎侧重于升级关键依赖项和支持的技术，同时提高多种功能的性能。	2024 年 11 月 13 日
在容器上部署 AWS 蓝光时代运行时	你可以学习如何在容器上设置 AWS Blu Age Runtime，将其部署在亚马逊 ECS（由亚马逊管理 EC2或 AWS Fargate）和由亚马逊管理的 Amazon EKS 上 EC2。	2024 年 10 月 28 日
更新了 LDAP 用户凭证	现在，您可以使用 AWS 控制台或 AWS CLI （或 SDK）创建和管理 LDAP 用户凭据进行身份验证和授权。	2024 年 10 月 21 日
配置 Rocket 软件托管应用程序	现在，您可以使用 Rocket Software 运行时引擎配置应用程序，以自定义其他属性，包括集成。	2024 年 10 月 21 日

使用 NTT DATA Unikix 运行时进行平台改造	了解如何使用 Amazon Machine Images (AMIs) 创建量身定制的环境，以便使用 NTT DATA 在中 AWS 重新托管和平台大型机应用程序。	2024 年 9 月 25 日
AWS 大型机现代化应用程序测试 IAM	您可以了解如何使用可用的 IAM 功能和策略来管理 AWS 大型机现代化应用程序测试的访问权限。	2024 年 9 月 20 日
AWS Blu Age 发布	AWS 大型机现代化用户指南的这一部分包含了 AWS 蓝光时代的所有版本控制细节、Bl AWS u Age 发行说明、 AWS Blu Age 升级说明以及整个 AWS 蓝光时代生命周期。	2024 年 9 月 16 日
AWS 大型机现代化组件生命周期	本页记录了每个 AWS 大型机现代化组件的生命周期，包括其版本升级、总体发布计划以及终止支持和停用计划。	2024 年 9 月 5 日
采用 mLogica 的汇编器转换	AWS 使用 mLogica 进行大型机现代化代码转换是一项 AWS 大型机现代化功能，可自动将 z/OS 大型机汇编器代码转换为 COBOL。	2024 年 7 月 22 日
应用程序测试功能 GA 版本	应用程序测试的通用可用性文档。 AWS 大型机现代化应用程序测试为您的迁移项目提供自动功能等效性测试。此版本包含了数据保护页面、控制台工作流程以及自预览版以来对其他文档页面的更新。	2024 年 6 月 12 日

更新了 Rocket 软件的托管运行时教程	本教程介绍如何使用 Rocket Software 运行时引擎在 AWS 大型机现代化托管运行时环境中部署和运行 CardDemo 示例应用程序。	2024 年 2 月 5 日
AWS Blu Age 运行时和现代化工具 3.9.0 版本的发布说明。	此版本的 AWS Blu Age Runtime 和现代化工具侧重于整个产品的多项横向增强，旨在提高高可用性架构的性能，以及将任务执行提升到更高水平的新功能。	2023 年 12 月 18 日
在大型机与主机之间传输文件 AWS	发布了将文件从源大型机传输到 AWS的新功能。	2023 年 11 月 27 日
管理应用程序事务	发布的新特征，用于显示和编辑 AWS Mainframe Modernization 应用程序事务。	2023 年 10 月 16 日
AWS Blu Age 运行时和现代化工具 3.6.0 版本的发布说明。	此版本的 AWS Blu Age 运行时和现代化工具为 zOS 和 AS400 传统迁移提供了新功能，主要面向扩展 CICS 支持机制、补充 JCL 功能、优化并发和大容量功能的性能以及添加功能。 multi-data-source	2023 年 8 月 4 日
现在，您可以在应用程序停止时部署其新版本。	以前，必须删除已部署的应用程序版本才能部署其新版本。现在，您只需停止已部署的版本即可部署新版本。	2023 年 7 月 26 日

AWS Blu Age 运行时打包便于亚马逊 EC2 部署	AWS 借助 AWS Blu Age 运行时进行大型机现代化改造，现在可以更灵活地配置完整的堆栈并在您 AWS 账户的 Amazon EC2 实例上进行部署。	2023 年 7 月 6 日
单点登录 Blu Age AWS Blu Insights	AWS Blu Age Blu Insights 可 AWS 管理控制台 通过单点登录获得。	2023 年 3 月 31 日
GA 版本	GA 发布了《AWS 大型机现代化用户指南》。	2022 年 6 月 8 日
初始版本	《AWS 大型机现代化用户指南》的初始版本（公开预览版）。	2021 年 11 月 30 日