实施指南

AWS 上的分布式负载测试

AWS 上的分布式负载测试: 实施指南

Table of Contents

自动对您的软件应用程序进行大规模测试

发布日期：2025 年 11 月

AWS 上的分布式负载测试可帮助您自动对软件应用程序进行大规模性能测试，以便在发布应用程序之前识别瓶颈。该解决方案模拟成千上万的连接用户以持续的速率生成 HTTP 请求，而无需配置服务器。

该解决方案利用 AWS Fargate 上的亚马逊弹性容器服务 (Amazon ECS) 来部署运行您的负载测试模拟的容器，并提供以下功能：

- 在独立运行的 AWS Fargate 容器上部署 Amazon ECS，以测试应用程序的负载容量。
- 模拟多个 AWS 区域中成千上万的并发用户以持续的速度生成请求。
- 使用 K6 JMeter、Locust 测试脚本或简单的 HTTP 端点配置自定义应用程序测试。
- 将负载测试安排为立即运行、在 future 的某个日期和时间运行，或者定期运行。
- 在不同的场景和地区同时运行多个负载测试。

本实施指南概述了 AWS 上的分布式负载测试解决方案、其参考架构和组件、部署规划注意事项以及将该解决方案部署到 Amazon Web Services (AWS) 云的配置步骤。它包括指向 A WS CloudFormation 模板的链接，该模板使用安全性和可用性方面的 AWS 最佳实践启动和配置部署此解决方案所需的 AWS 服务。

在其环境中使用此解决方案的特性和功能的目标受众包括在 AWS 云中具有架构实践经验的 IT 基础设施架构师、管理员和 DevOps 专业人士。

使用以下导航表可快速找到这些问题的答案：

如果您想. . .	阅读. . .
了解运行此解决方案的成本。 在美国东部（弗吉尼亚北部）地区运行此解决方案的 AWS 资源费用估计为每月 30.90 美元。	成本
了解此解决方案的安全注意事项。	安全性
了解如何为此解决方案规划限额。	配额

如果您想. . .	阅读. . .
了解哪些 AWS 区域支持此解决方案。	支持的 AWS 区域
了解用于人工智能辅助负载测试分析的可选 MCP 服务器。	MCP 服务器集成
查看或下载此解决方案中包含的 AWS CloudFormation 模板，以自动部署该解决方案的基础设施资源（“堆栈”）。	AWS CloudFormation 模板
访问源代码，也可以选择使用 AWS Cloud Development Kit (AWS CDK) 来部署解决方案。	GitHub 存储库

特征

此解决方案提供以下功能：

多测试框架 Support

支持 JMeter K6 和 Locust 测试脚本，以及无需自定义脚本即可进行简单的 HTTP 端点测试。有关更多信息，请参阅架构详细信息部分中的测试类型。

高用户负荷模拟

模拟成千上万的并发虚拟用户，在真实的负载条件下对您的应用程序进行压力测试。

多区域负载分布

将负载测试分发到多个 AWS 区域，以模拟地理分布的用户流量并评估全球性能。

灵活的测试计划

使用自动回归测试的 cron 表达式将测试安排为立即运行、在特定的未来日期和时间运行，或者定期运行。

实时监控

提供可选的实时数据流，通过响应时间、虚拟用户数量和请求成功率等实时指标来监控测试进度。

全面的测试结果

显示详细的测试结果，包括性能指标、百分位数（p50、p90、p95、p99）、错误分析和可供离线分析的可下载工件。

基线比较

指定用于性能比较的基准测试运行，以跟踪一段时间内的改进或回归。

端点灵活性

在 AWS 区域、本地环境或其他云提供商之间测试任何 HTTP 或 HTTPS 终端节点。

直观的 Web 控制台

提供基于 Web 的控制台，无需命令行交互即可创建、管理和监控测试。

AI 辅助分析（可选）

通过模型上下文协议 (MCP) 服务器与 AI 开发工具集成，对负载测试数据进行智能分析。

多协议 Support

通过自定义测试脚本支持各种协议，包括 HTTP WebSocket、HTTPS、JDBC、JMS、FTP 和 gRPC。

优势

该解决方案具有以下优点：

全面的性能测试

支持负载测试、压力测试和耐久性测试，以全面评估各种条件下的应用性能。

早期问题检测

在生产部署之前识别性能瓶颈、内存泄漏和可扩展性问题，从而降低停机风险。

真实世界的使用模拟

准确模拟真实世界的用户行为和流量模式，以验证应用程序在真实条件下的性能。

切实可行的性能 Insights

提供详细的指标、百分位数和错误分析，以了解应用程序行为并指导优化工作。

自动测试工作流程

支持定期和定期测试，无需人工干预即可进行持续性能监控和回归测试。

具有成本效益的基础架构

使用 pay-per-use具有定价功能的无服务器 AWS Fargate 容器，无需专门的测试基础设施和持续的订阅费。

快速测试部署

无需预置或管理服务器，即可在几分钟内部署和扩展测试基础架构。

轻松查询测试结果

通过可选的模型上下文协议 (MCP) 服务器与 AI 开发工具集成，支持自然语言查询和对负载测试数据的智能分析，从而更快地获得见解和故障排除。

使用案例

生产前验证

在启动新版本之前，在类似生产的负载条件下测试 Web 和移动应用程序，以验证性能并发现问题。

容量规划

确定您的应用程序在当前基础架构下可以支持的最大并发用户数，并确定何时需要扩展。

峰值负载验证

验证您的基础设施是否能够应对高峰负载、季节性流量峰值或意外需求激增而不会降低性能。

性能优化

确定性能瓶颈，例如数据库查询缓慢、代码执行效率低下、网络延迟或资源限制。

回归测试

安排定期负载测试，以检测新代码部署或基础架构变更带来的性能下降。

全球绩效评估

评估来自多个地理区域的应用程序性能，以确保为全球受众提供一致的用户体验。

API 负载测试

测试 REST APIs、GraphQL 端点或微服务，以验证负载下的响应时间、吞吐量和错误率。

CI/CD 管道集成

将自动性能测试集成到持续集成和部署管道中，以便在开发周期的早期发现性能问题。

第三方服务测试

在各种负载条件下，测试您的应用程序所依赖的第三方 APIs 或服务的性能和可靠性。

概念和定义

本节介绍重要概念并定义此解决方案特有的术语：

场景

测试定义，包括测试名称、描述、任务计数、并发性、AWS 区域、ramp-up、hold-for、测试类型、计划日期和重复配置。

任务计数

将在 Fargate 集群中启动以运行测试场景的容器数量。一旦达到 Fargate 资源的账户限制，就不会创建其他任务。但是，已经在运行的任务将继续进行。

concurrency

并发度（每个任务的并发虚拟用户数）。基于默认设置的推荐并发度为 200。并发受到 CPU 和内存的限制。对于基于 Apache 的测试 JMeter，较高的并发度会增加 JVM 在 ECS 任务上使用的内存。默认 ECS 任务定义创建内存为 4 GB 的任务。建议从 1 个任务的较低并发值开始，然后监控任务集群的 ECS CloudWatch 指标。请参阅 A mazon ECS 集群利用率指标。

rampup

从零逐渐增加到目标并发级别的时间段。

坚持下去

扩容完成后保持目标并发水平的时间段。

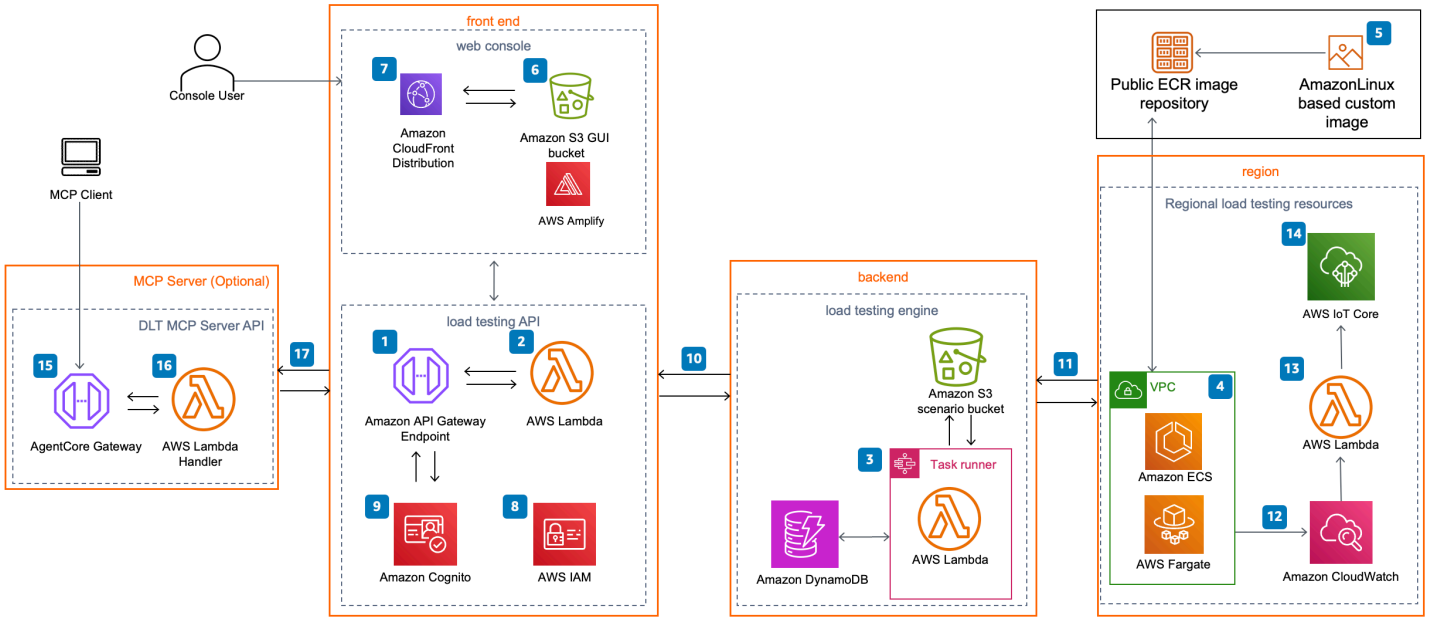
有关 AWS 术语的一般参考，请参阅 AWS 术语表。

架构概述

架构图

使用默认参数部署此解决方案将在您的 AWS 账户中部署以下组件。

在 AWS 上的 AWS 架构上进行分布式负载测试



> **Note**
>
> AWS CloudFormation 资源是基于 AWS Cloud Development Kit (AWS CDK) 结构创建的。

使用 AWS CloudFormation 模板部署的解决方案组件的高级流程如下：

1. 分布式负载测试器 API 利用 Amazon API Gateway 来调用解决方案的微服务（AWS Lambda 函数）。
2. 微服务提供了用于管理测试数据和运行测试的业务逻辑。
3. 这些微服务与亚马逊简单存储服务 (Amazon S3)、Amazon DynamoD B 和 AWS Step Fun ctions 交互，以存储测试场景的详细信息和结果并协调测试执行。

4. 部署的亚马逊虚拟私有云（亚马逊 VPC）网络拓扑结构包含在 AWS Farg ate 上运行的该解决方案的亚马逊弹性容器服务 (Amazon ECS) 容器。
5. 这些容器使用安装了 Ta ur us 负载测试框架的 Amazon Linux 2023 基础镜像。Taurus 是一个开源测试自动化框架 JMeter，支持 K6、Locust 和其他测试工具。容器镜像符合开放容器计划 (OCI) 标准，由 AWS 托管在亚马逊弹性容器注册表 (Amazon ECR) 公共存储库中。有关更多信息，请参阅容器镜像自定义。
6. 由 AWS Amplify 提供支持的网络控制台部署到为静态虚拟主机配置的 S3 存储桶中。
7. Amazon CloudFront 提供对解决方案网站存储桶内容的安全公开访问权限。
8. 在初始配置期间，该解决方案会创建默认管理员角色（IAM 角色），并向客户指定的用户电子邮件地址发送访问邀请。
9. A mazon Cognito 用户池管理用户对控制台、分布式负载测试器 API 和 MCP 服务器的访问权限。
10. 部署此解决方案后，您可以使用 Web 控制台或 APIs 创建和运行定义一系列任务的测试方案。
11. 微服务使用此测试场景在指定区域的 Fargate 上运行 ECS 任务。
12. 测试完成后，该解决方案会将结果存储在 S3 和 DynamoDB 中，并将输出记录在亚马逊中。CloudWatch
13. 如果您启用实时数据选项，则该解决方案将在测试期间将运行测试的每个区域的 Fargate 任务中的 CloudWatch 日志发送到 Lambda 函数。
14. Lambda 函数将数据发布到部署主堆栈所在区域的 AWS IoT Core 中的相应主题。Web 控制台订阅主题并在测试运行时显示实时数据。

> **Note**
>
> 以下步骤描述了用于人工智能辅助负载测试分析的可选 MCP 服务器集成。只有在解决方案部署期间选择 MCP 服务器选项，才会部署此组件。

15. MCP 客户端（AI 开发工具）连接到 AWS AgentCore Gateway 终端节点，通过模型上下文协议访问分布式负载测试解决方案的数据。 AgentCore Gateway 会验证用户的 Cognito 身份验证令牌，以确保对 MCP 服务器的授权访问。
16. 成功进行身份验证后， AgentCore Gateway 会将 MCP 工具请求转发给 DLT MCP 服务器 Lambda 函数。Lambda 函数将结构化数据返回给 Gatewa AgentCore y，网关将其发送回 MCP 客户端，以进行人工智能辅助的分析和见解。
17. Lambda 函数处理请求并查询相应的 AWS 资源（DynamoDB 表、S3 存储桶或 CloudWatch 日志），以检索请求的负载测试数据。

AWS Well-Architected 设计注意事项

该解决方案采用 AWS Well-Architected Fram ework 中的最佳实践，可帮助客户在云中设计和运行可靠、安全、高效且经济实惠的工作负载。

本节介绍 Well-Architected Framework 的设计原则和最佳实践如何使该解决方案受益。

卓越运营

本节介绍我们是如何使用卓越运营支柱的原则和最佳实践来设计此解决方案的。

- 使用从 AWS CDK 结构生成的 AWS CloudFormation 模板将所有资源定义为基础设施即代码。
- 该解决方案将指标推送到各个 CloudWatch 阶段，以提供对 Lambda 函数、ECS 任务、S3 存储桶和其他解决方案组件的可观察性。

安全性

本节介绍我们是如何使用安全性支柱的原则和最佳实践来设计此解决方案的。

- Cognito 对 Web 控制台用户和 API 请求进行身份验证和授权。
- 所有服务间通信都使用访问权限最低的 AWS Identity and AWS Identity and Access Managem ent (IAM) 角色，仅包含所需的最低权限。
- 所有数据存储，包括 S3 存储桶和 DynamoDB 表，都使用 AWS 托管密钥对静态数据进行加密。
- 出于审计和合规目的，如果适用，则启用日志记录、跟踪和版本控制。
- 默认情况下，网络访问是私有的，启用 VPC 终端节点以保持流量在 AWS 网络内。

> **Note**
>
> 该解决方案根据 CloudWatch 日志量和成本考虑因素创建多个保留期不同的日志组：

日志类型	保留周期
ECS 容器见解	1 天
Step Functions、ECS 自定义日志、API Gateway 访问日志	1 年

日志类型	保留周期
Lambda 运行时日志	2 年
API Gateway 执行日志	永不过期

您可以根据需要在 CloudWatch 控制台中修改这些保留期。

可靠性

本节介绍我们是如何使用可靠性支柱的原则和最佳实践来设计此解决方案的。

- 该解决方案尽可能使用 AWS 无服务器服务（例如：Lambda、API Gateway、Amazon S3、AWS Step Functions、Amazon DynamoDB 和 AWS Fargate）来确保高可用性并从服务故障中恢复。
- 所有计算处理都使用 Lambda 函数或 AWS Fargate 上的 Amazon ECS。
- 数据存储在 DynamoDB 和 Amazon S3 中，因此默认情况下数据会保留在多个可用区中。

性能效率

本节介绍我们是如何使用性能效率支柱的原则和最佳实践来设计此解决方案的。

- 该解决方案使用无服务器架构，能够根据需要进行水平扩展。
- 该解决方案可以在支持本解决方案中的 AWS 服务的任何区域启动，例如：AWS Lambda、Amazon API Gateway、Amazon S3、AWS Step Functions、亚马逊 DynamoDB、亚马逊 ECS、AWS Fargate 和 Amazon Cognito。
- 该解决方案自始至终都使用托管服务，以减轻资源配置和管理的运营负担。
- 该解决方案每天都会自动测试和部署，以实现随着 AWS 服务的变化保持一致性，并由解决方案架构师和主题专家对需要实验和改进的领域进行审查。

成本优化

本节介绍我们是如何使用成本优化支柱的原则和最佳实践来设计此解决方案的。

- 该解决方案使用无服务器架构；因此，客户只需为其使用量付费。

- Amazon DynamoDB 可按需扩展容量，因此您只需为使用的容量付费。
- AWS Fargate 上的 AWS ECS 允许您仅为使用的计算资源付费，无需支付任何前期费用。
- AWS AgentCore Gateway 可作为分布式负载测试 API 的经济高效的基于 Lambda 的代理，无需使用专用基础设施，并通过无服务器 pay-per-request定价降低成本。

可持续性

本节介绍我们是如何使用可持续性支柱的原则和最佳实践来设计此解决方案的。

- 与持续运行本地服务相比，该解决方案使用托管无服务器服务来最大限度地减少后端服务对环境的影响。
- 无服务器服务允许您根据需求扩展或缩减规模。

架构详情

本节介绍构成此解决方案的组件和 AWS 服务，以及这些组件如何协同工作的架构详情。

AWS 上的分布式负载测试解决方案由三个高级组件组成：前端、后端和可选的 MCP 服务器。

前端

前端提供与解决方案交互的接口，包括：

- 用于编程访问的负载测试 API
- 用于创建、安排和运行性能测试的 Web 控制台
- 可选的 MCP 服务器，用于 AI 辅助分析测试结果和错误

负载测试 API

AWS 上的分布式负载测试将 Amazon API Gateway 配置为托管解决方案的 RESTful API。用户可以通过随附的 Web 控制台、 RESTful API 和可选的 MCP 服务器与负载测试系统进行安全交互。该 API 充当访问存储在 Amazon DynamoDB 中的测试数据的 “前门”。您还可以使用 APIs 访问您在解决方案中内置的任何扩展功能。

此解决方案利用了 Amazon Cognito 用户池的用户身份验证功能。成功对用户进行身份验证后，Amazon Cognito 会发布一个 JSON 网络令牌，该令牌用于允许控制台向解决方案的（Amazon API Gat APIs eway 终端节点）提交请求。HTTPS 请求由控制台发送到包含令牌的授权标头。 APIs

根据请求，API Gateway 调用相应的 AWS Lambda 函数对存储在 DynamoDB 表中的数据执行必要的任务，将测试场景作为 JSON 对象存储在亚马逊 S3 中，检索 CloudWatch 亚马逊指标图像，并将测试场景提交到 AWS Step Functions 状态机。

有关解决方案 API 的更多信息，请参阅本指南的分布式负载测试 API 部分。

Web 控制台

此解决方案包括一个 Web 控制台，可用于配置和运行测试、监控正在运行的测试以及查看详细的测试结果。该控制台是使用 Cloudscape 构建的 ReactJS 应用程序，Cloudscape 是一个用于构建直观的 Web 应用程序的开源设计系统。该控制台托管在 Amazon S3 中，可通过亚马逊进行访问

CloudFront。该应用程序利用 AWS Amplify 与 Amazon Cognito 集成来对用户进行身份验证。Web 控制台还包含一个选项，用于查看正在运行的测试的实时数据，在该选项中，它可以订阅 AWS IoT Core 中的相应主题。

Web 控制台 URL 是 CloudFront 分发域名，可以在 CloudFormation 输出中作为控制台找到。启动 CloudFormation 模板后，您还将收到一封电子邮件，其中包含 Web 控制台 URL 和登录该模板的一次性密码。

MCP 服务器（可选）

可选的模型上下文协议 (MCP) 服务器为 AI 开发工具提供了一个额外的接口，用于通过自然语言交互访问和分析负载测试数据。只有在解决方案部署期间选择 MCP 服务器选项，才会部署此组件。

MCP 服务器使 AI 代理能够使用 Amazon Q、Claude 和其他兼容 MCP 的人工智能助手等工具查询测试结果、分析性能指标并深入了解您的负载测试数据。有关 MCP 服务器架构和配置的详细信息，请参阅本节中的 MCP 服务器。

Backend

后端由容器镜像管道和用于生成测试负载的负载测试引擎组成。你通过前端与后端进行交互。此外，为每次测试启动的 AWS Fargate 上的 Amazon ECS 任务都标有唯一的测试标识符 (ID)。这些测试 ID 标签可用于帮助您监控此解决方案的成本。有关更多信息，请参阅 AWS Billing and Cost Management 用户指南中的用户定义成本分配标签。

容器镜像管道

此解决方案使用使用 A mazon Linux 2023 构建的容器映像作为安装了 Ta ur us 负载测试框架的基础映像。Taurus 是一个开源测试自动化框架 JMeter，支持 K6、Locust 和其他测试工具。AWS 将此映像托管在亚马逊弹性容器注册表 (Amazon ECR) 的 Elastic Registry 公共存储库中。该解决方案使用此镜像在 AWS Fargate 集群上的 Amazon ECS 中运行任务。

有关更多信息，请参阅本指南的容器镜像自定义部分。

测试基础架构

除了主 CloudFormation 模板外，该解决方案还提供了一个区域模板，用于启动在多个区域运行测试所需的资源。该解决方案将此模板存储在 Amazon S3 中，并在 Web 控制台中提供指向该模板的链接。每个区域堆栈都包括一个 VPC、一个 AWS Fargate 集群和一个用于处理实时数据的 Lambda 函数。

有关如何在其他区域部署测试基础设施的更多信息，请参阅本指南的多区域部署部分。

负载测试引擎

分布式负载测试解决方案使用亚马逊弹性容器服务 (Amazon ECS) 和 AWS Fargate 来模拟多个区域的数千名并发用户，以持续的速度生成 HTTP 请求。

您可以使用随附的 Web 控制台定义测试参数。该解决方案使用这些参数生成 JSON 测试场景并将其存储在 Amazon S3 中。有关测试脚本和测试参数的更多信息，请参阅本节中的测试类型。

AWS Step Functions 状态机在 AWS Fargate 集群中运行和监控 AWS ECS 任务。AWS Step Functions 状态机包括一个 ecr-checker AWS Lambda 函数、一个 AWS Lambda 函数、一个任务运行器 task-status-checker AWS Lambda 函数、一个任务取消器 AWS Lambda 函数和一个结果解析器 AWS Lambda 函数。有关工作流程的更多信息，请参阅本指南的测试工作流程部分。有关测试结果的更多信息，请参阅本指南的测试结果部分。有关取消考试工作流程的更多信息，请参阅本指南的 “取消考试工作流程” 部分。

如果您选择实时数据，则该解决方案将通过 CloudWatch 与该区域中的 Fargate 任务对应的日志在每个区域中启动 Lamb real-time-data-publisher da 函数。然后，该解决方案会处理数据并将其发布到您启动主堆栈的区域内的 AWS IoT Core 中的某个主题。有关更多信息，请参阅本指南的实时数据部分。

MCP 服务器

可选的模型上下文协议 (MCP) 服务器集成使 AI 代理能够通过自然语言交互以编程方式访问和分析您的负载测试数据。只有在解决方案部署期间选择 MCP 服务器选项，才会部署此组件。

MCP 服务器充当 AI 开发工具和 DLT 部署之间的桥梁，为性能测试结果的智能分析提供了标准化接口。该架构集成了多个 AWS 服务，为 AI 代理交互创建了一个安全、可扩展的接口：

AWS AgentCore 网关

AWS AgentCore Gateway 是一项完全托管的服务，可为 MCP 服务器提供标准化的托管和协议管理。在此解决方案中， AgentCore Gateway 充当 AI 代理在请求访问您的负载测试数据时连接到的公共端点。

该服务处理所有 MCP 协议通信，包括工具发现、身份验证令牌验证和请求路由。 AgentCore Gateway 作为多租户服务运行，具有内置安全保护功能，可抵御公共端点面临的常见威胁，同时验证每个请求的 Cognito 令牌签名和声明。

DLT MCP 服务器 Lambda

DLT MCP 服务器 Lambda 函数是一个自定义的无服务器组件，用于处理来自 AI 代理的 MCP 请求并将其转换为针对您的 DLT 资源的查询。

此 Lambda 函数充当 MCP 集成的智能层，可从 DynamoDB 表中检索测试结果，访问存储在 S3 存储桶中的性能项目，并查询日志以获取详细的执行信息。 CloudWatch Lambda 函数实现只读访问模式，并将原始 DLT 数据转换为结构化的、人工智能友好的格式，代理可以轻松解释和分析。

身份验证集成

身份验证系统利用您现有的 Cognito 用户池基础架构，在 Web 控制台和 MCP 服务器界面上保持一致的访问控制。

此集成使用基于 OAuth 2.0 令牌的身份验证。用户通过 Cognito 登录过程进行一次身份验证，即可获得用于用户界面交互和 MCP 服务器访问的令牌。该系统保持与 Web 界面相同的权限边界和访问控制，确保用户只能通过 AI 代理访问他们可以通过控制台访问的负载测试数据。

此解决方案中的 AWS 服务

此解决方案中包含以下 AWS 服务：

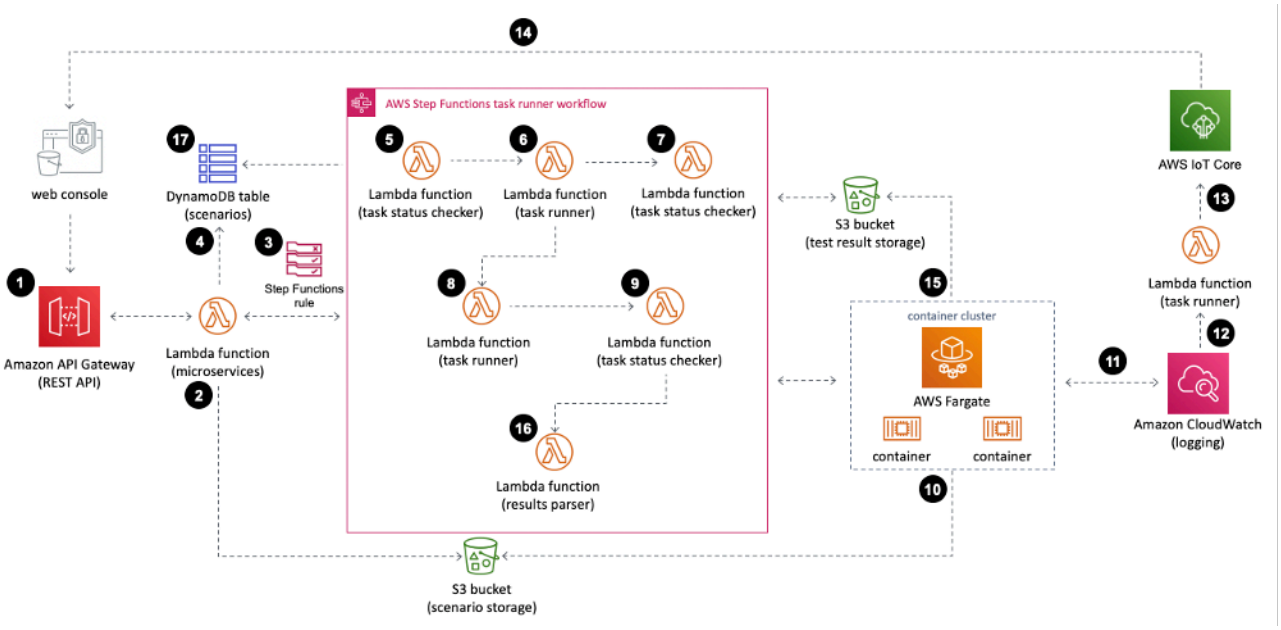
AWS 服务	描述
Amazon API Gateway	核心。在解决方案中托管 REST API 端点。
AWS CloudFormation	核心。管理解决方案基础架构的部署。
Amazon CloudFront	核心。提供托管在 Amazon S3 中的网络内容。
Amazon CloudWatch	核心。存储解决方案日志和指标。
Amazon Cognito	核心。处理 API 的用户管理和身份验证。
Amazon DynamoDB	核心。存储部署信息并测试场景详细信息和结果。
Amazon Elastic Container Service	核心。在 AWS Fargate 容器上部署和管理独立的 Amazon ECS 任务。
AWS Fargate	核心。托管解决方案的 Amazon ECS 容器

AWS 服务	描述
AWS 身份和访问管理 AWS Identity Access	核心。处理用户角色和权限管理。
AWS Lambda	核心。为 APIs 实施、测试结果解析和启动 workers/leader 任务提供逻辑。
AWS Step Functions	核心。为指定区域的 AWS Fargate 任务编排 Amazon ECS 容器的预配置
AWS Amplify	支持。提供由 AW S Ampli fy 提供支持的网络控制台。
亚马逊 CloudWatch 活动	支持。将测试安排为在指定日期或重复日期自动开始。
Amazon Elastic Container Registry	支持。将容器镜像托管在公共 ECR 存储库中。
AWS IoT Core	支持。订阅 AWS IoT Core 中的相应主题，即可查看正在运行的测试的实时数据。
AWS Systems Manager	支持。提供应用程序级资源监控，并可视化资源操作和成本数据。
Amazon S3	支持。托管静态 Web 内容、日志、指标和测试数据。
Amazon Virtual Private Cloud	支持。包含在 AWS Fargate 上运行的解决方案的 Amazon ECS 容器。
Amazon Bedrock AgentCore	支持，可选。托管解决方案的可选远程模型上下文协议 (MCP) 服务器，用于 AI 代理与 API 集成。

AWS 上的分布式负载测试的工作原理

以下详细分解显示了运行测试场景所涉及的步骤。

测试工作流



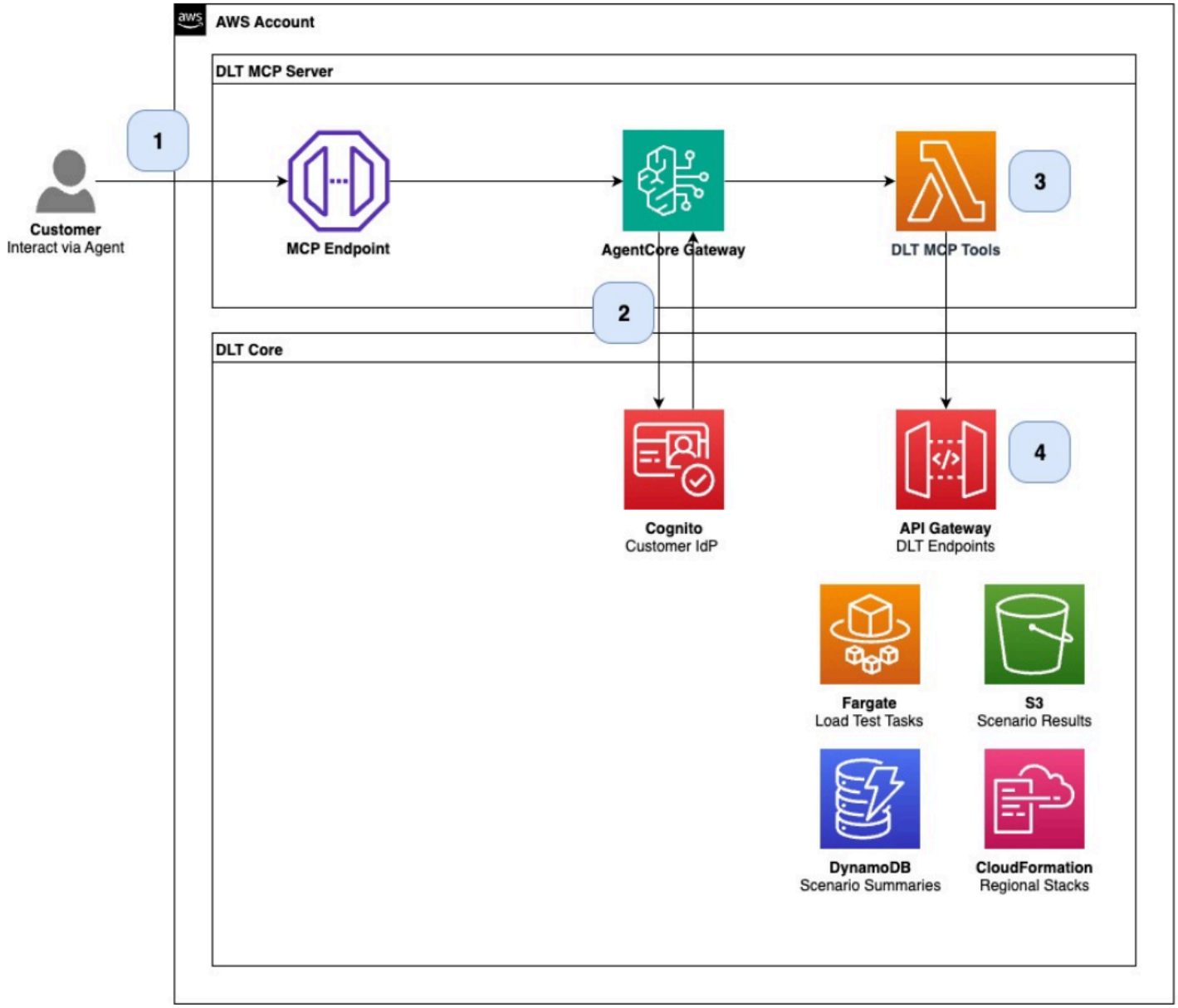
1. 您可以使用 Web 控制台向解决方案的 API 提交包含配置详细信息的测试场景。
2. 测试场景配置以 JSON 文件 `s3://<bucket-name>/test-scenarios/<$TEST_ID>/<$TEST_ID>.json` () 的形式上传到亚马逊简单存储服务 (Amazon S3)。
3. AWS Step Functions 状态机使用测试 ID、任务计数、测试类型和文件类型作为 AWS Step Functions 状态机输入来运行。如果已安排测试，它将首先创建一个 CloudWatch 事件规则，该规则将在指定日期触发 AWS Step Functions。有关计划工作流程的更多详细信息，请参阅本指南的测试计划工作流程部分。
4. 配置详细信息存储在场景 Amazon DynamoDB 表中。
5. 在 AWS Step Functions 任务运行器工作流程中， task-status-checkerAWS Lambda 函数检查亚马逊弹性容器服务 (Amazon ECS) 任务是否已经在使用相同的测试 ID 运行。如果发现具有相同测试 ID 的任务正在运行，则会导致错误。如果 AWS Fargate 集群中没有运行 Amazon ECS 任务，则该函数将返回测试 ID、任务计数和测试类型。
6. 任务运行器 AWS Lambda 函数获取上一步中的任务详细信息，并在 AWS Fargate 集群中运行 Amazon ECS 工作线程任务。Amazon ECS API 使用该 RunTask 操作来运行工作线程任务。启动这些工作线程任务，然后等待领导者任务的启动消息以开始测试。每个定义的 RunTask 操作限制为 10 个任务。如果您的任务计数大于 10，则任务定义将多次运行，直到所有工作人员任务都已启动。该函数还会生成一个前缀来区分结果解析 AWS Lambda 函数中的当前测试。
7. task-status-checkerAWS Lambda 函数检查所有的 Amazon ECS 工作线程任务是否都使用相同的测试 ID 运行。如果任务仍在置备，则它会等待一分钟，然后再次检查。所有 Amazon ECS 任务运行后，它会返回测试 ID、任务计数、测试类型、所有任务 IDs 和前缀，并将其传递给任务运行器函数。

8. 任务运行器 AWS Lambda 函数再次运行，这次是启动单个 Amazon ECS 任务来充当领导节点。此 ECS 任务向每个工作任务发送启动测试消息，以便同时启动测试。
9. task-status-checkerAWS Lambda 函数再次检查 Amazon ECS 任务是否使用相同的测试 ID 运行。如果任务仍在运行，它会等待一分钟，然后再次检查。一旦没有正在运行的 Amazon ECS 任务，它就会返回测试 ID、任务计数、测试类型和前缀。
10. 当任务运行器 AWS Lambda 函数在 AWS Fargate 集群中运行 Amazon ECS 任务时，每个任务都会从 Amazon S3 下载测试配置并开始测试。
11. 测试运行后，每个任务的平均响应时间、并发用户数、成功请求数和失败请求数将记录在 Amazon 中， CloudWatch 并可在 CloudWatch 控制面板中查看。
12. 如果您在测试中包含实时数据，则解决方案会 CloudWatch 使用订阅筛选器筛选实时测试结果。然后，该解决方案将数据传递给 Lambda 函数。
13. 然后，Lambda 函数会对收到的数据进行结构化处理，并将其发布到 AWS IoT Core 主题中。
14. Web 控制台订阅测试的 AWS IoT Core 主题，并接收发布到该主题的数据，以便在测试运行时绘制实时数据。
15. 测试完成后，容器镜像会将一份详细报告作为 XML 文件导出到 Amazon S3。每个文件都有一个 UUID 作为文件名。例如，s3://dlte-bucket/test-scenarios/ <$TEST_ID> /results/ <$UUID > .json。
16. 当 XML 文件上传到 Amazon S3 时，结果解析器 AWS Lambda 函数读取以前缀开头的 XML 文件中的结果，然后解析所有结果并将其汇总为一个汇总结果。
17. 结果解析器 AWS Lambda 函数将汇总结果写入亚马逊 DynamoDB 表。

MCP 服务器工作流程（可选）

如果您部署了可选的 MCP 服务器集成，AI 代理可以通过以下工作流程访问和分析您的负载测试数据：

MCP 服务器架构



1. 客户互动-客户通过 AWS Gateway 托管的 MCP 终端节点与 DLT 的 MCP 进行交互。 AgentCore AI 代理连接到此端点以请求访问负载测试数据。
2. 授权- AgentCore Gateway 处理针对解决方案 Cognito 用户池应用程序客户端的授权。网关会验证用户的 Cognito 令牌，以确保他们有权访问 DLT MCP 服务器。授权用户将获得访问权限，代理工具访问权限仅限于只读操作。
3. 工具规格 —— AgentCore 网关连接到 DLT MCP 服务器 Lambda 函数。工具规范定义了 AI 代理可用于与您的负载测试数据进行交互的可用工具。
4. 只读 API 访问权限-Lambda 函数的范围仅限于通过现有 DLT API Gateway 终端节点进行只读 API 访问。该函数提供四个主要操作：

- 列出方案-从 DynamoDB 场景表中检索测试场景列表
- 获取场景测试结果-访问来自 DynamoDB 和 S3 的特定场景的详细测试结果
- 获取 Fargate 负载测试运行器-查询有关在 ECS 集群中运行 Fargate 任务的信息
- 获取可用的区域堆栈-从中检索有关已部署的区域基础设施的信息 CloudFormation

MCP 服务器集成利用现有的 DLT 基础架构（API Gateway、Cognito、DynamoDB、S3），为基于人工智能的分析和见解提供对测试数据的安全、只读访问权限。

设计注意事项

本节介绍了 AWS 上的分布式负载测试解决方案的重要设计决策和配置选项，包括支持的应用程序、测试类型、计划选项和部署注意事项。

受支持的应用程序

只要您的 AWS 账户与应用程序之间存在网络连接，该解决方案就支持测试基于云的应用程序和本地应用程序。该解决方案支持 APIs 使用 HTTP 或 HTTPS 协议。

测试类型

AWS 上的分布式负载测试支持多种测试类型：简单的 HTTP 终端节点测试 JMeter、K6 和 Locust。

简单的 HTTP 端点测试

Web 控制台提供了 HTTP 端点配置接口，允许您测试任何 HTTP 或 HTTPS 端点，而无需编写自定义脚本。您可以定义端点 URL，从下拉菜单中选择 HTTP 方法（GET、POST、PUT、DELETE 等），还可以选择添加自定义请求标头和正文有效负载。此配置使您能够 APIs 使用自定义授权令牌、内容类型或应用程序所需的任何其他 HTTP 标头和请求正文进行测试。

JMeter 测试

使用 Web 控制台创建测试场景时，可以上传 JMeter 测试脚本。该解决方案将脚本上传到场景 S3 存储桶。当 Amazon ECS 任务运行时，他们会从 S3 下载 JMeter 脚本并执行测试。

> **Important**
>
> 尽管您的 JMeter 脚本可以定义并发性（虚拟用户）、交易速率 (TPS)、加速时间和其他加载参数，但该解决方案将使用您在创建测试期间在 Traffic Shape 屏幕中指定的值来覆盖这些配

置。Traffic Shape 配置控制测试执行的任务计数、并发度（每个任务的虚拟用户数）、加速持续时间和保持持续时间。

如果您有 JMeter 输入文件，则可以将输入文件与 JMeter 脚本一起压缩。您可以在创建测试场景时选择 zip 文件。

如果要包含插件，则捆绑的 zip 文件的 /plugins 子目录中包含的任何.jar 文件都将被复制到 JMeter 扩展目录中，可供负载测试。

> **Note**
>
> 如果在 JMeter 脚本文件中包含 JMeter 输入文件，则必须在 JMeter 脚本文件中包含输入文件的相对路径。此外，输入文件必须位于相对路径上。例如，当您的 JMeter 输入文件和脚本文件位于/ home/user directory and you refer to the input files in the JMeter script file, the path of input files must be ./INPUT_FILES. If you use /home/user/INPUT _FILES中时，测试将失败，因为它无法找到输入文件。

如果包含 JMeter 插件，则.jar 文件必须捆绑在 zip 文件根目录下名为 /plugins 的子目录中。相对于压缩文件的根目录，jar 文件的路径必须是。 /plugins/bundled_plugin.jar。

有关如何使用 JMeter 脚本的更多信息，请参阅JMeter 用户手册。

K6 测试

该解决方案支持基于 K6 框架的测试。K6 是在 AGPL -3.0 许可下发布的。创建新的 K6 测试时，解决方案会显示许可证确认消息。您可以将 K6 测试文件以及任何必要的输入文件上传到存档文件中。

> **Important**
>
> 尽管您的 K6 脚本可以定义并发性（虚拟用户）、阶段、阈值和其他负载参数，但该解决方案将使用您在创建测试期间在 “流量形状” 屏幕中指定的值来覆盖这些配置。Traffic Shape 配置控制测试执行的任务计数、并发度（每个任务的虚拟用户数）、加速持续时间和保持持续时间。

蝗虫试验

该解决方案支持基于 Locust 框架的测试。您可以将 Locust 测试文件以及任何必要的输入文件上传到存档文件中。

> **Important**
>
> 尽管您的 Locust 脚本可以定义并发性（用户数）、生成率和其他加载参数，但该解决方案将使用您在测试创建期间在 Traffic Shape 屏幕中指定的值来覆盖这些配置。Traffic Shape 配置控制测试执行的任务计数、并发度（每个任务的虚拟用户数）、加速持续时间和保持持续时间。

安排测试

该解决方案为运行负载测试提供了三个执行定时选项：

- 立即运行-创建后立即执行负载测试
- Run Onc e-在未来的特定日期和时间执行测试
- 按计划运行-使用 cron 表达式创建重复测试来定义计划

选择 “运行一次” 时，可以以 24 小时格式指定运行时间以及负载测试应开始运行的运行日期。

选择 “按计划运行” 时，可以手动输入 cron 表达式，也可以从常见的 cron 模式中进行选择（例如每小时、每天在特定时间、工作日或每月）。cron 表达式使用精细的计划格式，其中包含分钟、小时、月日、月、星期和年的字段。您还必须指定到期日期，该日期定义了计划测试何时停止运行。有关计划工作原理的更多信息，请参阅本指南的测试计划工作流程部分。

> **Note**
>
> - 测试时长：安排考试时请考虑测试的总持续时间。例如，启动时间为 10 分钟、保持时间为 40 分钟的测试大约需要 80 分钟才能完成。
> - 最小间隔：确保计划测试之间的间隔长于预计的测试持续时间。例如，如果测试需要大约 80 分钟，则将其安排为不超过每 3 小时运行一次。
> - 每小时限制：即使预计的考试时间少于一小时，系统也不允许安排只有一小时差异的测试。

并行测试

此解决方案为每项测试创建一个 Amazon CloudWatch 控制面板，实时显示 Amazon ECS 集群中运行的所有任务的组合输出。 CloudWatch 控制面板显示平均响应时间、并发用户数、成功请求数和失败请求数。该解决方案按秒汇总每个指标，并每分钟更新一次仪表板。

User management

在初始配置期间，您需要提供一个用户名和电子邮件地址，Amazon Cognito 使用该用户名和电子邮件地址来授予您访问解决方案网络控制台的权限。控制台不提供用户管理。要添加其他用户，您必须使用 Amazon Cognito 控制台。有关更多信息，请参阅 Amazon Cognito 开发者指南中的管理用户池中的用户。

要将现有用户迁移到 Amazon Cognito 用户池，请参阅 AWS 博客《将用户迁移到 Amazon Cognito 用户池的方法》。

区域部署

此解决方案使用 Amazon Cognito，仅在特定的 AWS 区域可用。因此，您必须在可用 Amazon Cognito 的区域部署此解决方案。有关按地区划分的最新可用服务，请参阅 AWS 区域服务列表。

规划您的部署

本节介绍成本、安全性、支持的区域、配额以及部署解决方案之前应查看的其他注意事项。

费用

运行此解决方案时使用的 AWS 服务的费用由您承担。总成本取决于运行的负载测试次数、这些测试的持续时间以及生成的数据量。从本次修订开始，在美国东部（弗吉尼亚北部）地区使用默认设置运行此解决方案的估计费用约为每月 30.90 美元。

下表提供了在美国东部（弗吉尼亚北部）地区使用默认参数部署此解决方案一个月的成本明细示例。

AWS 服务	Dimensions	成本 [美元]
AWS Fargate	10 个按需任务（使用两个 v CPUs 和 4 GB 的内存）运行 30 小时	29.62 美元
Amazon DynamoDB	1,000 个按需写入容量单位 1,000 个按需读取容量单位	0.0015
AWS Lambda	1,000 个请求 总时长 10 分钟	1.25 美元
AWS Step Functions	1,000 个状态转换	0.025 美元
总计：		每月 30.90 美元

解决方案资源标有 Key= SolutionId 和 value=so0062。您可以按照文档中的激活标签 SolutionId 来激活标签密钥。激活标签后，您可以按照文档创建成本类别来创建成本类别规则。您可以通过监控成本类别控制台并选择成本类别名称来查看解决方案产生的成本。

我们建议通过 AWS Cost Explorer 创建预算，以帮助管理成本。价格可能会发生变化。有关完整详情，请参阅本解决方案中使用的每项 AWS 服务的定价网页。

> **Note**
>
> 默认任务配置为每个任务使用 2 v CPUs 和 4 GB 的内存。如果您的负载测试不需要这些资源，则可以减少这些资源以降低成本。相反，您可以增加资源以支持每项任务的更高并发度。有关更多信息，请参阅本指南中的增加容器资源部分。

> **Note**
>
> 此解决方案提供了在运行测试时包含实时数据的选项。此功能需要额外的 AWS Lambda 函数和 AWS IoT Core 主题，这会产生额外费用。

MCP 服务器额外费用（可选）

下表提供了 MCP 服务器集成的成本明细以及美国东部（弗吉尼亚北部）地区一个月的定价。

服务组件	Dimensions	成本 [美元]
AgentCore 网关-工具索引	10 把工具 × 每 100 把刀具 0.02 美元	0.002
AgentCore 网关-搜索 API	10,000 次互动 × 每 1,000 次 0.025 美元	0.25 美元
AgentCore 网关-API 调用	50,000 次调用 × 每 1,000 次调用 0.005 美元	0.25 美元
AWS Lambda 函数	因使用情况而异（典型工作负载）	5.00 美元至 20.00 美元
估计的额外费用总额：		每月 5.50 美元至 20.50 美元

价格可能会发生变化。有关 AgentCore 网关定价的完整详情，请参阅 Amazon Bedrock 定价（AgentCore 网关部分）。有关 Lambda 定价的信息，请参阅 AWS Lambda 定价。

安全性

当您在 AWS 基础设施上构建系统时，安全责任由您和 AWS 共同承担。这种分担责任模式减轻了您的运营负担，因为 AWS 运营、管理和控制包括主机操作系统、虚拟化层和服务运行设施的物理安全在内的组件。有关 AWS 安全的更多信息，请访问 AWS 云安全。

IAM 角色

AWS Identity and Access Management (IAM) 角色允许客户向 AWS 云上的服务和用户分配精细的访问策略和权限。此解决方案创建 IAM 角色，这些角色向解决方案的 AWS Lambda 函数授予创建区域资源的访问权限。

Amazon CloudFront

此解决方案部署了托管在 Amazon S3 存储桶中的网页用户界面，该存储桶由亚马逊 CloudFront分发。为了帮助减少延迟和提高安全性，该解决方案包括一个具有原始访问身份的 CloudFront 分发，即提供对解决方案网站存储桶内容的公开访问权限的 CloudFront 用户。默认情况下，该 CloudFront 发行版使用 TLS 1.2 来强制执行最高级别的安全协议。有关更多信息，请参阅《亚马逊 CloudFront 开发者指南》中的限制对 Amazon S3 来源的访问。

CloudFront 激活其他安全缓解措施，将 HTTP 安全标头附加到每个查看者响应中。有关更多信息，请参阅在 CloudFront 响应中添加或删除 HTTP 标头。

此解决方案使用默认 CloudFront 证书，其支持的最低安全协议为 TLS v1.0。要强制使用 TLS v1.2 或 TLS v1.3，必须使用自定义 SSL 证书而不是默认 CloudFront 证书。有关更多信息，请参阅如何将我的 CloudFront 发行版配置为使用 SSL/TLS 证书。

Amazon API Gateway

该解决方案部署了边缘优化的 Amazon API Gateway 终端节点，以使用默认 API Gateway 终端节点而不是自定义域来提供 RESTful APIs 负载测试功能。对于 APIs 使用默认端点进行边缘优化，API Gateway 使用 TLS-1-0 安全策略。有关更多信息，请参阅 Amazon API Gateway 开发者指南 APIs中的使用 REST。

此解决方案使用默认 API Gateway 证书，该证书支持的最低安全协议为 TLS v1.0。要强制使用 TLS v1.2 或 TLS v1.3，您必须使用带有自定义 SSL 证书的自定义域名，而不是默认的 API Gateway 证书。有关更多信息，请参阅为 REST 设置自定义域名 APIs。

AWS Fargate 安全组

默认情况下，此解决方案向公众开放 AWS Fargate 安全组的出站规则。如果您想阻止 AWS Fargate 向任何地方发送流量，请将出站规则更改为特定的无类域间路由 (CIDR)。

该安全组还包括一条入站规则，允许端口 50,000 上的本地流量流向属于同一安全组的任何来源。这用于允许容器相互通信。

网络 stress test

根据网络压力测试政策，您有责任使用此解决方案。本政策涵盖了诸如您计划将大量网络测试直接从您的 Amazon EC2 实例运行到其他位置（例如其他亚马逊实例、AWS 属性/服务或外部终端节点） EC2 的情况。这些测试有时被称为压力测试、负载测试或比赛日测试。大多数客户测试不属于本政策的范围；但是，如果您认为自己产生的流量总共将持续超过 1 分钟、超过 1 Gbps（每秒 10 亿位）或超过 1 Gpps（每秒 10 亿个数据包），则请参阅本政策。

限制对公共用户界面的访问

要在 IAM 和 Amazon Cognito 提供的身份验证和授权机制之外限制对面向公众的用户界面的访问，请使用 AWS WAF（网络应用程序防火墙）安全自动化解决方案。

此解决方案会自动部署一组 AWS WAF 规则，用于过滤常见的基于 Web 的攻击。用户可以从预配置的保护功能中进行选择，这些功能定义了 AWS WAF Web 访问控制列表 (Web ACL) 中包含的规则。

MCP 服务器安全（可选）

如果您部署了可选的 MCP 服务器集成，则该解决方案将使用 AWS AgentCore Gateway 为 AI 代理提供对负载测试数据的安全访问。 AgentCore Gateway 会验证每个请求的 Amazon Cognito 身份验证令牌，确保只有经过授权的用户才能访问 MCP 服务器。MCP 服务器 Lambda 函数实现只读访问模式，从而防止 AI 代理修改测试配置或结果。所有 MCP 服务器交互都使用与 Web 控制台相同的权限边界和访问控制。

支持的 AWS 区域

该解决方案使用 Amazon Cognito 服务，但该服务目前并非在所有 AWS 区域都可用。要了解按地区划分的 AWS 服务的最新可用性，请参阅 AWS 区域服务列表。

AWS 上的分布式负载测试可在以下 AWS 区域中使用：

区域名称	
美国东部（俄亥俄州）	亚太地区（东京）
美国东部（弗吉尼亚州北部）	加拿大（中部）
美国西部（加利福尼亚北部）	欧洲地区（法兰克福）
美国西部（俄勒冈州）	欧洲地区（爱尔兰）
亚太地区（孟买）	欧洲地区（伦敦）
亚太地区（首尔）	欧洲地区（巴黎）
亚太地区（新加坡）	欧洲地区（斯德哥尔摩）
亚太地区（悉尼）	南美洲（圣保罗）

支持 MCP 服务器的 AWS 区域（可选）

如果您计划部署可选的 MCP 服务器集成，则必须在提供 AWS Gate AgentCore way 的 AWS 区域部署该解决方案。MCP 服务器功能仅在以下 AWS 区域可用：

区域名称	区域代码
美国东部（弗吉尼亚州北部）	us-east-1
美国西部（俄勒冈州）	us-west-2
亚太地区（新加坡）	ap-southeast-1
亚太地区（悉尼）	ap-southeast-2
Asia Pacific (Tokyo)	ap-northeast-1
欧洲地区（法兰克福）	eu-central-1
欧洲地区（爱尔兰）	eu-west-1
欧洲地区（伦敦）	eu-west-2

区域名称	区域代码
欧洲地区（巴黎）	eu-west-3

要了解按地区划分的 AWS AgentCore Gateway 的最新可用性，请参阅 AWS AgentCore 网关开发人员指南中的 AWS AgentCore 网关终端节点和配额。

限额

服务限额（也称为限制）是您的 AWS 账户使用的服务资源或操作的最大数量。

此解决方案中 AWS 服务的配额

请确保此解决方案中实施的每项服务都有足够的限额。有关更多信息，请参阅 AWS 服务限额。

使用以下链接转到该服务的页面。要在不切换页面的情况下查看文档中所有 AWS 服务的服务配额，请改为查看 PDF 中服务终端节点和配额页面中的信息。

AWS CloudFormation 配额

您的 AWS 账户有 AWS CloudFormation 配额，在此解决方案中启动堆栈时应注意这些配额。通过了解这些限额，可以避免阻碍成功部署此解决方案的限制错误。有关更多信息，请参阅 AWS CloudFormation 用户指南中的 AWS CloudFormation 配额。

负载测试配额

使用 AWS Fargate 启动类型在 Amazon ECS 中可以运行的最大任务数取决于任务的 vCPU 大小。AWS 上的分布式负载测试中的默认任务大小为 2 个 vCPU。要查看当前的默认配额，请参阅 Amazon ECS 服务配额。经常账户配额可能与列出的配额不同。要查看特定于账户的配额，请在 AWS 管理控制台中查看 Fargate 按需 vCPU 资源数量的服务配额。有关如何申请增加配额的说明，请参阅 AWS 一般参考指南中的 AWS 服务配额。

Amazon Linux 2023 容器镜像（安装了 Taurus）不限制每个任务的并发连接，但这并不意味着它可以支持无限数量的用户。要确定容器可以为测试生成的并发用户数，请参阅本指南的确定用户数量部分。

> **Note**
>
> 根据默认设置，建议的并发用户限制为 200 个用户。

并行测试

此解决方案为每项测试创建一个 Amazon CloudWatch 控制面板，实时显示 Amazon ECS 集群中运行的所有任务的组合输出。 CloudWatch 控制面板显示平均响应时间、并发用户数、成功请求数和失败请求数。该解决方案按秒汇总每个指标，并每分钟更新一次仪表板。

亚马逊 EC2 测试政策

只要您的网络流量保持在 1 Gbps 以下，您就无需获得 AWS 的批准即可使用此解决方案运行负载测试。如果您的测试产生的速度超过 1 Gbps，请联系 AWS。有关更多信息，请参阅 A mazon EC2 测试政策。

亚马逊 CloudFront 负载测试政策

如果您计划对 CloudFront 终端节点进行负载测试，请参阅《Amazon CloudFront 开发者指南》中的负载测试指南。我们还建议将流量分散到多个任务和区域。为负载测试提供至少 30 分钟的启动时间。对于每秒发送超过 500,000 个请求或要求超过 300 Gbps 数据的负载测试，我们建议先获得发送流量的预先批准。 CloudFront 可能会限制影响 CloudFront 服务可用性的未经批准的负载测试流量。

在部署后监控解决方案

部署解决方案后，我们建议使用 Amazon CloudWatch 警报和指标持续监控解决方案的资源。

设置 CloudWatch 警报

您可以设置CloudWatch 警报以监控关键指标，并在超过阈值时接收通知。考虑为以下资源设置警报：

亚马逊 CloudFront 配送指标

监控 CloudFront 配送性能和错误。有关更多信息，请参阅《Amazon CloudFront 开发者指南》中的CloudFront 配送指标。

Amazon API Gateway 指标

监控 API 请求速率、延迟和错误。有关更多信息，请参阅亚马逊 API Gateway 开发者指南中的亚马逊 API Gateway 维度和指标。

AWS Lambda 函数指标

监控解决方案微服务的 Lambda 函数调用、持续时间、错误和限制。

亚马逊 ECS 和 AWS Fargate 指标

在负载测试期间监控任务 CPU 和内存利用率，以确保有足够的资源。

亚马逊 DynamoDB 指标

监控读取和写入容量消耗、受限请求和延迟。

部署解决方案

该解决方案使用 AWS CloudFormation 模板和堆栈来自动部署。这些 CloudFormation 模板指定了此解决方案中包含的 AWS 资源及其属性。 CloudFormation 堆栈提供模板中描述的资源。

部署流程概述

在部署解决方案之前，请查看本指南前面讨论的成本、架构、安全性和其他注意事项。

部署时间：主堆栈大约 15 分钟，再加上每增加一个区域 5 分钟

AWS CloudFormation 模板

您可以先下载此解决方案的 CloudFormation 模板，然后再进行部署。此解决方案使用 AWS 在 AWS CloudFormation 上自动部署分布式负载测试。它包括以下 AWS CloudFormation 模板，您可以在部署前下载该模板：

View template

distribu load-testing-on-aws.templat e-使用此模板启动解决方案和所有关联组件。默认配置部署了本解决方案部分的 AWS 服务中的核心和支持服务，但您可以自定义模板以满足您的特定需求。

> **Note**
>
> AWS CloudFormation 资源是基于 AWS Cloud Development Kit (AWS CDK) 结构创建的。如果您之前部署过此解决方案，请参阅更新解决方案以获取更新说明。

启动 堆栈

按照以下步骤将 AWS 上的分布式负载测试解决方案部署到您的账户。

> **Note**
>
> 该解决方案包括 AWS 的数据收集指标。我们使用这些数据来更好地了解客户如何使用此解决方案以及相关服务和产品。通过本次调查收集的数据归AWS所有。数据收集受 AWS 隐私声明的约束。

此自动化 AWS CloudFormation 模板在 AWS 上部署分布式负载测试。

> **Note**
>
> 运行此解决方案时使用的 AWS 服务的费用由您承担。有关更多详情，请访问本指南中的成本部分，并参阅本解决方案中使用的每项 AWS 服务的定价网页。

1. 登录 AWS 管理控制台并选择启动 CloudFormation 模板的按钮。

   Launch solution

   或者，您可以下载模板作为自己实现的起点。
2. 默认情况下，此模板在美国东部（弗吉尼亚州北部）区域启动。要在不同的 AWS 区域启动此解决方案，请使用控制台导航栏中的区域选择器。

   > **Note**
   >
   > 该解决方案使用 Amazon Cognito，目前仅在特定的 AWS 区域可用。因此，您必须在提供 Amazon Cognito 的 AWS 地区启动此解决方案。有关按地区划分的最新可用服务，请参阅 AWS 区域服务列表。
3. 在创建堆栈页面上，验证 Amazon S3 URL 文本框中是否显示了正确的模板 URL，然后选择下一步。
4. 在指定堆栈详细信息页面上，为您的解决方案堆栈分配一个名称。
5. 在参数下，检查模板的参数，并根据需要进行修改。该解决方案使用以下默认值。

参数	默认值	描述
管理员姓名	<需要输入>	初始解决方案管理员的用户名。
管理员邮箱	*<Requires input>*	管理员用户的电子邮件地址。启动后，将向该地址发送一封包含控制台登录说明的电子邮件。

参数	默认值	描述
现有 VPC ID	<Optional input>	如果您有要使用的 VPC 并且已经创建，请输入部署堆栈的同一区域中的现有 VPC 的 ID。例如，vpc-1a2b3c4d5e6f。
第一个现有子网	<Optional input>	现有 VPC 中第一个子网的 ID。此子网需要一条通往 Internet 的路由，才能提取容器镜像以进行运行测试。例如，subnet-7h8i9j0k。
第二个现有子网	<Optional input>	现有 VPC 中第二个子网的 ID。此子网需要一条通往 Internet 的路由，才能提取容器镜像以进行运行测试。例如，subnet-1x2y3z。
为创建 VPC 的解决方案提供有效的 CIDR 块	192.168.0.0/16	如果您使用的是现有 VPC，则可以将此参数留空
为子网 A 提供有效的 CIDR 块，以便解决方案创建 VPC	192.168.0.0/20	AWS Fargate VPC 子网 A 的 CIDR 块
为子网 B 提供有效的 CIDR 块，以便解决方案创建 VPC	192.168.16.0/20	AWS Fargate VPC 子网 B 的 CIDR 块
提供 CIDR 区块以允许 Fargate 任务的出站流量	0.0.0.0/0	限制 Amazon ECS 容器出站访问的 CIDR 块。
自动更新容器镜像	No	在下一个次要版本发布之前，自动使用最新、最安全的映像。选择No将拉取最初发布的映像，不进行任何安全更新。

参数	默认值	描述
部署可选 MCP 服务器	No	部署可选的远程 MCP 服务器，使用 AgentCore Gateway 将 AI 应用程序连接到 AWS 上的分布式负载测试。

6. 选择 Next(下一步)。
7. 在 配置堆栈选项 页面上，请选择 下一步。
8. 在 Review 页面上，审核并确认设置。选中确认模板将创建 AWS Identity and Access Management (IAM) 资源的复选框。
9. 选择 Create stack（创建堆栈）以部署堆栈。

您可以在 AWS CloudFormation 控制台的 “状态” 列中查看堆栈的状态。大约 15 分钟后，您应该会收到 “创建完成” 状态。

> **Note**
>
> 除了主要 AWS Lambda 函数外，该解决方案还包括自定义资源 Lambda 函数，该函数仅在初始配置期间或资源更新或删除时运行。

运行此解决方案时，自定义资源 Lambda 函数处于非活动状态。但是，请勿删除此功能，因为这是管理关联资源所必需的。

多区域部署

部署时间：每个区域大约 5 分钟

您可以跨多个区域运行测试。部署分布式负载测试解决方案时，它会创建一个区域 CloudFormation 模板并将其存储在方案 S3 存储桶中。

> **Note**
>
> 场景存储桶名称包括您的堆栈名称和关键字 “场景”。您可以通过导航到 S3 控制台并搜索名称中包含 “场景” 的存储桶来找到它。

要运行多区域部署，您必须在要运行测试的区域中部署存储在 Amazon S3 场景存储桶中的区域 CloudFormation 模板。您可以通过执行以下操作来安装区域模板：

1. 在解决方案的 Web 控制台中，导航到左侧菜单中的控制面板。
2. 使用剪贴板图标在 Amazon S3 中复制 CloudFormation 模板链接。
3. 登录 A WS CloudFormation 控制台并选择正确的区域。
4. 在创建堆栈页面上，验证 Amazon S3 URL 文本框中是否显示了正确的模板 URL，然后选择下一步。
5. 在指定堆栈详细信息页面上，为您的解决方案堆栈分配一个名称。
6. 在参数下，检查模板的参数，并根据需要进行修改。该解决方案使用以下默认值。

参数	默认值	描述
现有 VPC ID	<Optional input>	如果您有要使用的 VPC 并且已经创建，请输入部署堆栈的同一区域中的现有 VPC 的 ID。例如，vpc-1a2b3c4d5e6f。
第一个现有子网	<Optional input>	现有 VPC 中第一个子网的 ID。此子网需要一条通往 Internet 的路由，才能提取容器镜像以进行运行测试。例如，subnet-7h8i9j0k。
第二个现有子网	<Optional input>	现有 VPC 中第二个子网的 ID。此子网需要一条通往 Internet 的路由，才能提取容器镜像以进行运行测试。例如，subnet-1x2y3z。
为创建 VPC 的解决方案提供有效的 CIDR 块	192.168.0.0/16	如果您没有为现有 VPC 提供值，则解决方案创建的 Amazon VPC 的 CIDR 块将包含 AWS Fargate 的 IP 地址。

参数	默认值	描述
提供 CIDR 区块以允许 Fargate 任务的出站流量	0.0.0.0/0	限制 Amazon ECS 容器出站访问的 CIDR 块。

7. 选择 Next(下一步)。
8. 在 配置堆栈选项 页面上，请选择 下一步。
9. 在 Review 页面上，审核并确认设置。请务必勾选复选框，确认模板将创建 AWS Identity and Access Management (IAM) 资源。
10. 选择 Create stack（创建堆栈）以部署堆栈。

    您可以在 AWS CloudFormation 控制台的 “状态” 列中查看堆栈的状态。大约五分钟后，您应该会收到 CREATE_ COMPLETE 状态。

成功部署区域后，它们将显示在 Web 控制台中。创建测试时，控制面板和场景创建中会列出所有可用区域。您可以在场景创建的 “流量形状” 步骤中向测试中添加区域。

该解决方案在场景表中为每个部署的区域创建一个 DynamoDB 项目，其中包含有关该区域中测试资源的必要信息。您可以在 Web 控制台中按区域对测试结果进行排序。要在多区域测试中查看所有区域的汇总结果，请使用 Amazon CloudWatch 指标。测试完成后，您可以在测试结果中找到图形的源代码。

> **Note**
>
> 您可以在没有 Web 控制台的情况下启动区域堆栈。在 Amazon S3 场景存储桶中获取区域模板的链接，并在所需区域启动区域堆栈时将其作为来源提供。或者，您可以下载模板并将其作为所需区域的来源上传。

更新此解决方案

更新解决方案会将最新的功能、安全补丁和错误修复应用于您的部署。如果您之前部署过该解决方案，请按照以下步骤将 CloudFormation 堆栈更新到最新版本。

> **Important**
>
> 更新之前，请确保当前未运行任何负载测试。更新过程可能会暂时中断解决方案的可用性。

1. 登录CloudFormation 控制台，选择现有 CloudFormation 堆栈，然后选择更新堆栈。
2. 选择 “直接更新”。
3. 选择 “替换现有模板”。
4. 在指定模板下：
   a. 选择 Amazon S3 URL。
   b. 复制最新模板的链接。
   c. 将链接粘贴到 Amazon S3 URL 框中。
   d. 确认 Amazon S3 网址文本框中显示的模板网址是否正确。
   e. 选择下一步。
   f. 再次选择下一步。
5. 在参数下，检查模板的参数，并根据需要进行修改。有关参数的详细信息，请参阅启动堆栈。
6. 选择 Next(下一步)。
7. 在 配置堆栈选项 页面上，请选择 下一步。
8. 在 Review 页面上，审核并确认设置。
9. 选中确认模板可能创建 IAM 资源的复选框。
10. 选择查看更改集并验证更改。
11. 选择更新堆栈以部署堆栈。

您可以在 AWS CloudFormation 控制台的 “状态” 列中查看堆栈的状态。您将在大约 15 分钟后收到`UPDATE_COMPLETE`状态。

> **Note**
>
> 如果您在堆栈升级后从浏览器登录时遇到 Amazon Cognito 身份验证问题，请刷新浏览器（打开 Ctrl+Shift+R Windows/Linux 或 Mac 上的 Cmd+Shift+R）以清除缓存的数据并重试。

对 v3.3.0 之前版本的更新进行故障排除

> **Note**
>
> 本节仅适用于 v3.3.0 之前版本的更新。如果您要从 v3.3.0 或更高版本进行更新，请按照上述标准更新程序进行操作。

1. 下载 distributed-load-testing-on-aws.templat e。
2. 打开模板并导航至 “条件:”，然后查找 DLTCommon ResourcesAppRegistryCondition
3. 您应看到类似如下所示的内容：

```
Conditions:
DLTCommonResourcesAppRegistryConditionCCEF54F8:
Fn::Equals:
- "true"
- "true"
```

4. 将第二个真值更改为 false：

```
Conditions:
DLTCommonResourcesAppRegistryConditionCCEF54F8:
Fn::Equals:
- "true"
- "false"
```

5. 使用自定义模板更新您的堆栈。
6. 此更新将从堆栈中移除与应用程序注册表相关的资源，从而使更新成功完成。
7. 使用最新的模板 URL 执行另一次堆栈更新，将应用程序注册表应用程序资源重新添加到堆栈中。

更新区域堆栈

如果您已在多个区域部署解决方案，则必须分别更新每个区域堆栈。在已部署测试基础设施的区域中，按照每个区域 CloudFormation 堆栈的标准更新程序进行操作。

AWS Systems Manager 应用程序管理器

更新解决方案后，AWS Systems Manager 应用程序管理器提供解决方案及其资源的应用程序级视图。您可以使用应用程序管理器来：

- 从中央位置监控资源、跨堆栈和 AWS 账户部署的资源的成本以及日志。
- 在应用程序环境中查看解决方案资源的操作数据，例如部署状态、 CloudWatch 警报、资源配置和操作问题。

故障排除

已知问题解决方案提供了缓解已知错误的说明。如果这些说明无法解决您的问题，请联系 AWS Support 提供有关如何为该解决方案提出 AWS Support 案例的说明。

已知问题解决方案

问题：您使用的是现有 VPC，并且测试失败，状态为 “失败”，从而导致出现以下错误消息：

```
Test might have failed to run.
```

- 解决方案：

确保子网存在于指定的 VPC 中，并且这些子网具有通过互联网网关或 NAT 网关到互联网的路由。AWS Fargate 需要访问权限才能从公共存储库中提取容器映像才能成功运行测试。

问题：测试运行时间太长或无法无限期运行

- 解决方案：

取消测试并检查 AWS Fargate，确保所有任务都已停止。如果他们尚未停止，请手动停止所有 Fargate 任务。检查您账户的按需 Fargate 任务限制，确保您可以启动所需数量的任务。您还可以查看 Lambda 任务运行器函数的 CloudWatch 日志，以更深入地了解启动 Fargate 任务时出现的故障。查看 CloudWatch ECS 日志，详细了解正在运行的 Fargate 容器中发生的事情。

问题：测试已开始但未能完成，或者 ECS 任务的状态未知

- 解决方案：

如果您选择在部署解决方案的账户中提供现有 VPC 的选项，请确保 ECS 任务使用的 VPC 有足够的免费 IP 地址来启动测试输入中提供的任务数量。ECS 任务定义使用需要互联网网关或互联网路由的 ECR 映像，这样 ECS 服务就可以通过从 aw distributed-load-testing-on s-solutions/-下载解决方案 ECR 映像来配置任务。aws-load-tester如果由于 VPC 中的所有子网都是私有子网而无法提供互联网路由，则可以使用 ECR 拉取缓存将 ECR 镜像托管在您的账户中。使用新的 ECR 镜像 URI 更新任务定义并创建新的修订版。更新任务定义后，需要更新 DynamoDB 表中的解决方案配置以使用新的修订版。DynamoDB 表名可以在密钥下方的堆栈输出选项卡 CloudFormation 中找到。 ScenariosTable使用密钥 testID 和值 region-[SOLUTION-DEPLOYED-REGION] 更新该项目的属性 TaskDefinition。

问题：测试需要使用私有或无法通过 Internet 网关访问的端点

- 解决方案：

在测试无法通过互联网网关访问的私有 API 端点时，请考虑以下方法：

1. 网络配置：确保将 ECS 任务使用的子网路由表更新为指向正在测试的私有终端节点 IP 地址范围的路由。这允许测试流量到达您的 VPC 内的私有终端节点。
2. DNS 解析：对于自定义域，请在您的 VPC 中配置 DNS 设置以解析私有终端节点的域名。有关详细说明，请参阅 VPC DNS 文档。
3. VPC 终端节点：如果测试 AWS 服务，请考虑使用 VPC 终端节点 (AWS PrivateLink) 建立私有连接。例如，要测试私有 API 网关，您可以为 API Gateway 创建 VPC 终端节点。参见私有 API Gateway 文档。
4. VPC 对等互连：如果私有终端节点位于其他 VPC 中，请在部署解决方案的 VPC 和包含私有终端节点的 VPC 之间建立 VPC 对等关系。在两者中配置适当的路由表 VPCs。请参阅 VPC 对等互连文档。
5. T@@ ransit Gateway：对于涉及多个的更复杂的联网场景 VPCs，可以考虑使用 AWS Transit Gateway 在解决方案的 VPC 和包含私有终端节点的 VPC 之间路由流量。请参阅 Transit Gateway 文档
6. 安全组：确保与您的 ECS 任务关联的安全组允许私有终端节点的出站流量，并且私有终端节点的安全组允许来自 ECS 任务的入站流量。

要测试内部应用程序负载均衡器或 EC2 实例，请确保 VPC CIDR 范围不重叠，并在路由表中配置必要的路由。

问题：测试已完成，但用户界面上未显示结果

- 解决方案：

如果测试已完成，但用户界面中没有结果，则结果文件仍应位于运行测试的 ECS 任务的 S3 存储桶中。这是解决方案中的一个已知限制。在当前架构中，该解决方案使用结果解析 Lambda 函数来汇总多个 ECS 任务的结果，然后将这些结果作为一个项目存储在 DynamoDB 表中。DynamoDB 表的最大项目大小限制为 400 KB。是否达到此限制取决于测试脚本的复杂性、并发性和正在使用的任务数。该错误并不意味着测试失败；它表示汇总结果并将其存储在 DynamoDB 表中以进行 CRUD 操作的过程失败。测试场景的结果仍可在 S3 存储桶中找到。

联系 AWS Support

如果您有 AWS 开发者支持、AWS 商业支持或 AWS 企业支持，则可以使用支持中心获取有关此解决方案的专家帮助。以下部分提供了说明。

创建案例

1. 登录 Su pport Center。
2. 选择创建案例。

我们能帮上什么忙？

1. 选择 “技术”
2. 对于 “服务”，选择 “解决方案”。
3. 对于类别，选择在 AWS 上进行分布式负载测试。
4. 在 “严重性” 中，选择与您的用例最匹配的选项。
5. 当您输入 “服务”、“类别” 和 “严重性” 时，界面会填充常见疑难解答问题的链接。如果您无法通过这些链接解决问题，请选择下一步：其他信息。

其他信息

1. 在 “主题” 中，输入总结您的问题或问题的文本。
2. 在描述中，详细描述问题。
3. 选择 “附加文件”。
4. 附上 AWS Support 处理请求所需的信息。

帮助我们更快地解决您的问题

1. 输入所需的信息。
2. 选择下一步：立即解决或联系我们。

立即解决或联系我们

1. 查看 “立即解决” 解决方案。

2. 如果您无法使用这些解决方案解决问题，请选择 “联系我们”，输入所需信息，然后选择 “提交”。

卸载此解决方案

您可以从 AWS 管理控制台或使用 AWS 命令行界面卸载 AWS 上的分布式负载测试解决方案。您必须手动删除此解决方案创建的控制台、场景和日志亚马逊简单存储服务 (Amazon S3) 存储桶。如果您有数据要保留，AWS 解决方案实施不会自动将其删除。

> **Note**
>
> 如果您已部署区域堆栈，则必须先删除这些区域中的堆栈，然后再删除主堆栈。

使用 AWS 管理控制台

1. 登录 A WS CloudFormation 控制台。
2. 在堆栈页面上，选择此解决方案的安装堆栈。
3. 选择删除。

使用 AWS 命令行界面

确定 AWS 命令行界面 (AWS CLI) Line CLI 在您的环境中是否可用。有关安装说明，请参阅 AWS CLI 用户指南中的 AWS 命令行界面是什么。确认 AWS CLI 可用后，运行以下命令。

```
$ aws cloudformation delete-stack --stack-name <installation-stack-name>
```

删除 Amazon S3 存储桶

如果您决定删除 AWS CloudFormation 堆栈以防止数据意外丢失，则此解决方案配置为保留解决方案创建的 Amazon S3 存储桶（用于在可选区域进行部署）。卸载解决方案后，如果您不需要保留数据，则可以手动删除此 S3 存储桶。按照以下步骤删除 Amazon S3 存储桶。

1. 登录 Amazon S3 控制台。
2. 在左侧导航窗格中，选择桶。
3. 在 “按名称查找存储桶” 字段中，输入此解决方案堆栈的名称。
4. 选择解决方案的 S3 存储桶之一，然后选择 Empty 。

5. 在验证字段中输入 “永久删除”，然后选择 “空”。
6. 选择您刚刚清空的 S3 存储桶，然后选择删除。
7. 在验证字段中输入 S3 存储桶名称，然后选择删除存储桶。

重复步骤 4 到 7，直到删除所有 S3 存储桶。

要使用 AWS CLI 删除 S3 存储桶，请运行以下命令：

```
$ aws s3 rb s3://<bucket-name> --force
```

使用解决方案

本节提供了使用 AWS 分布式负载测试解决方案的全面指南，从创建第一个测试场景到分析详细结果。该工作流程包括创建测试场景、运行测试和浏览测试结果。

创建测试场景

创建测试场景包括四个主要步骤：配置常规设置、定义场景、调整流量模式以及查看配置。

步骤 1：常规设置

配置负载测试的基本参数，包括测试名称、描述和常规配置选项。

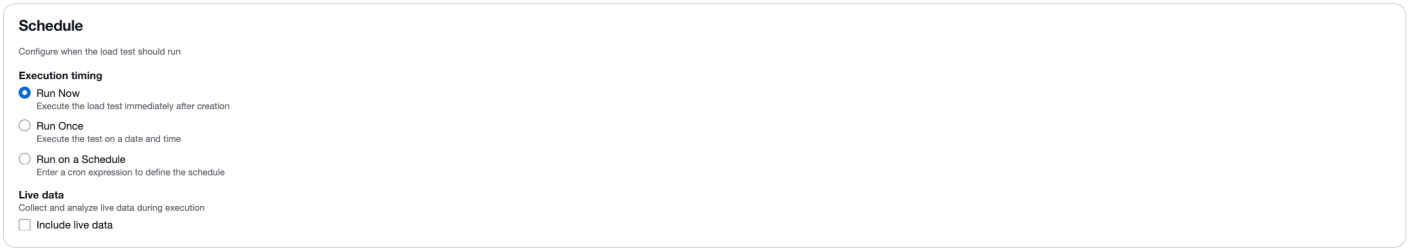
测试识别

- 测试名称（必填）-测试场景的描述性名称
- 测试描述（必填）-有关测试目的和配置的其他详细信息
- 标签（可选）-添加最多 5 个标签来对您的测试场景进行分类和整理

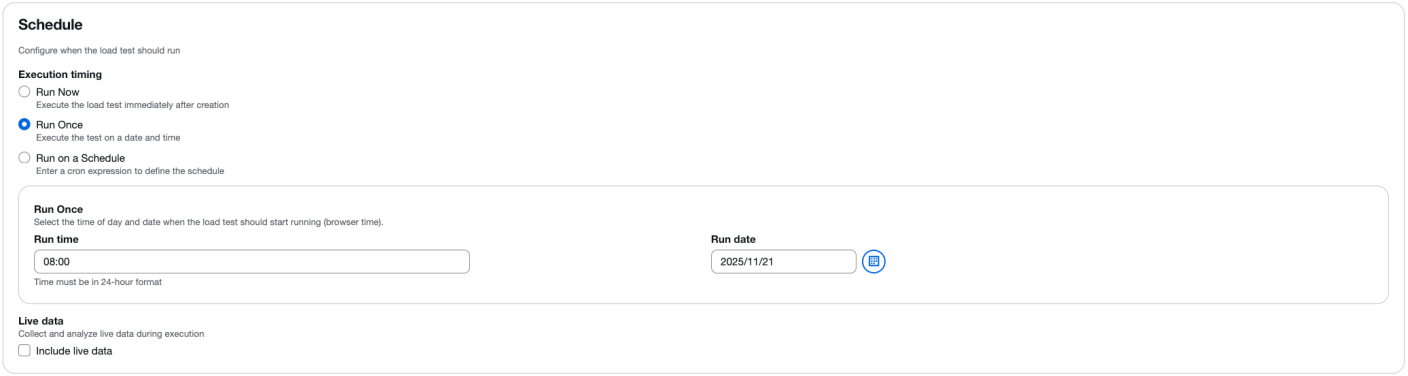
日程安排选项

配置测试的运行时间：

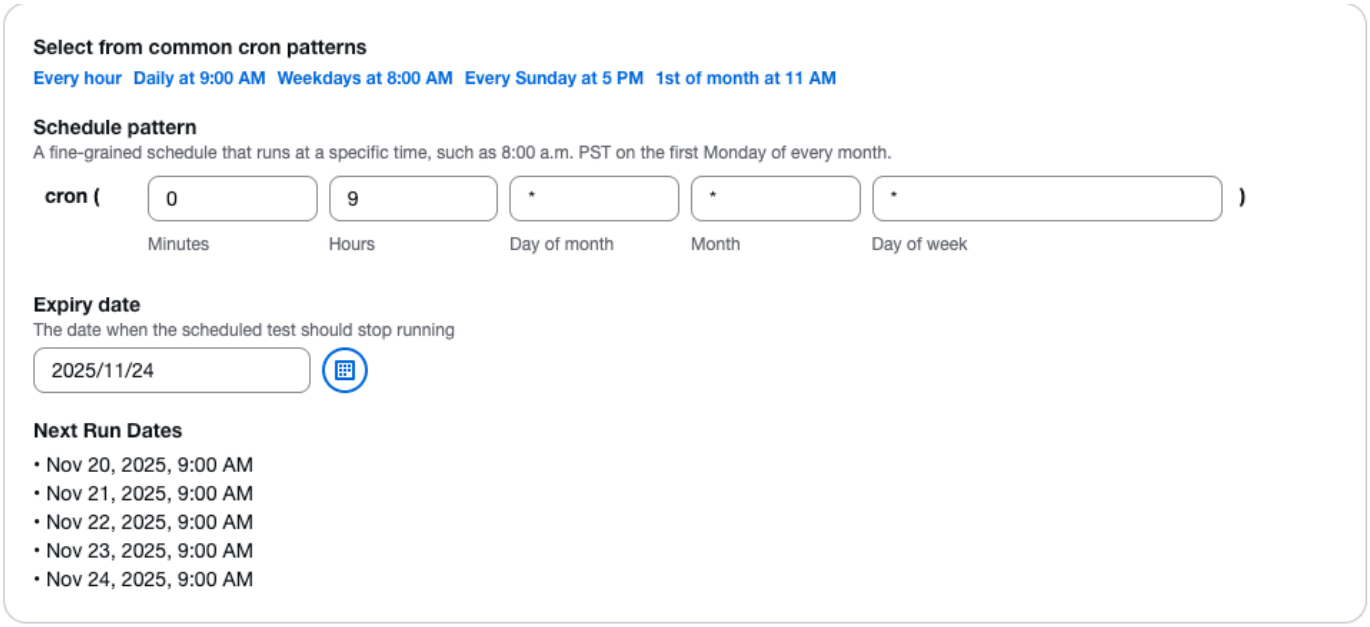
- 立即运行-创建后立即执行测试。



- 运行一次-将测试安排在特定的日期和时间运行。



- 按计划运行-使用基于 cron 的计划定期自动运行测试。您可以从常用模式（每小时、每天、每周）中进行选择，也可以定义自定义 cron 表达式。



调度工作流程

安排测试时，会出现以下工作流程：

- 计划参数通过 Amazon API Gateway 发送到解决方案的 API。
- API 将参数传递给 Lambda 函数，该函数创建计划在指定日期运行 CloudWatch 的事件规则。
- 对于一次性测试（运行一次），CloudWatch 事件规则在指定日期运行，`api-services`Lambda 函数执行测试。
- 对于定期测试（按计划运行），CloudWatch 事件规则在指定日期激活，而 Lamb `api-services` da 函数会创建一个根据指定频率立即重复运行的新规则。

实时数据

选中 “包括实时数据” 复选框可在测试运行时查看实时指标。启用后，您可以监控：

- 平均响应时间。
- 虚拟用户计数。
- 成功的请求计数。
- 失败的请求计数。

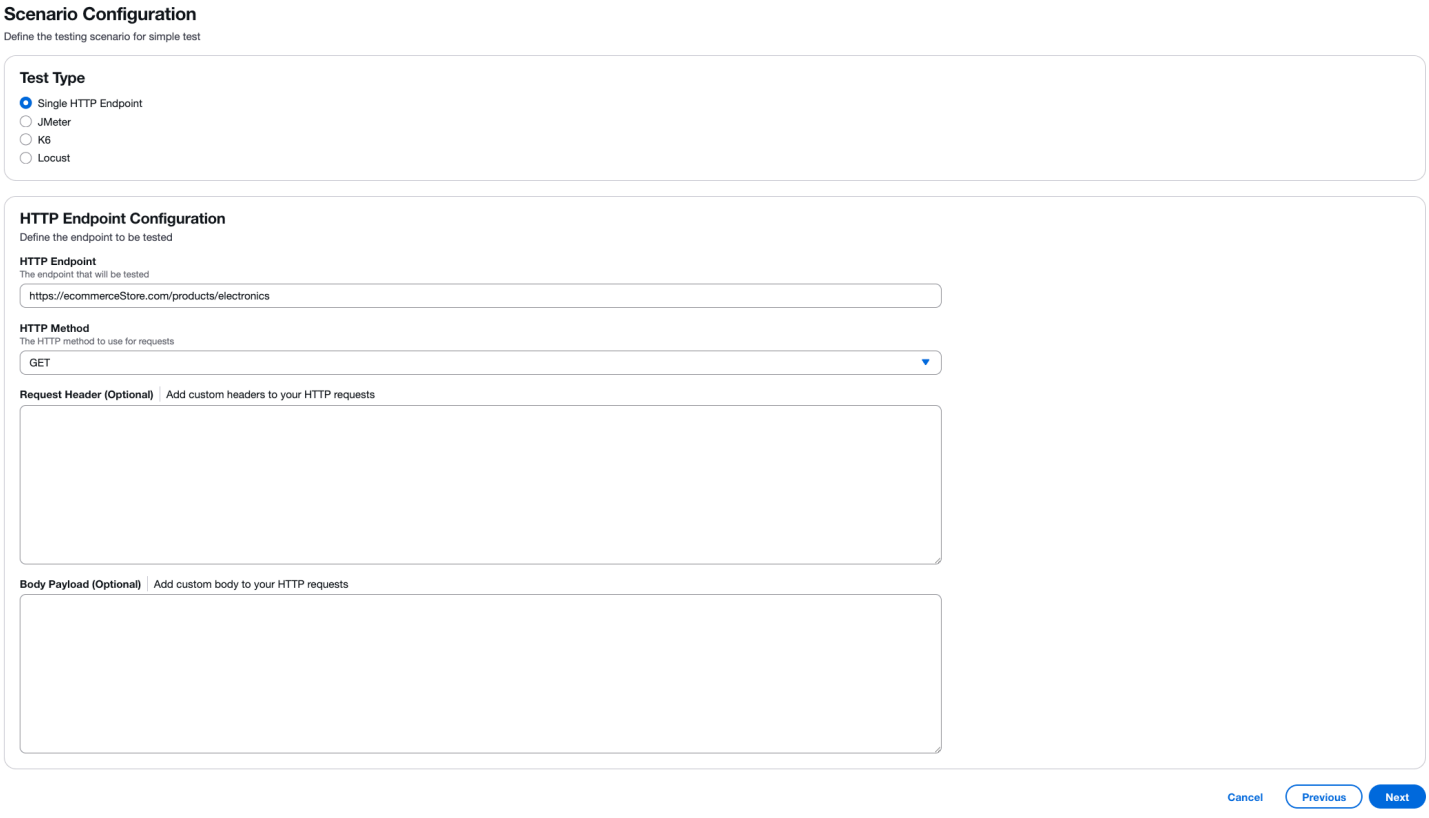
实时数据功能提供实时图表，每隔一秒钟聚合数据。有关更多信息，请参阅使用实时数据进行监控。

步骤 2：场景配置

定义特定的测试场景并选择您的首选测试框架。

测试类型选择

选择要执行的负载测试类型：



- 单个 HTTP 端点-使用简单的配置测试单个 API 端点或网页。

- JMeter-上传 JMeter 测试脚本（.jmx 文件或.zip 存档）。
- K6-上传 K6 测试脚本（.js 文件或.zip 存档）。
- Locust-上传 Locust 测试脚本（.py 文件或.zip 存档）。

HTTP 端点配置图片:: images/test-types.png [选择要运行的测试类型] 选择 “单个 HTTP 端点” 后，配置以下设置：

HTTP 终端节点（必填）

输入要测试的端点的完整 URL。例如 https://api.example.com/users 。确保可以从 AWS 基础设施访问终端节点。

HTTP 方法（必填）

为您的请求选择 HTTP 方法。默认值为 GET。其他选项包括POSTPUT、DELETE、PATCH、HEAD、和OPTIONS。

请求标头（可选）

在请求中添加自定义 HTTP 标头。常见的例子包括：

- `Content-Type: application/json`
- `Authorization: Bearer <token>`
- `User-Agent: LoadTest/1.0`

选择 “添加标题” 以包含多个标题。

车身有效载荷（可选）

为 POST 或 PUT 请求添加请求正文内容。支持 JSON、XML 或纯文本格式。例如：`{"userId": 123, "action": "test"}`。

测试框架脚本

使用 JMeter K6 或 Locust 时，请上传您的测试脚本文件或包含测试脚本和支持文件的.zip 存档。对于 JMeter，您可以将自定义插件包含在.zip 存档中的`/plugins`文件夹中。

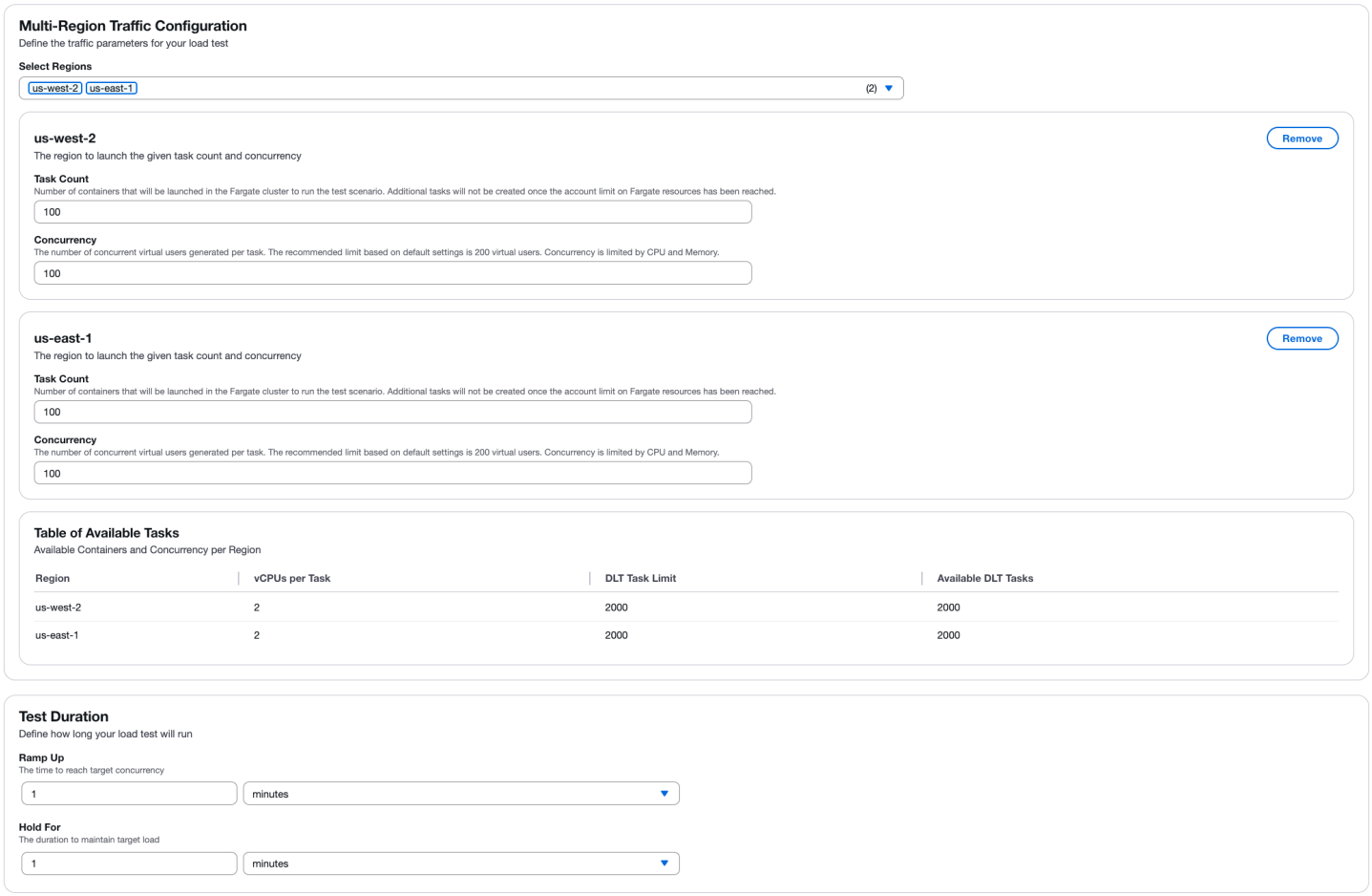
Important

尽管您的测试脚本（JMeter、K6 或 Locust）可以定义并发性（虚拟用户）、交易速率 (TPS)、加速时间和其他加载参数，但解决方案将使用您在测试创建期间在 Traffic Shape 屏幕

中指定的值来覆盖这些配置。Traffic Shape 配置控制测试执行的任务计数、并发度（每个任务的虚拟用户数）、加速持续时间和保持持续时间。

第 3 步：流量形状

配置测试期间流量的分配方式，包括多区域支持。



多区域流量配置

选择一个或多个 AWS 区域，按地理位置分配您的负载测试。对于每个选定的区域，请配置：

任务计数

在测试场景中，将在 Fargate 集群中启动的容器（任务）数量。一旦账户达到 “已达到 Fargate 资源” 限制，就不会创建其他任务。

并发

每个任务生成的并发虚拟用户数。建议的限制基于CPUs 每个任务 2 v 的默认设置。并发性受到 CPU 和内存资源的限制。

确定用户数量

容器在测试中可以支持的用户数量可以通过逐步增加用户数量和监控 Amazon 中的性能来确定 CloudWatch。一旦你观察到 CPU 和内存性能已接近极限，你就达到了容器在默认配置（2 个 vCPU 和 4 GB 内存）下可以支持的最大用户数。

校准过程

您可以使用以下示例开始确定测试的并发用户限制：

1. 创建不超过 200 个用户的测试。
2. 测试运行时，使用CloudWatch 控制台监控 CPU 和内存：
   a. 在左侧导航窗格的 “容器见解” 下，选择 “性能监控”。
   b. 在性能监控页面上，从左侧的下拉菜单中选择 ECS 集群。
   c. 从右侧的下拉菜单中，选择您的亚马逊弹性容器服务 (Amazon ECS) Container Service 集群。
3. 监控时，请注意CPU和内存。如果 CPU 未超过 75% 或内存未超过 85%（忽略一次性峰值），则可以对更多用户进行另一次测试。

如果测试未超过资源限制，请重复步骤 1-3。或者，您可以增加容器资源以允许更多的并发用户。但是，这会导致更高的成本。有关详细信息，请参阅《开发人员指南》。

> **Note**
>
> 为了获得准确的结果，在确定并发用户限制时，一次只能运行一个测试。所有测试都使用同一个集群， CloudWatch 容器见解会根据集群聚合性能数据。这会导致两个测试同时报告给 Conta CloudWatch iner Insights，从而导致单个测试的资源利用率指标不准确。

有关校准每个引擎的用户数的更多信息，请参阅文档中的校准 Taurus 测试。 BlazeMeter

> **Note**
>
> 该解决方案显示每个区域的可用容量信息，帮助您在可用限制范围内规划测试配置。

可用任务表

可用任务表显示每个选定区域的资源可用性：

- 区域-AWS 区域名称。
- v CPUs 每个任务- CPUs 分配给每个任务的虚拟数量（默认值：2）。
- DLT 任务限制-根据您的账户的 Fargate 限制可以创建的最大任务数（默认值：2000）。
- 可用的 DLT 任务-该地区当前可供使用的任务数（默认值：2000）。

Table of Available Tasks

Available Containers and Concurrency per Region

Region	vCPUs per Task	DLT Task Limit	Available DLT Tasks
us-west-2	2	2000	2000
us-east-1	2	2000	2000

要增加可用任务的数量或CPUs 每个任务的 v，请参阅开发者指南。

考试时长

定义您的负载测试将运行多长时间：

Ramp Up

达到目标并发的时间。在此期间，负载从 0 逐渐增加到配置的并发级别。

等一下

保持目标负荷的持续时间。在此期间，测试将以完全并发的状态继续进行。

步骤 4：审核并创建

在创建测试场景之前，请检查您的所有配置。验证:

- 常规设置（名称、描述、日程安排）。
- 场景配置（测试类型、端点或脚本）。

- 流量形状（任务、用户、时长、区域）。

查看完毕后，选择 “创建” 以保存您的测试场景。

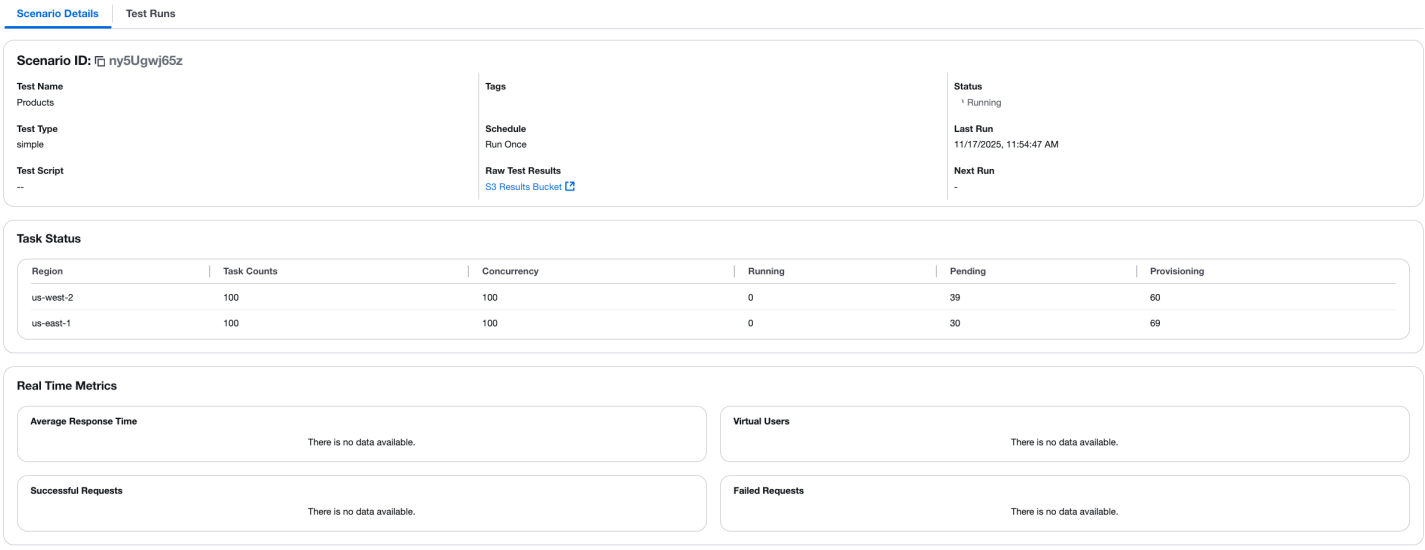
管理测试场景

创建测试场景后，您可以：

- 编辑-修改测试配置。常见使用案例包括：
  - 调整流量形态以实现所需的交易速率。
- 复制-复制现有测试场景以创建变体。常见使用案例包括：
  - 更新端点或添加 headers/body 参数。
  - 添加或修改测试脚本。
- 删除-删除不再需要的测试场景。

运行测试场景

创建测试场景后，您可以立即运行它，也可以将其安排在将来的特定时间运行。当您导航到正在运行的测试时，控制台会显示场景详细信息选项卡，其中包含实时任务状态和指标。



场景详细信息视图

场景详细信息选项卡显示有关您的测试的关键信息。每个区域的任务状态表实时信息。

任务状态表

任务状态表显示每个区域的实时信息：

- 区域-正在运行任务的 AWS 区域
- 任务计数-为该区域配置的任务总数
- 并发-每项任务的虚拟用户数
- 正在运行-当前正在执行测试的任务数
- 待处理-等待启动的任务数
- 置备-正在置备的任务数

测试执行工作流程

测试开始时，会出现以下工作流程：

1. 任务预置-该解决方案在指定的 AWS 区域预置容器（任务）。任务显示在 “配置” 列中。
2. 任务启动-解决方案会继续配置任务，直到每个区域达到目标任务数。任务从 “置备” 变为 “待处理”，再到 “正在运行”。
3. 流量生成-在解决方案配置一个区域中的所有任务后，它们会开始向您的目标终端节点发送流量。
4. 测试执行-测试在配置的持续时间（上升 + 保持时间）内运行。
5. 结果解析-测试结束后，后台解析作业会汇总和处理来自所有区域的结果。

测试运行状态

测试运行可以具有以下状态：

- 已计划-测试计划在将来运行。
- 正在运行-测试目前正在进行中。
- 已取消-用户取消了正在进行的测试运行。
- 错误-测试运行遇到错误。
- 完成-测试运行成功完成，结果已准备就绪。

使用实时数据进行监控

如果您在创建测试场景时启用了实时数据，则可以在测试运行时查看实时指标。实时指标部分显示四个图表，这些图表会随着测试的进行而不断更新，数据每隔一秒钟聚合。

图表描述

平均响应时间

显示每个区域处理的请求的平均响应时间（以秒为单位）。Y 轴以秒为单位显示响应时间，X 轴显示一天中的时间。每个区域在图例中用不同的颜色表示。

虚拟用户

显示每个区域中主动生成负载的并发虚拟用户数量。该图显示了虚拟用户在测试期间如何增加以及如何保持目标并发级别。

成功请求

显示一段时间内每个区域成功请求的累积计数。该图显示了成功请求的处理速率。

失败的请求

显示每个区域一段时间内失败请求的累积计数。计数较低或为零表示测试执行良好。

多区域可视化

在多个区域运行测试时，每个图表会同时显示所有区域的数据。每个图表底部的图例标识了哪种颜色代表每个区域（例如 us-west-2 和 us-east-1）。

技术实施

Fargate 任务的 CloudWatch 日志组包含一个用于捕获测试结果的订阅筛选器。检测到该模式后，Lambda 函数会对数据进行结构化并将其发布到 AWS IoT Core 主题中。Web 控制台订阅此主题并实时显示指标。

> **Note**
>
> 实时数据是短暂的，仅在测试运行时可用。Web 控制台最多可保存 5,000 个数据点，之后最旧的数据将替换为最新数据。如果页面刷新，图表将变为空白，并从下一个可用数据点开始。测试完成后，该解决方案会将结果数据存储在 DynamoDB 和 Amazon S3 中。如果尚无可用数据，则图表会显示 “没有可用数据”。

取消考试

您可以从 Web 控制台取消正在运行的测试。取消测试时，会出现以下工作流程：

1. 取消请求将发送到 `microservices` API
2. `microservices`API 调用 `task-canceler` Lambda 函数，该函数会停止所有当前启动的任务
3. 如果 `task-runner` Lambda 函数在初始取消调用后继续运行，则任务可能会继续短暂启动
4. `task-runner`Lambda 函数完成后，AWS Step Functions 继续`Cancel Test`执行该步骤，该步骤将再次运行 Lamb `task-canceler` da 函数以停止任何剩余的任务

> **Note**
>
> 取消的测试需要一段时间才能完成关闭过程，因为解决方案会终止所有容器。清理完所有资源后，测试状态将更改为 “已取消”。

浏览测试结果

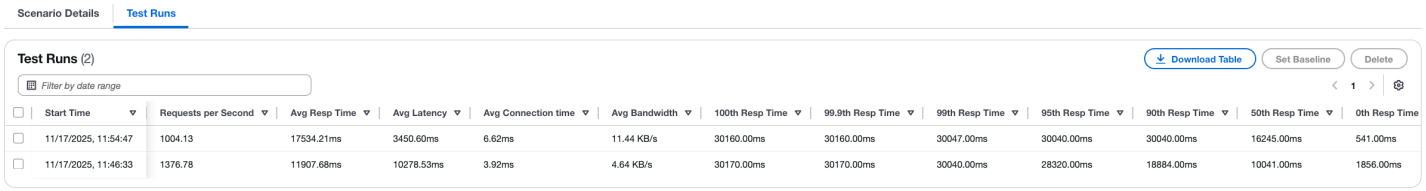
解析作业完成后，测试结果可供分析。该解决方案提供了全面的指标和工具，可帮助您了解应用程序在负载下的性能。

测试运行摘要指标

测试完成后，解决方案会生成包含以下指标的摘要：

- 平均响应时间-测试生成的所有请求的平均响应时间，以秒为单位。
- 平均延迟-测试生成的所有请求的平均延迟，以秒为单位。
- 平均连接时间-所有请求连接到主机所需的平均时间（以秒为单位）。
- 平均带宽-测试生成的所有请求的平均带宽。
- 总数-请求总数。
- 成功计数-成功请求的总数。
- 错误计数-错误总数。
- 每秒请求数-测试生成的所有请求的平均每秒请求数。
- 百分位数-响应时间百分位数，包括 p50（中位数）、p90、p95 和 p99，显示响应时间在所有请求中的分布情况。

测试运行表



测试运行表显示了某个场景的所有历史测试运行。你可以：

- 查看每次测试运行的摘要指标。
- 为性能比较设置基准测试。
- 将表格下载为 CSV 文件。
- 切换列以自定义视图。
- 选择测试运行以查看详细结果。

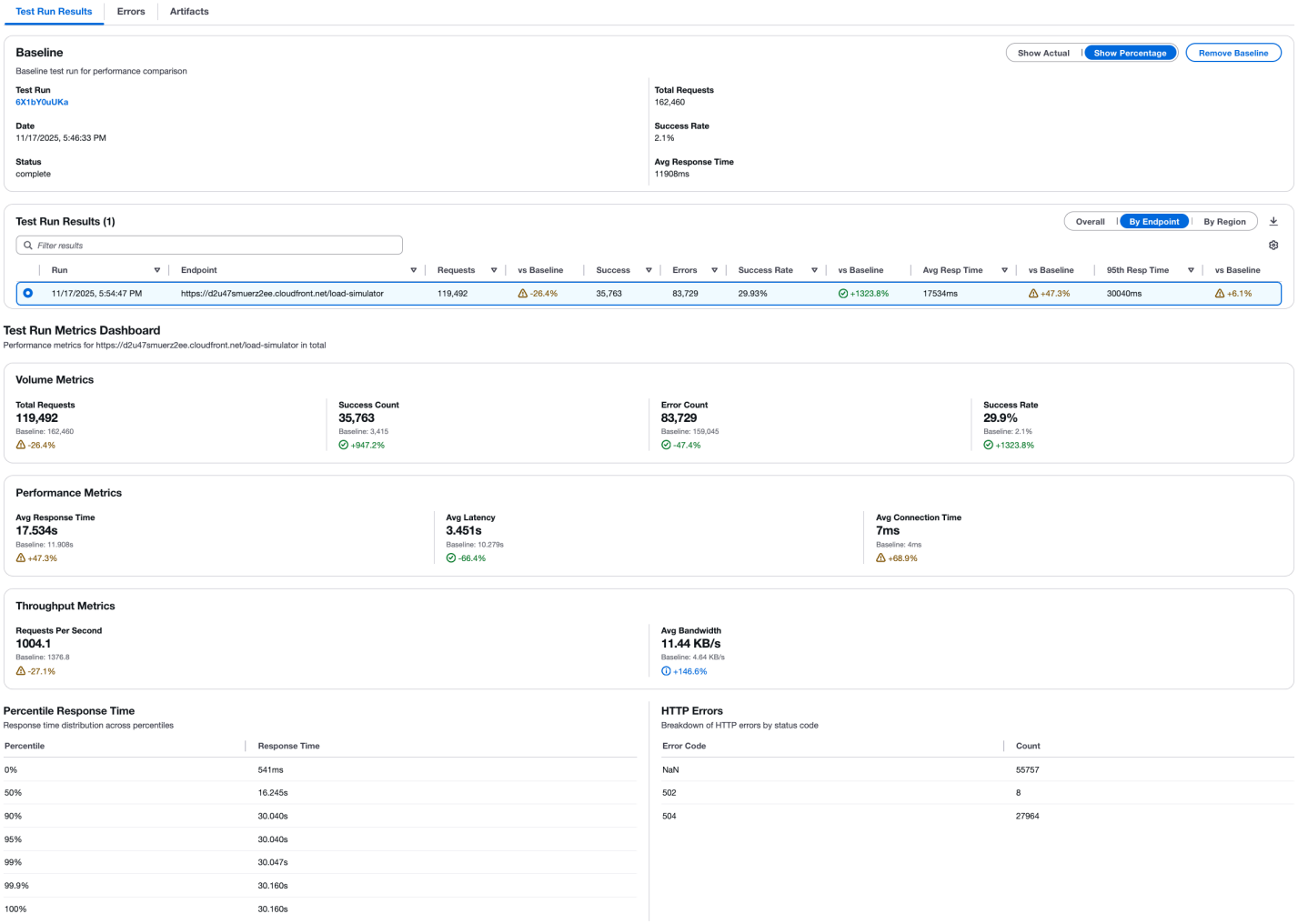
基线比较

您可以将测试运行指定为基准，以便将 future 的测试运行与之进行比较。设定基线时：

- 测试运行表显示每个指标与基线相比的百分比差异 (+/-%)。
- 基线指标可帮助您快速识别性能改进或回归情况。
- 您可以随时更改或清除基线。

详细的测试结果

选择测试运行会打开带有三个选项卡的详细结果视图：“测试运行结果”、“错误” 和 “对象”。



基线信息

如果设置了基准测试运行，它将显示在页面顶部。您可以选择 “显示实际”、“显示百分比” 或 “移除基线” 来控制基线比较的显示方式。

“测试运行结果” 表

结果表提供了具有以下功能的详细指标：

尺寸视图

使用尺寸按钮在三个视图之间切换：

- 总体-所有终端节点和区域的汇总结果

- 按端点划分-按各个端点细分的结果
- 按地区划分-按 AWS 地区细分的结果

操作按钮

- 显示实际值-显示实际指标值
- 显示百分比-显示与基线的百分比差异
- 移除基准-清除基线比较

数据导出和自定义

- 将结果表下载为 CSV 文件
- 切换列以自定义视图
- 对数据进行筛选和排序，将重点放在特定的指标上
- 对数据进行筛选和排序，将重点放在特定的指标上。

“错误” 选项卡

错误选项卡提供了详细的错误分析：

- 按类型查看错误计数。
- 查看按整体测试或端点汇总的错误。
- 识别失败请求中的模式。
- 对特定终端节点或区域的问题进行故障排除。

“对象” 选项卡

工件选项卡允许您访问测试运行期间生成的所有文件：

- 查看单个对象（日志、结果文件）。
- 下载特定的工件以进行离线分析。
- 将所有测试运行工件作为单个存档下载。

S3 结果结构

在 4.0 版本中，为了改进组织结构，S3 结果结构发生了变化：

- 新结构-`scenario-id/test-run-id/results-files`.
- 旧结构-在 4.0 版之前运行的测试会显示场景 ID 级别的所有结果文件。

> **Note**
>
> 测试结果显示在控制台中。您也可以直接在`Results`文件夹下的 Amazon S3 存储桶中访问原始测试结果。有关 Taurus 测试结果的更多信息，请参阅 Ta urus 用户手册中的生成测试报告。

MCP 服务器集成

如果您在解决方案部署期间部署了可选的 MCP 服务器组件，则可以将分布式负载测试解决方案与支持模型上下文协议的 AI 开发工具集成。MCP 服务器提供编程访问权限，以便通过 AI 助手检索、管理和分析负载测试。

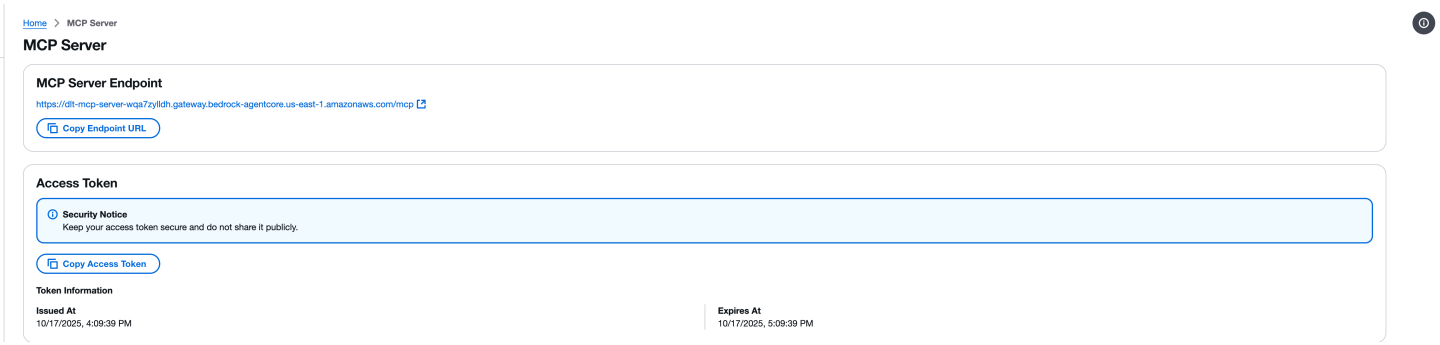
客户可以使用自己选择的客户端（Amazon Q、Claude 等）连接到 DLT MCP 服务器，每个客户端的配置说明略有不同。本节提供 MCP Inspector、Amazon Q CLI、Cline 和 Amazon Q Suite 的设置说明。

第 1 步：获取 MCP 端点和访问令牌

在配置任何 MCP 客户端之前，您需要从 DLT Web 控制台检索 MCP 服务器端点和访问令牌。

1. 导航到分布式负载测试 Web 控制台中的 MCP 服务器页面。
2. 找到 “MCP 服务器端点” 部分。
3. 使用复制终端节点 URL 按钮复制终端节点 URL。终端节点 URL 遵循以下格式：`https://[api-id].execute-api.[region].amazonaws.com/[stage]/gateway/backend-agent/sse/mcp`
4. 找到 “访问令牌” 部分。
5. 使用 “复制访问令牌” 按钮复制访问令牌。

> **Important**
>
> 请妥善保管您的访问令牌，不要将其公开共享。该令牌通过 MCP 接口提供对分布式负载测试解决方案的只读访问权限。



第 2 步：使用 MCP Inspector 进行测试

模型上下文协议提供了 MCP Inspector，这是一种直接连接到 MCP 服务器和调用工具的工具。这为在配置 AI 客户端之前测试 MCP 服务器连接提供了便捷的 UI 和示例网络请求。

> **Note**
>
> MCP Inspector 需要版本 0.17 或更高版本。也可以直接使用 JSON RPC 发出所有请求，但是 MCP Inspector 提供了一个更加用户友好的界面。

安装并启动 MCP Inspector

1. 如有必要，请安装 npm。
2. 运行以下命令启动 MCP Inspector：

```
npx @modelcontextprotocol/inspector
```

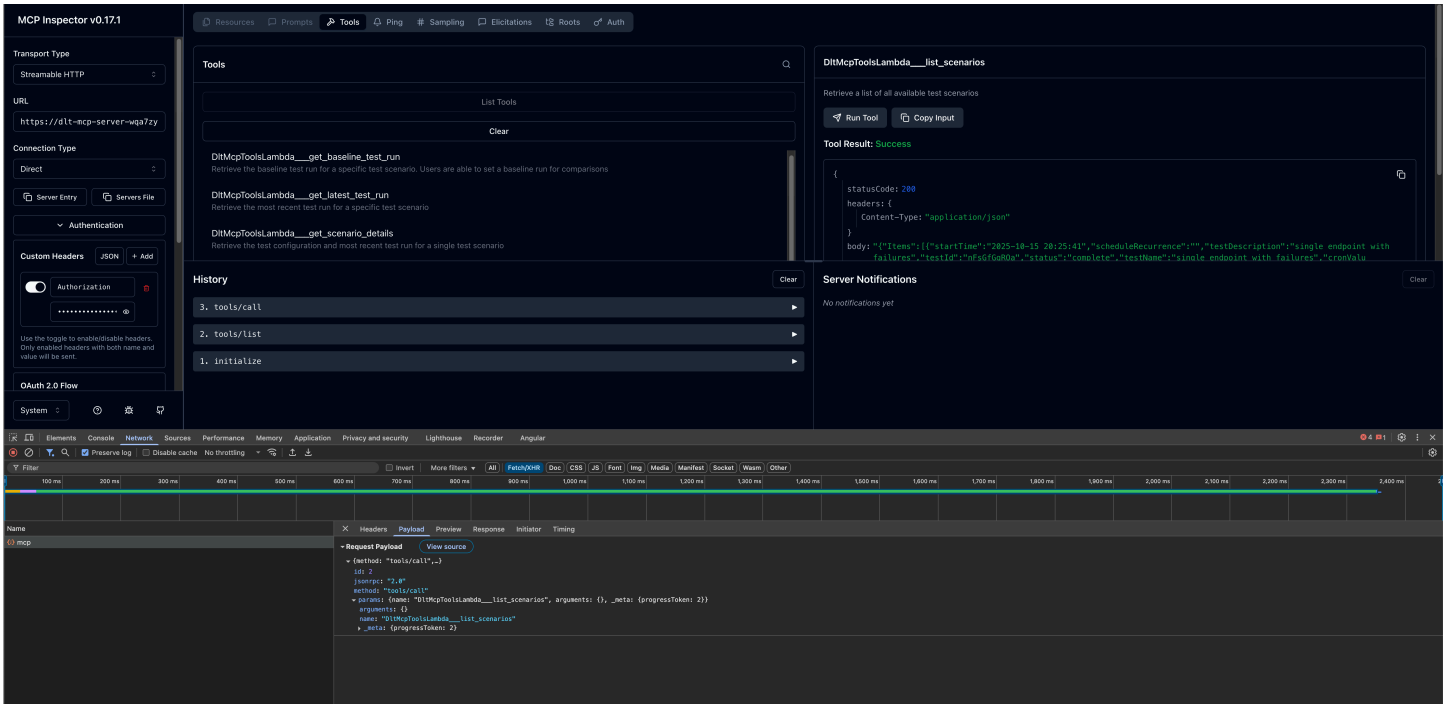
配置连接

1. 在 MCP Inspector 界面中，输入你的 MCP 服务器端点 URL。
2. 添加带有访问令牌的授权标头。
3. 单击 “连接” (Connect) 建立连接。

调用工具

连接后，您可以测试可用的 MCP 工具：

1. 浏览左侧面板中的可用工具列表。
2. 选择工具（例如，`list_scenarios`）。
3. 提供任何必需的参数。
4. 单击 “调用” 以执行该工具并查看响应。



步骤 3：配置 AI 开发客户端

验证 MCP 服务器与 MCP Inspector 的连接后，您可以配置首选的 AI 开发客户端。

Amazon Q CLI

Amazon Q CLI 通过 MCP 服务器集成，通过命令行访问人工智能辅助开发。

配置步骤

1. 编辑mcp.json配置文件。有关配置文件位置的更多信息，请参阅 Amazon Q 开发者用户指南中的配置远程 MCP 服务器。
2. 添加您的 DLT MCP 服务器配置：

```
{
  "mcpServers": {
    "dlt-mcp": {
      "type": "http",
      "url": "https://[api-id].execute-api.[region].amazonaws.com/[stage]/gateway/
backend-agent/sse/mcp",
      "headers": {
        "Authorization": "your_access_token_here"
      }
    }
```

```
  }
}
```

验证配置

1. 在终端中，键入q以启动 Amazon Q CLI。
2. 键/mcp入查看所有可用的 MCP 服务器。
3. 键入/tools以查看由dlt-mcp和其他已配置的 MCP 服务器提供的可用工具。
4. 验证是否dlt-mcp成功初始化。

Cline

Cline 是一款支持 MCP 服务器集成的人工智能编码助手。

配置步骤

1. 在 Cline 中，导航到管理 MCP 服务器 > 配置 > 配置 MC P 服务器。
2. 更新 cline_mcp_settings.json 文件：

```
{
  "mcpServers": {
    "dlt-mcp": {
      "type": "streamableHttp",
      "url": "https://[api-id].execute-api.[region].amazonaws.com/[stage]/gateway/
backend-agent/sse/mcp",
      "headers": {
        "Authorization": "your_access_token_here"
      }
    }
  }
}
```

3. 保存配置文件。
4. 重新启动 Cline 以应用更改。

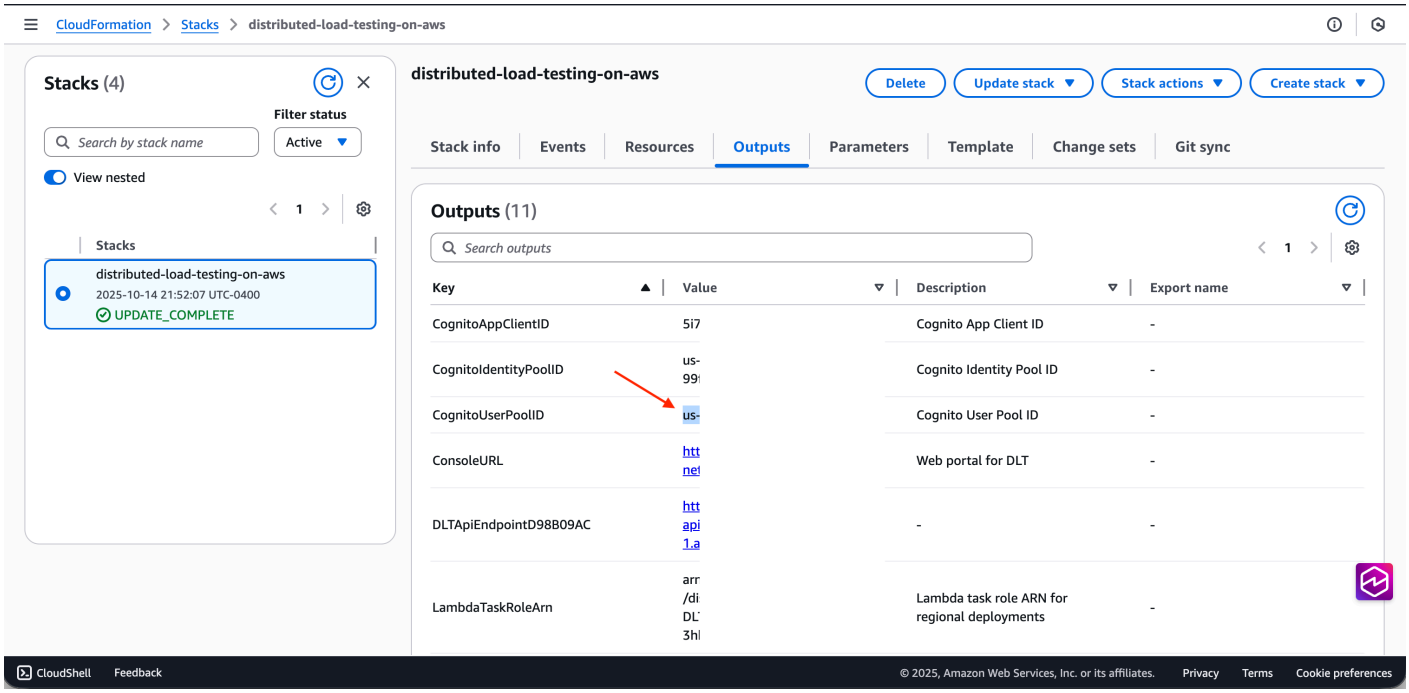
亚马逊 Q Suite

Amazon Q Suite 提供了一个全面的人工智能助手平台，支持 MCP 服务器操作。

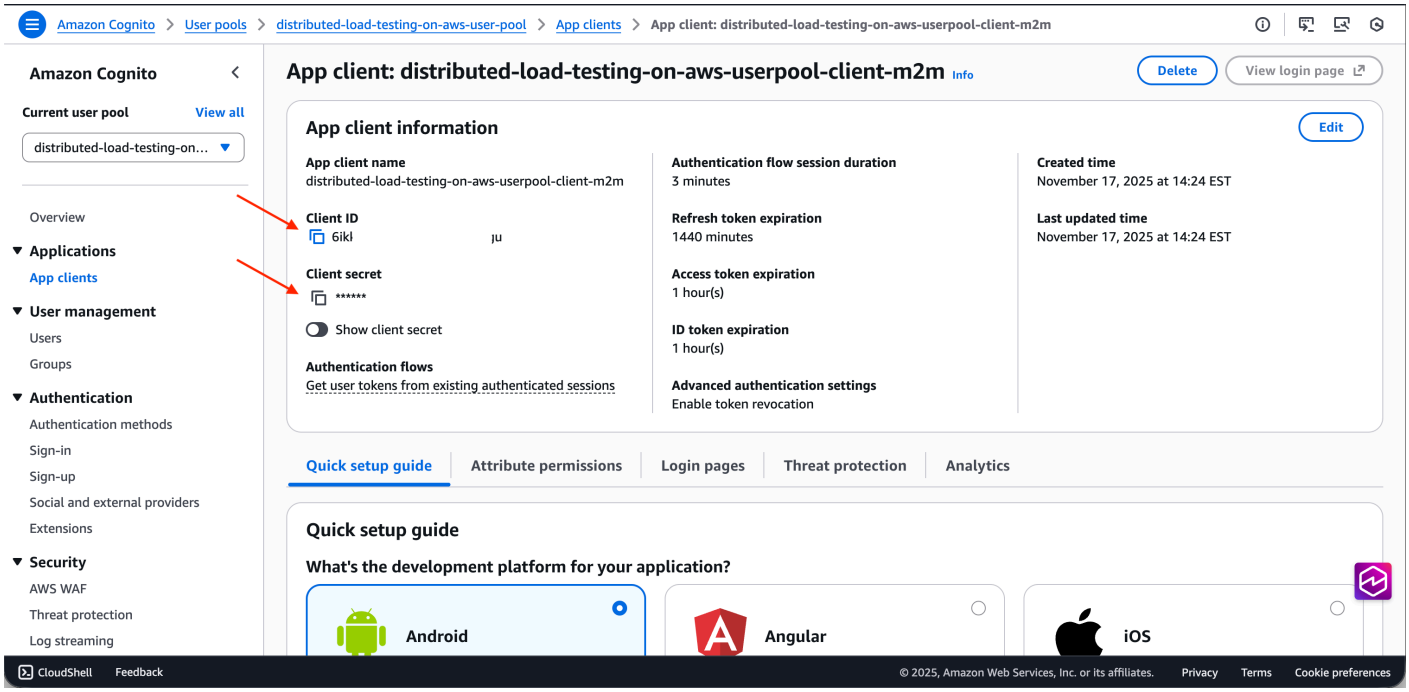
先决条件

在 Amazon Q Suite 中配置 MCP 服务器之前，您需要从 DLT 部署的 Cognito 用户池中检索 OAuth 证书：

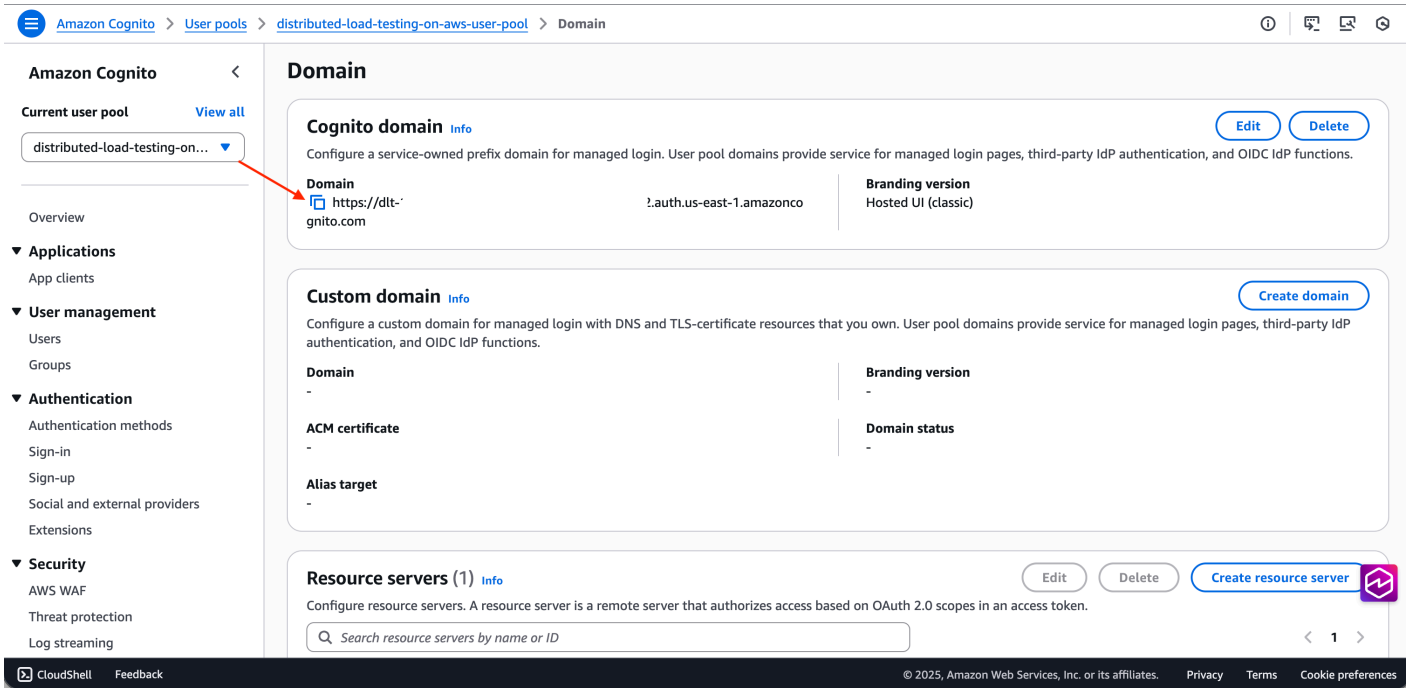
1. 导航到 AWS CloudFormation 控制台。
2. 选择分布式负载测试堆栈。
3. 在输出选项卡中，找到并复制与您的 DLT 部署关联的 Cognito 用户池 ID。



4. 导航到 Amazon Cognito 控制台。
5. 使用 CloudFormation 输出中的用户池 ID 选择用户池。
6. 在左侧导航栏中，选择应用程序集成 > 应用程序客户端。



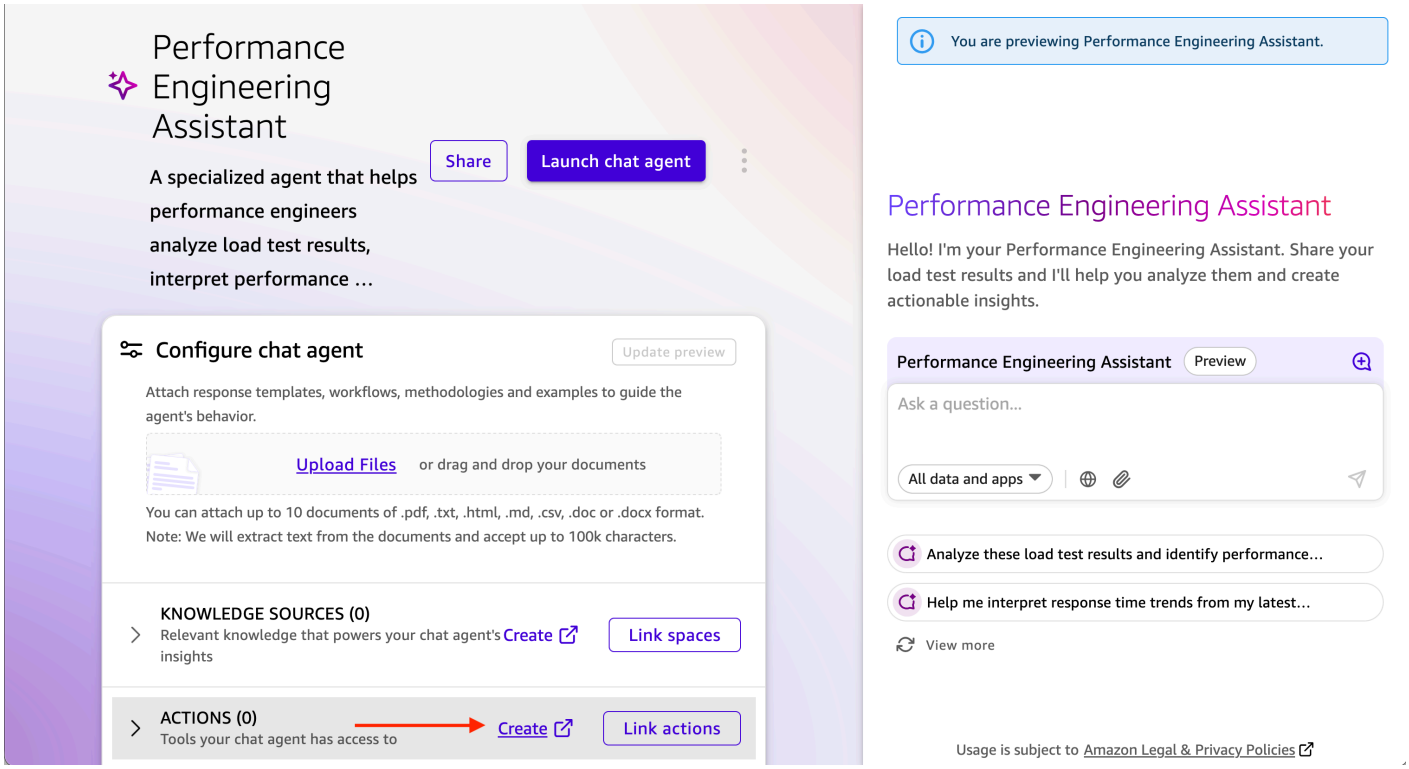
7. 找到名称以 m2m (machine-to-machine) 结尾的应用程序客户端。
8. 复制客户端 ID 和客户机密钥。
9. 从 “域” 选项卡获取用户池域。

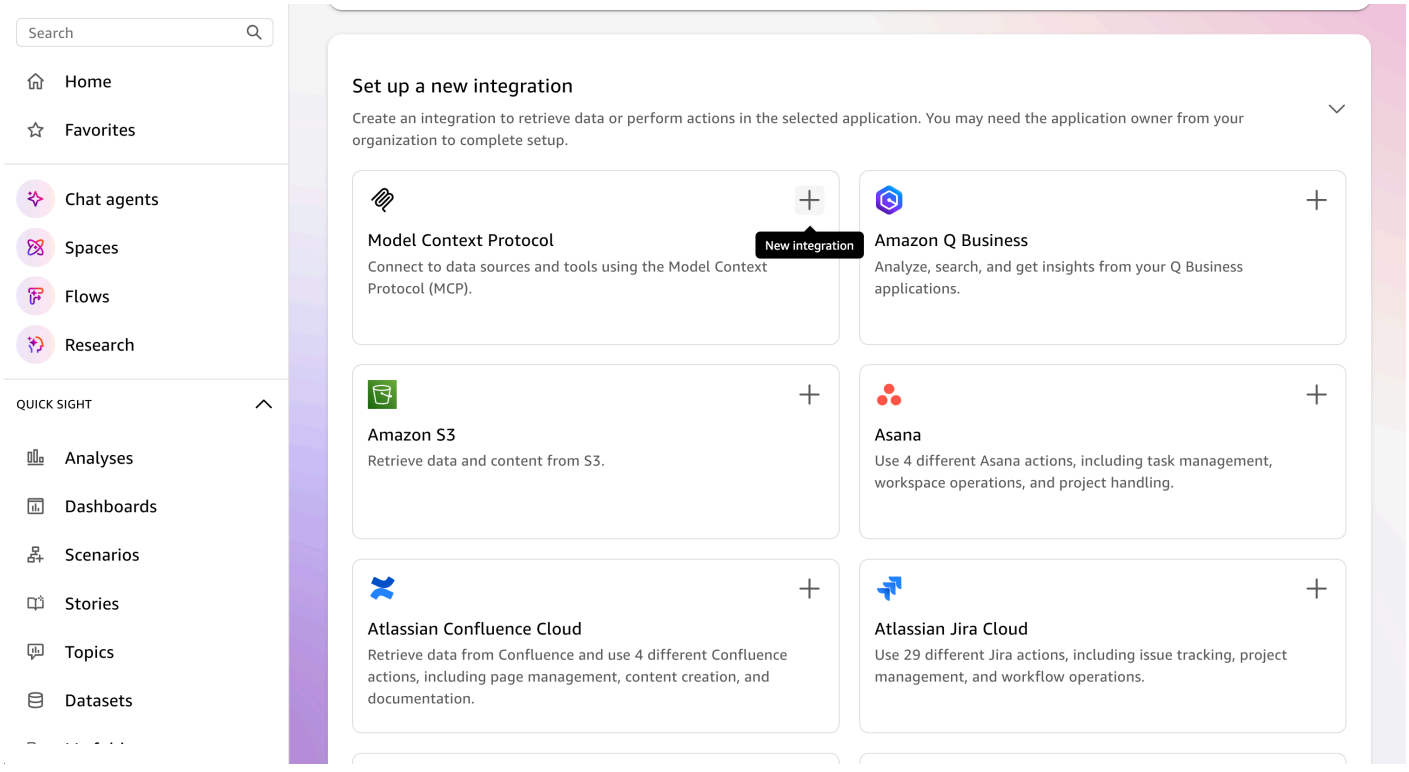


10通过追加/oauth2/token到域名末尾来构造令牌端点 URL。

配置步骤

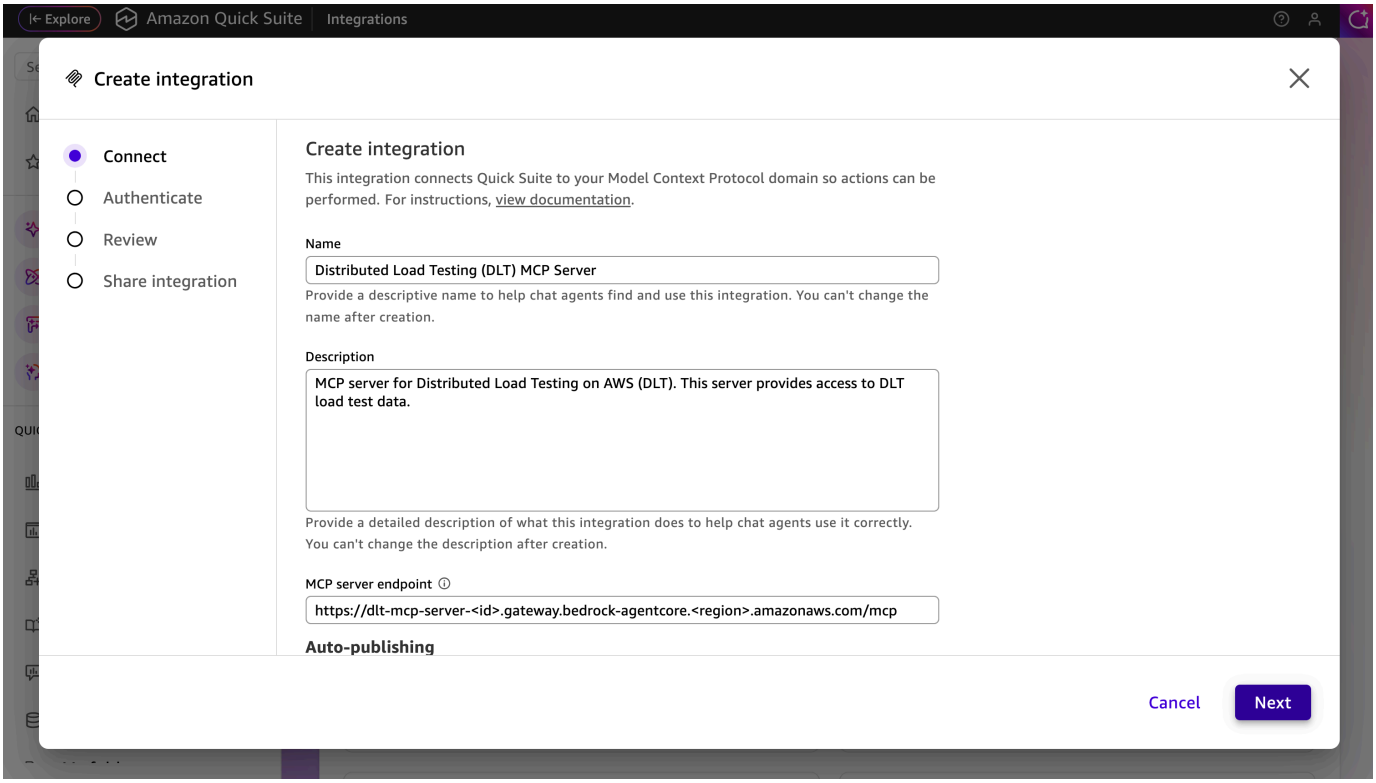
1. 在 Amazon Q Suite 中，创建新代理或选择现有代理。
2. 添加描述如何与 DLT MCP 服务器交互的代理提示符。
3. 添加新操作并选择 MCP 服务器操作。



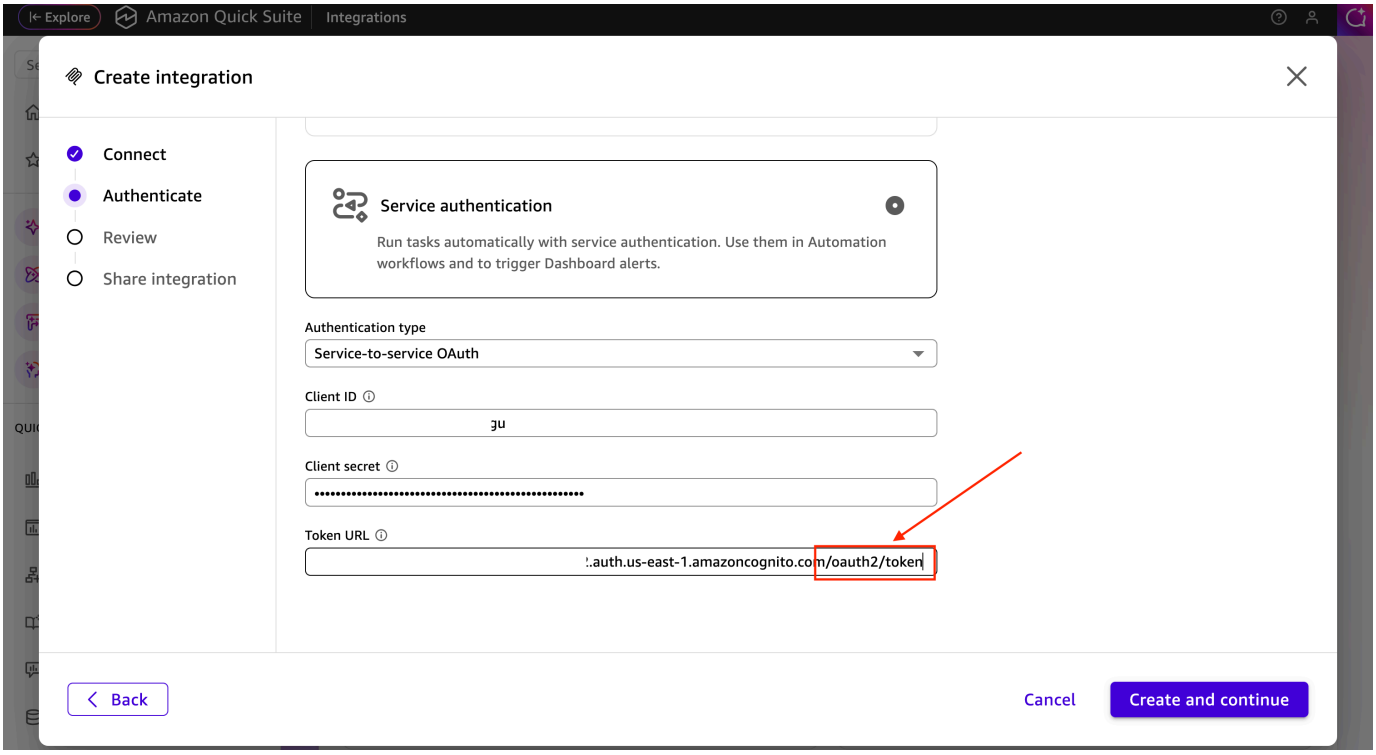


4. 配置 MCP 服务器详细信息：

- MCP 服务器网址：你的 DLT MCP 端点



- 身份验证类型：基于服务的身份验证
- 令牌端点：您的 Cognito 令牌端点网址
- 客户端 ID：来自 m2m 应用程序客户端的客户端 ID
- 客户端密钥：来自 m2m 应用程序客户端的客户端密钥



5. 保存 MCP 服务器操作配置。
6. 将新的 MCP 服务器操作添加到您的代理。

启动并测试代理

1. 在 Amazon Q Suite 中启动代理。
2. 使用自然语言提示与代理开始对话。
3. 代理将使用 MCP 工具检索和分析您的负载测试数据。

提示示例

以下示例演示如何与 AI 助手交互以通过 MCP 接口分析负载测试数据。自定义测试 IDs、日期范围和标准，以满足您的特定测试需求。

有关可用的 MCP 工具及其参数的详细信息，请参阅《开发人员指南》中的 MCP 工具规范。

简单的测试结果查询

与 MCP 服务器的自然语言交互可以很简单，`Show me the load tests that have completed in the last 24 hours with their associated completion status`也可以更具描述性，例如

```
Use list_scenarios to find my load tests. Then use get_latest_test_run to show me the
 basic execution data and performance metrics for the most recent test. If the results
 look concerning, also get the detailed performance metrics using get_test_run.
```

采用渐进式披露的交互式绩效分析

```
I need to analyze my load test performance, but I'm not sure which specific tests to
 focus on. Please help me by:

1. First, use list_scenarios to show me available test scenarios
2. Ask me which tests I want to analyze based on the list you show me
3. For my selected tests, use list_test_runs to get the test run history
4. Then use get_test_run with the test_run_id to get detailed response times,
 throughput, and error rates
5. If I want to compare tests, use get_baseline_test_run to compare against the
 baseline
6. If there are any issues, use get_test_run_artifacts to help me understand what went
 wrong

Please guide me through this step by step, asking for clarification whenever you need
 more specific information.
```

生产就绪性验证

```
Help me validate if my API is ready for production deployment:

1. Use list_scenarios to find recent test scenarios
2. For the most recent test scenario, use get_latest_test_run to get basic execution
 data
3. Use get_test_run with that test_run_id to get detailed response times, error rates,
 and throughput
4. Use get_scenario_details with the test_id to show me what load patterns and
 endpoints were tested
5. If I have a baseline, use get_baseline_test_run to compare current results with the
 baseline
6. Provide a clear go/no-go recommendation based on the performance data
7. If there are any concerns, use get_test_run_artifacts to help identify potential
 issues

My SLA requirements are: response time under [X]ms, error rate under [Y]%.
```

性能趋势分析

```
Analyze the performance trend for my load tests over the past [TIME_PERIOD]:

1. Use list_scenarios to get all test scenarios
2. For each scenario, use list_test_runs with start_date and end_date to get tests from
 that period
3. Use get_test_run for the key test runs to get detailed metrics
4. Use get_baseline_test_run to compare against the baseline
5. Identify any significant changes in response times, error rates, or throughput
6. If you detect performance degradation, use get_test_run_artifacts on the problematic
 tests to help identify causes
7. Present the trend analysis in a clear format showing whether performance is
 improving, stable, or degrading

Focus on completed tests and limit results to [N] tests if there are too many.
```

排除失败的测试故障

```
Help me troubleshoot my failed load tests:

1. Use list_scenarios to find test scenarios
2. For each scenario, use list_test_runs to find recent test runs
3. Use get_test_run with the test_run_id to get the basic execution data and failure
 information
4. Use get_test_run_artifacts to get detailed error messages and logs
5. Use get_scenario_details to understand what was being tested when it failed
6. If I have a similar test that passed, use get_baseline_test_run to identify
 differences
7. Summarize the causes of failure and suggest next steps for resolution

Show me the most recent [N] failed tests from the past [TIME_PERIOD].
```

开发者指南

本节提供解决方案的源代码和其他自定义设置。

源代码

访问我们的GitHub 存储库，下载此解决方案的模板和脚本，并与其他人共享您的自定义设置。

Maintenance

此解决方案使用固定版本的 Docker 镜像，这些镜像与每个解决方案版本相对应。默认情况下，自动更新处于禁用状态，这使您可以完全控制何时以及将哪些版本更新应用于您的部署。AWS 解决方案团队使用 Amazon ECR 增强扫描来检测基础映像和已安装软件包中的常见漏洞和风险敞口 (CVEs)。如果可能，团队会发布带有相同版本标签的修补镜像，以便在 CVEs 不破坏与已发布解决方案版本的兼容性的前提下进行解析。

在同一个次要版本上修补图像时，稳定标签会自动更新，并以该格式<solution-version>_<date-of-fix>创建额外的图像标签。如果发布了主要版本或次要版本，则必须执行完整堆栈更新才能获得最新的映像版本，因为稳定标签会递增以匹配解决方案版本。

如果您在部署期间选择自动更新，则对映像的更改（包括 CVE 补丁和次要错误修复）将自动应用到最新的匹配次要版本。

版本

默认情况下，此解决方案在部署时禁用自动更新。这意味着容器映像版本已锁定到与您部署的解决方案版本相匹配的特定版本，从而使您可以完全控制版本更新。

如果您在 CloudFormation 部署期间选择 “是” 来启用自动更新，则该解决方案将自动接收安全补丁和不间断的次要错误修复，直到最新的匹配次要版本。例如，如果您在启用自动更新的情况下部署 4.0.0 版，则将收到高达 4.0.x 的更新，但不会收到 4.1.0 或更高版本的更新。

要手动控制容器映像版本，您可以编辑任务定义以使用标记版本格式指定特定的映像版本。这样，无论自动更新设置如何，您都可以固定到特定的图像版本。

容器镜像自定义

此解决方案使用由 AWS 管理的公共 Amazon Elastic Container Registry (Amazon ECR) 图像存储库来存储用于运行已配置测试的映像。如果您想自定义容器镜像，可以重建镜像并将其推送到您自己的 AWS 账户中的 ECR 镜像存储库中。

如果要自定义此解决方案，则可以使用默认容器镜像，或者根据需要编辑此容器。如果您自定义解决方案，请在构建自定义解决方案之前使用以下代码示例声明环境变量。

```
#!/bin/bash
export REGION=aws-region-code # the AWS region to launch the solution (e.g. us-east-1)
export BUCKET_PREFIX=my-bucket-name # prefix of the bucket name without the region code
export BUCKET_NAME=$BUCKET_PREFIX-$REGION # full bucket name where the code will reside
export SOLUTION_NAME=my-solution-name
export VERSION=my-version # version number for the customized code
export PUBLIC_ECR_REGISTRY=public.ecr.aws/awssolutions/distributed-load-testing-on-
aws-load-tester # replace with the container registry and image if you want to use a
 different container image export PUBLIC_ECR_TAG=v3.1.0 # replace with the container
 image tag if you want to use a different container image
```

如果您选择自定义容器镜像，则可以将其托管在私有镜像存储库或您的 AWS 账户中的公共镜像存储库中。图像资源位于代码库中的`deployment/ecr/distributed-load-testing-on-aws-load-tester`目录中。

您可以构建映像并将其推送到主机目的地。

- 有关私有 Amazon ECR 存储库和映像的信息，请参阅《亚马逊 ECR 用户指南》中的 A mazon ECR 私有存储库和私有镜像。
- 有关公共 Amazon ECR 存储库和镜像，请参阅《亚马逊 ECR 公共用户指南》中的 Amazon ECR 公共存储库和公共镜像。

创建自己的映像后，可以在构建自定义解决方案之前声明以下环境变量。

```
#!/bin/bash
export PUBLIC_ECR_REGISTRY=YOUR_ECR_REGISTRY_URI # e.g. YOUR_ACCOUNT_ID.dkr.ecr.us-
east-1.amazonaws.com/YOUR_IMAGE_NAME
export PUBLIC_ECR_TAG=YOUR_ECR_TAG # e.g. latest, v3.4.0
```

以下示例显示了容器文件。

```
FROM public.ecr.aws/amazonlinux/amazonlinux:2023-minimal

RUN dnf update -y && \
    dnf install -y python3.11 python3.11-pip java-21-amazon-corretto bc procps jq
 findutils unzip && \
    dnf clean all

ENV PIP_INSTALL="pip3.11 install --no-cache-dir"


# install bzt
RUN $PIP_INSTALL --upgrade bzt awscli setuptools==78.1.1 h11 urllib3==2.2.2 && \
    $PIP_INSTALL --upgrade bzt
COPY ./.bzt-rc /root/.bzt-rc
RUN chmod 755 /root/.bzt-rc

# install bzt tools
RUN bzt -install-tools -o modules.install-
checker.exclude=selenium,gatling,tsung,siege,ab,k6,external-results-
loader,locust,junit,testng,rspec,mocha,nunit,xunit,wdio,robot,newman
RUN rm -rf /root/.bzt/selenium-taurus
RUN mkdir /bzt-configs /tmp/artifacts
ADD ./load-test.sh /bzt-configs/
ADD ./*.jar /bzt-configs/
ADD ./*.py /bzt-configs/

RUN chmod 755 /bzt-configs/load-test.sh
RUN chmod 755 /bzt-configs/ecslistener.py
RUN chmod 755 /bzt-configs/ecscontroller.py
RUN chmod 755 /bzt-configs/jar_updater.py
RUN python3.11 /bzt-configs/jar_updater.py

# Remove jar files from /tmp
RUN rm -rf /tmp/jmeter-plugins-manager-1* && \
    rm -rf /usr/local/lib/python3.11/site-packages/setuptools-65.5.0.dist-info && \
    rm -rf /usr/local/lib/python3.11/site-packages/urllib3-1.26.17.dist-info

# Add settings file to capture the output logs from bzt cli
RUN mkdir -p /etc/bzt.d && echo '{"settings": {"artifacts-dir": "/tmp/artifacts"}}' > /
etc/bzt.d/90-artifacts-dir.json

WORKDIR /bzt-configs
ENTRYPOINT ["./load-test.sh"]
```

除容器文件外，该目录还包含以下 bash 脚本，该脚本可在运行 Taurus/Blazemeter 程序之前从 Amazon S3 下载测试配置。

```
#!/bin/bash

# set a uuid for the results xml file name in S3
UUID=$(cat /proc/sys/kernel/random/uuid)
pypid=0
echo "S3_BUCKET:: ${S3_BUCKET}"
echo "TEST_ID:: ${TEST_ID}"
echo "TEST_TYPE:: ${TEST_TYPE}"
echo "FILE_TYPE:: ${FILE_TYPE}"
echo "PREFIX:: ${PREFIX}"
echo "UUID:: ${UUID}"
echo "LIVE_DATA_ENABLED:: ${LIVE_DATA_ENABLED}"
echo "MAIN_STACK_REGION:: ${MAIN_STACK_REGION}"

cat /proc/self/cgroup
TASK_ID=$(grep -oE '[a-f0-9]{32}' /proc/self/cgroup | head -n 1)
echo $TASK_ID

sigterm_handler() {
  if [ $pypid -ne 0 ]; then
    echo "container received SIGTERM."
    kill -15 $pypid
    wait $pypid
    exit 143 #128 + 15
  fi
}
trap 'sigterm_handler' SIGTERM

echo "Download test scenario"
aws s3 cp s3://$S3_BUCKET/test-scenarios/$TEST_ID-$AWS_REGION.json test.json --region
 $MAIN_STACK_REGION

# Set the default log file values to jmeter
LOG_FILE="jmeter.log"
OUT_FILE="jmeter.out"
ERR_FILE="jmeter.err"
KPI_EXT="jtl"

# download JMeter jmx file
if [ "$TEST_TYPE" != "simple" ]; then
```

```
  # setting the log file values to the test type
  LOG_FILE="${TEST_TYPE}.log"
  OUT_FILE="${TEST_TYPE}.out"
  ERR_FILE="${TEST_TYPE}.err"

  # set variables based on TEST_TYPE
  if [ "$TEST_TYPE" == "jmeter" ]; then
    EXT="jmx"
    TYPE_NAME="JMeter"
    # Copy *.jar to JMeter library path. See the Taurus JMeter path: https://
gettaurus.org/docs/JMeter/
    JMETER_LIB_PATH=`find ~/.bzt/jmeter-taurus -type d -name "lib"`
    echo "cp $PWD/*.jar $JMETER_LIB_PATH"
    cp $PWD/*.jar $JMETER_LIB_PATH
  elif [ "$TEST_TYPE" == "k6" ]; then
    curl --output /tmp/artifacts/k6.rpm https://dl.k6.io/rpm/x86_64/k6-v0.58.0-
amd64.rpm
    rpm -ivh /tmp/artifacts/k6.rpm
    dnf install -y k6
    rm -rf /tmp/artifacts/k6.rpm
    EXT="js"
    KPI_EXT="csv"
    TYPE_NAME="K6"
  elif [ "$TEST_TYPE" == "locust" ]; then
    EXT="py"
    TYPE_NAME="Locust"

  fi

  if [ "$FILE_TYPE" != "zip" ]; then
    aws s3 cp s3://$S3_BUCKET/public/test-scenarios/$TEST_TYPE/$TEST_ID.$EXT ./ --
region $MAIN_STACK_REGION
  else
    aws s3 cp s3://$S3_BUCKET/public/test-scenarios/$TEST_TYPE/$TEST_ID.zip ./ --region
 $MAIN_STACK_REGION
    unzip $TEST_ID.zip
    echo "UNZIPPED"
    ls -l

    # If zip and locust, make sure to pick locustfile
    if [ "$TEST_TYPE" != "locust" ]; then
      TEST_SCRIPT=$(find . -name "*.${EXT}" | head -n 1)
    else
      TEST_SCRIPT=$(find . -name "locustfile.py" | head -n 1)
```

```
    fi
    # only looks for the first test script file.
    TEST_SCRIPT=`find . -name "*.${EXT}" | head -n 1`
    echo $TEST_SCRIPT
    if [ -z "$TEST_SCRIPT" ]; then
      echo "There is no test script (.${EXT}) in the zip file."
      exit 1
    fi

    sed -i -e "s|$TEST_ID.$EXT|$TEST_SCRIPT|g" test.json

    # copy bundled plugin jars to jmeter extension folder to make them available to
 jmeter
    BUNDLED_PLUGIN_DIR=`find $PWD -type d -name "plugins" | head -n 1`
    # attempt to copy only if a /plugins folder is present in upload
    if [ -z "$BUNDLED_PLUGIN_DIR" ]; then
      echo "skipping plugin installation (no /plugins folder in upload)"
    else
      # ensure the jmeter extensions folder exists
      JMETER_EXT_PATH=`find ~/.bzt/jmeter-taurus -type d -name "ext"`
      if [ -z "$JMETER_EXT_PATH" ]; then
        # fail fast - if plugins bundled they will be needed for the tests
        echo "jmeter extension path (~/.bzt/jmeter-taurus/**/ext) not found - cannot
 install bundled plugins"
        exit 1
      fi
      cp -v $BUNDLED_PLUGIN_DIR/*.jar $JMETER_EXT_PATH
    fi
  fi
fi

#Download python script
if [ -z "$IPNETWORK" ]; then
    python3.11 -u $SCRIPT  $TIMEOUT &
    pypid=$!
    wait $pypid
    pypid=0
else
    aws s3 cp s3://$S3_BUCKET/Container_IPs/${TEST_ID}_IPHOSTS_${AWS_REGION}.txt ./ --
region $MAIN_STACK_REGION
    export IPHOSTS=$(cat ${TEST_ID}_IPHOSTS_${AWS_REGION}.txt)
    python3.11 -u $SCRIPT $IPNETWORK $IPHOSTS
fi
```

```
echo "Running test"

stdbuf -i0 -o0 -e0 bzt test.json -o modules.console.disable=true | stdbuf -i0 -o0 -e0
 tee -a result.tmp | sed -u -e "s|^|$TEST_ID $LIVE_DATA_ENABLED |"
CALCULATED_DURATION=`cat result.tmp | grep -m1 "Test duration" | awk -F ' ' '{ print
 $5 }' | awk -F ':' '{ print ($1 * 3600) + ($2 * 60) + $3 }'`

# upload custom results to S3 if any
# every file goes under $TEST_ID/$PREFIX/$UUID to distinguish the result correctly
if [ "$TEST_TYPE" != "simple" ]; then
  if [ "$FILE_TYPE" != "zip" ]; then
    cat $TEST_ID.$EXT | grep filename > results.txt
  else
    cat $TEST_SCRIPT | grep filename > results.txt
  fi

  if [ -f results.txt ]; then
    sed -i -e 's/<stringProp name="filename">//g' results.txt
    sed -i -e 's/<\/stringProp>//g' results.txt
    sed -i -e 's/ //g' results.txt

    echo "Files to upload as results"
    cat results.txt

    files=(`cat results.txt`)
    extensions=()
    for f in "${files[@]}"; do
      ext="${f##*.}"
      if [[ ! " ${extensions[@]} " =~ " ${ext} " ]]; then
        extensions+=("$ext")
      fi
    done

    # Find all files in the current folder with the same extensions
    all_files=()
    for ext in "${extensions[@]}"; do
      for f in *."$ext"; do
        all_files+=("$f")
      done
    done

    for f in "${all_files[@]}"; do
      p="s3://$S3_BUCKET/results/$TEST_ID/${TYPE_NAME}_Result/$PREFIX/$UUID/$f"
      if [[ $f = /* ]]; then
```

```
        p="s3://$S3_BUCKET/results/$TEST_ID/${TYPE_NAME}_Result/$PREFIX/$UUID$f"
      fi

        echo "Uploading $p"
        aws s3 cp $f $p --region $MAIN_STACK_REGION
    done
    fi
fi

if [ -f /tmp/artifacts/results.xml ]; then

  # Insert the Task ID at the same level as <FinalStatus>
  curl -s $ECS_CONTAINER_METADATA_URI_V4/task
  Task_CPU=$(curl -s $ECS_CONTAINER_METADATA_URI_V4/task | jq '.Limits.CPU')
  Task_Memory=$(curl -s $ECS_CONTAINER_METADATA_URI_V4/task | jq '.Limits.Memory')
  START_TIME=$(curl -s "$ECS_CONTAINER_METADATA_URI_V4/task" | jq -r
 '.Containers[0].StartedAt')
  # Convert start time to seconds since epoch
  START_TIME_EPOCH=$(date -d "$START_TIME" +%s)
  # Calculate elapsed time in seconds
  CURRENT_TIME_EPOCH=$(date +%s)
  ECS_DURATION=$((CURRENT_TIME_EPOCH - START_TIME_EPOCH))


  sed -i.bak 's/<\/FinalStatus>/<TaskId>'"$TASK_ID"'<\/TaskId><\/FinalStatus>/' /tmp/
artifacts/results.xml
  sed -i 's/<\/FinalStatus>/<TaskCPU>'"$Task_CPU"'<\/TaskCPU><\/FinalStatus>/' /tmp/
artifacts/results.xml
  sed -i 's/<\/FinalStatus>/<TaskMemory>'"$Task_Memory"'<\/TaskMemory><\/
FinalStatus>/' /tmp/artifacts/results.xml
  sed -i 's/<\/FinalStatus>/<ECSDuration>'"$ECS_DURATION"'<\/ECSDuration><\/
FinalStatus>/' /tmp/artifacts/results.xml

  echo "Validating Test Duration"
  TEST_DURATION=$(grep -E '<TestDuration>[0-9]+.[0-9]+</TestDuration>' /tmp/artifacts/
results.xml | sed -e 's/<TestDuration>//' | sed -e 's/<\/TestDuration>//')

  if (( $(echo "$TEST_DURATION > $CALCULATED_DURATION" | bc -l) )); then
    echo "Updating test duration: $CALCULATED_DURATION s"
    sed -i.bak.td 's/<TestDuration>[0-9]*\.[0-9]*<\/TestDuration>/
<TestDuration>'"$CALCULATED_DURATION"'<\/TestDuration>/' /tmp/artifacts/results.xml
  fi

  if [ "$TEST_TYPE" == "simple" ]; then
```

```
    TEST_TYPE="jmeter"
  fi

  echo "Uploading results, bzt log, and JMeter log, out, and err files"
  aws s3 cp /tmp/artifacts/results.xml s3://$S3_BUCKET/results/${TEST_ID}/${PREFIX}-
${UUID}-${AWS_REGION}.xml --region $MAIN_STACK_REGION
  aws s3 cp /tmp/artifacts/bzt.log s3://$S3_BUCKET/results/${TEST_ID}/bzt-${PREFIX}-
${UUID}-${AWS_REGION}.log --region $MAIN_STACK_REGION
  aws s3 cp /tmp/artifacts/$LOG_FILE s3://$S3_BUCKET/results/${TEST_ID}/${TEST_TYPE}-
${PREFIX}-${UUID}-${AWS_REGION}.log --region $MAIN_STACK_REGION
  aws s3 cp /tmp/artifacts/$OUT_FILE s3://$S3_BUCKET/results/${TEST_ID}/${TEST_TYPE}-
${PREFIX}-${UUID}-${AWS_REGION}.out --region $MAIN_STACK_REGION
  aws s3 cp /tmp/artifacts/$ERR_FILE s3://$S3_BUCKET/results/${TEST_ID}/${TEST_TYPE}-
${PREFIX}-${UUID}-${AWS_REGION}.err --region $MAIN_STACK_REGION
  aws s3 cp /tmp/artifacts/kpi.${KPI_EXT} s3://$S3_BUCKET/results/${TEST_ID}/kpi-
${PREFIX}-${UUID}-${AWS_REGION}.${KPI_EXT} --region $MAIN_STACK_REGION

else
  echo "An error occurred while the test was running."
fi
```

除了 Dockerfile 和 bash 脚本外，该目录中还包含两个 Python 脚本。每个任务都在 bash 脚本中运行一个 Python 脚本。工作任务运行`ecslistener.py`脚本，而领导任务将运行`ecscontroller.py`脚本。该`ecslistener.py`脚本在端口 50000 上创建一个套接字并等待消息。该`ecscontroller.py`脚本连接到套接字并将启动测试消息发送给工作器任务，这样它们就可以同时启动。

分布式负载测试 API

此负载测试解决方案可帮助您以安全的方式公开测试结果数据。该 API 充当访问存储在 Amazon DynamoDB 中的测试数据的 “前门”。您还可以使用 APIs 访问您在解决方案中内置的任何扩展功能。

该解决方案使用与 Amazon API Gateway 集成的 Amazon Cognito 用户池进行识别和授权。当用户池与 API 一起使用时，仅允许客户端在提供有效的身份令牌后调用用户池激活的方法。

有关直接通过 API 运行测试的更多信息，请参阅 Amazon API Gateway REST API 参考文档中的签署请求。

解决方案的 API 中提供了以下操作。

> **Note**
>
> 有关`testScenario`和其他参数的更多信息，请参阅 GitHub 存储库中的场景和负载示例。

堆栈信息

- 获取 /stack-info

场景

- 获取 /场景
- 帖子/场景
- 选项/场景
- 获取 /scenarios/ {testID}
- POST /scenarios/ {testID}
- 删除 /scenarios/ {testID}
- 选项 /scenarios/ {testID}

测试运行

- GET /scenarios/ {testID} /testruns
- GET /scenarios/ {testID} /testruns/ {} testRunId
- 删除 /scenarios/ {testID} /testruns/ {} testRunId

基准

- GET /scenarios/ {testID} /baseline
- PUT /scenarios/ {testID} /baseline
- 删除 /scenarios/ {testID} /baseline

任务

- 获取 /tasks
- 选项 /任务

区域

- 获取 /区域
- 选项 /区域

获取 /stack-info

描述

该GET /stack-info操作检索有关已部署堆栈的信息，包括创建时间、区域和版本。此端点由前端使用。

响应

200-成功

Name	描述
created_time	创建堆栈时的 ISO 8601 时间戳（例如，）2025-09-09T19:40:22Z
region	部署堆栈的 AWS 区域（例如，us-east-1 ）
version	已部署解决方案的版本（例如，v4.0.0）

错误响应

- 403-禁止：访问堆栈信息的权限不足
- 404-未找到：堆栈信息不可用
- 500-服务器内部错误

获取 /场景

描述

该GET /scenarios操作允许您检索测试场景列表。

响应

Name	描述
data	场景列表，包括每项测试的 ID、名称、描述、状态、运行时间、标签、总运行次数和上次运行时间

帖子/场景

描述

该POST /scenarios操作允许您创建或安排测试场景。

请求正文

Name	描述
testName	测试的名称
testDescription	测试的描述
testTaskConfigs	指定concurrency （并行运行次数）、taskCount （运行测试所需的任务数）和场景region的对象
testScenario	测试定义包括并发性、测试时间、主机和测试方法
testType	测试类型（例如simple、jmeter）
fileType	上传文件类型（例如、nonescript、zip）
tags	用于对测试进行分类的字符串数组。最大长度为 5 的可选字段（例如，["blue", "3.0", "critical"] ）

Name	描述
scheduleDate	运行测试的日期。仅在安排测试时提供（例如，2021-02-28 ）
scheduleTime	运行测试的时间。仅在安排测试时提供（例如，21:07）
scheduleStep	计划过程中的步骤。仅在安排定期测试时提供。（可用步骤包括create和start）
cronvalue	用于自定义定期调度的 cron 值。如果使用，请省略 ScheduleDate 和 ScheduleTime。
cronExpiryDate	必填日期，以便 cron 过期且不会无限期运行。
recurrence	预定测试的重复性。仅在安排重复测试（例如、dailyweeklybiweekly、或monthly）时提供

响应

Name	描述
testId	测试的唯一 ID
testName	测试的名称
status	测试的状态

选项/场景

描述

该OPTIONS /scenarios操作使用正确的 CORS 响应标头为请求提供响应。

响应

Name	描述
testId	测试的唯一 ID
testName	测试的名称
status	测试的状态

获取 /scenarios/ {testID}

描述

该GET /scenarios/{testId}操作允许您检索特定测试场景的详细信息。

请求参数

testId

- 测试的唯一 ID

  类型：字符串

  必需：是

latest

- 查询参数以仅返回最新的测试运行。默认为 true

  类型：布尔值

  必需：否

history

- 用于在响应中包含测试运行历史记录的查询参数。默认值为 true。设置为false可排除历史记录

  类型：布尔值

  必需：否

响应

Name	描述
testId	测试的唯一 ID
testName	测试的名称
testDescription	测试的描述
testType	正在运行的测试类型（例如，simple，jmeter）
fileType	上传的文件类型（例如、nonescript、zip）
tags	用于对测试进行分类的字符串数组
status	测试的状态
startTime	上次测试开始的时间和日期
endTime	上次考试结束的时间和日期
testScenario	测试定义包括并发性、测试时间、主机和测试方法
taskCount	运行测试所需的任务数
taskIds	运行测试 IDs 的任务列表
results	测试的最终结果
history	过去测试的最终结果清单（不包括何时history=false ）
totalRuns	此场景的测试运行总数
lastRun	上次测试运行的时间戳
errorReason	发生错误时生成的错误消息

Name	描述
nextRun	下一次计划运行（例如，2017-04-22 17:18:00）
scheduleRecurrence	测试的重复性（例如 、daily、weekly、biweekly、monthly）

POST /scenarios/ {testID}

描述

该POST /scenarios/{testId}操作允许您取消特定的测试场景。

请求参数

testId

- 测试的唯一 ID

  类型：字符串

  必需：是

响应

Name	描述
status	测试的状态

删除 /scenarios/ {testID}

描述

该DELETE /scenarios/{testId}操作允许您删除与特定测试场景相关的所有数据。

请求参数

`testId`

- 测试的唯一 ID

  类型：字符串

  必需：是

响应

Name	描述
`status`	测试的状态

选项 /scenarios/ {testID}

描述

该`OPTIONS /scenarios/{testId}`操作使用正确的 CORS 响应标头为请求提供响应。

响应

Name	描述
`testId`	测试的唯一 ID
`testName`	测试的名称
`testDescription`	测试的描述
`testType`	正在运行的测试类型（例如，`simple`，`jmeter`）
`fileType`	上传的文件类型（例如、`nonescript`、`zip`）
`status`	测试的状态

Name	描述
startTime	上次测试开始的时间和日期
endTime	上次考试结束的时间和日期
testScenario	测试定义包括并发性、测试时间、主机和测试方法
taskCount	运行测试所需的任务数
taskIds	运行测试 IDs 的任务列表
results	测试的最终结果
history	过去测试的最终结果清单
errorReason	发生错误时生成的错误消息

GET /scenarios/ {testID} /testruns

描述

该GET /scenarios/{testId}/testruns操作会检索特定测试场景 IDs 的测试运行，可以选择按时间范围进行筛选。W latest=true hen，仅返回最近的单个测试运行。

请求参数

testId

- 测试场景 ID

  类型：字符串

  必需：是

latest

- 仅返回最新的测试运行 ID

  类型：布尔值

  默认值：false

必需：否

start_timestamp

- 用于筛选测试运行的 ISO 8601 时间戳（包括在内）。例如，2024-01-01T00:00:00Z

  类型：字符串（日期时间格式）

  必需：否

end_timestamp

- ISO 8601 时间戳，用于筛选测试运行直到（含）。例如，2024-12-31T23:59:59Z

  类型：字符串（日期时间格式）

  必需：否

limit

- 要返回的最大测试运行次数（在以下情况下忽略latest=true）

  类型：整数（最小值：1，最大值：100）

  默认值：20

  必需：否

next_token

- 从上一个回复中获取下一页的分页令牌

  类型：字符串

  必需：否

响应

200-成功

Name	描述
testRuns	测试运行对象数组，每个对象包含testRunId （字符串）和startTime （ISO 8601 日期时间）

Name	描述
pagination	包含limit（整数）和next_token （字符串或空）的对象。如果没有更多结果，则令牌为空

错误响应

- 400-时间戳格式或参数无效
- 404-未找到测试场景
- 500-服务器内部错误

示例用法

- 仅限最新测试：GET /scenarios/test123/testruns?latest=true
- 时间范围内的最新：GET /scenarios/test123/testruns?latest=true&start_timestamp=2024-01-01T00:00:00Z
- 下一页请求：GET /scenarios/test123/testruns?limit=20&next_token=eyJ0ZXN0SWQiOiJzZVFVeTEyTEtMIiwic3RhcnRUaW1lIjoiMjAyNC0wMS0

GET /scenarios/ {testID} /testruns/ {} testRunId

描述

该GET /scenarios/{testId}/testruns/{testRunId}操作会检索特定测试运行的完整结果和指标。（可选）省略历史记录结果，history=false以便更快地做出响应。

请求参数

testId

- 测试场景 ID

  类型：字符串

  必需：是

testRunId

- 具体的测试运行 ID

类型：字符串

必需：是

history

- 在响应中包含历史数组。设置为可省略历史记录false以加快响应速度

类型：布尔值

默认值：true

必需：否

响应

200-成功

Name	描述
testId	测试的唯一 ID（例如，seQUy12LKL ）
testRunId	特定的测试运行 ID（例如，2DEwHItEne ）
testDescription	负载测试的描述
testType	测试的类型（例如，simple，jmeter）
status	测试运行的状态：complete、runningfailed、或cancelled
startTime	测试开始的时间和日期（例如，2025-09-09 21:01:00）
endTime	测试结束的时间和日期（例如，2025-09-09 21:18:29）
succPercent	成功百分比（例如，100.00）

Name	描述
testTaskConfigs	包含region、taskCount 和的任务配置对象数组 concurrency
completeTasks	将对象映射到已完成任务计数的区域
results	包含详细指标的对象，包括avg_lt（平均延迟）、百分位数（p0_0p50_0p90_0、p95_0、p99_0、p99_9、、、p均响应时间）、avg_ct（平均连接时间）、stdev_rt（标准差响应时间）、concurrency throughput 、succ（成功计数）、fail（失败计数）、bytes、testDuration metricS3Location 、rc（响应代码数组）和labels数组
testScenario	包含带有execution reporting 、和scenarios 属性的测试配置的对象
history	历史测试结果数组（不包括何时history=false ）

错误响应

- 400-测试编号无效或 testRunId
- 404-未找到测试运行
- 500-服务器内部错误

删除 /scenarios/ {testID} /testruns/ {} testRunId

描述

该DELETE /scenarios/{testId}/testruns/{testRunId}操作会删除与特定测试运行相关的所有数据和工件。测试运行数据已从 DynamoDB 中移除，而 S3 中的实际测试数据保持不变。

请求参数

`testId`

- 测试场景 ID

  类型：字符串

  必需：是

`testRunId`

- 要删除的特定测试运行 ID

  类型：字符串

  必需：是

响应

204-成功

测试运行已成功删除（未返回任何内容）

错误响应

- 400-测试编号无效或 testRunId
- 403-禁止：权限不足，无法删除测试运行
- 404-未找到测试运行
- 409-冲突：测试运行当前正在运行，无法删除
- 500-服务器内部错误

GET /scenarios/ {testID} /baseline

描述

该`GET /scenarios/{testId}/baseline`操作会检索场景的指定基准测试结果。根据`data`参数返回基准测试运行 ID 或完整基线结果。

请求参数

`testId`

- 测试场景 ID

  类型：字符串

  必需：是

`data`

- 如果返回完整的基线测试运行数据`true`，否则只有 testRunId

  类型：布尔值

  默认值：`false`

  必需：否

响应

200-成功

何时`data=false`（默认）：

Name	描述
`testId`	测试场景 ID（例如，`seQUy12LKL` ）
`baselineTestRunId`	基准测试运行 ID（例如，`2DEwHItEne` ）

什么时候`data=true`：

Name	描述
`testId`	测试场景 ID（例如，`seQUy12LKL` ）
`baselineTestRunId`	基准测试运行 ID（例如，`2DEwHItEne` ）

Name	描述
baselineData	完成测试运行结果对象（结构与GET /scenarios/{testId}/testruns/{testRunId} ）

错误响应

- 400-无效的 testID 参数
- 404-未找到测试场景或未设置基准
- 500-服务器内部错误

PUT /scenarios/ {testID} /baseline

描述

该PUT /scenarios/{testId}/baseline操作将特定的测试运行指定为性能比较的基准。每个场景只能设置一个基准。

请求参数

testId

- 测试场景 ID

  类型：字符串

  必需：是

请求正文

Name	描述
testRunId	要设置为基准的测试运行 ID（例如，2DEwHItEne ）

响应

200-成功

Name	描述
message	确认消息（例如，Baseline set successfully ）
testId	测试场景 ID（例如，seQUy12LKL ）
baselineTestRunId	设置的基准测试运行 ID（例如，2DEwHItEne ）

错误响应

- 400-测试编号无效或 testRunId
- 404-未找到测试场景或测试运行
- 409-冲突：无法将测试运行设置为基准（例如，测试失败）
- 500-服务器内部错误

删除 /scenarios/ {testID} /baseline

描述

该DELETE /scenarios/{testId}/baseline操作通过将场景的基线值设置为空字符串来清除该场景的基准值。

请求参数

testId

- 测试场景 ID

  类型：字符串

  必需：是

响应

204-成功

已成功清除基线（未返回任何内容）

错误响应

- 400-无效的测试 ID
- 500-服务器内部错误

获取 /tasks

描述

该GET /tasks操作允许您检索正在运行的亚马逊弹性容器服务 (Amazon ECS) 任务列表。

响应

Name	描述
tasks	运行测试 IDs 的任务列表

选项 /任务

描述

OPTIONS /tasks任务操作使用正确的 CORS 响应标头为请求提供响应。

响应

Name	描述
taskIds	运行测试 IDs 的任务列表

获取 /区域

描述

该GET /regions操作允许您检索在该区域运行测试所需的区域资源信息。

响应

Name	描述
testId	区域 ID
ecsCloudWatchLogGroup	该地区的 Amazon Fargate 任务的亚马逊 CloudWatch 日志组的名称
region	表中资源所在的区域
subnetA	该区域中一个子网的 ID
subnetB	该区域中一个子网的 ID
taskCluster	该地区的 AWS Fargate 集群的名称
taskDefinition	该区域中任务定义的 ARN
taskImage	区域中任务镜像的名称
taskSecurityGroup	该区域中安全组的 ID

选项 /区域

描述

该OPTIONS /regions操作使用正确的 CORS 响应标头为请求提供响应。

响应

Name	描述
testId	区域 ID

Name	描述
ecsCloudWatchLogGroup	该地区的 Amazon Fargate 任务的亚马逊 CloudWatch 日志组的名称
region	表中资源所在的区域
subnetA	该区域中一个子网的 ID
subnetB	该区域中一个子网的 ID
taskCluster	该地区的 AWS Fargate 集群的名称
taskDefinition	该区域中任务定义的 ARN
taskImage	区域中任务镜像的名称
taskSecurityGroup	该区域中安全组的 ID

增加容器资源

要增加负载测试可以模拟的并发虚拟用户数量（并发），您需要增加分配给每个 Amazon ECS 任务的 CPU 和内存资源。这包括创建具有更高资源限制的新任务定义修订版，然后更新解决方案的 DynamoDB 配置，以便在将来的测试运行中使用新的任务定义。

创建新的任务定义修订版

按照以下步骤创建具有增加 CPU 和内存资源的新任务定义：

1. 登录 Amazon 弹性容器服务控制台。
2. 在左侧导航菜单中，选择任务定义。
3. 选中与此解决方案对应的任务定义旁边的复选框。例如，[replaceable]<stackName>`-EcsTaskDefinition-*<system-generated-random-Hash>*。
4. 选择 Create new revision（创建新修订）。
5. 在 “创建新修订版本” 页面上，执行以下操作：
   a. 在 “任务大小” 下，将任务内存和任务 CPU 修改为所需的值。值越高，每个任务的并发虚拟用户就越多。
   b. 在 “容器定义” 下，查看硬/软内存限制。如果此限制低于所需的内存，请选择容器。

c. 在 “编辑容器” 对话框中，转到 “内存限制”，然后更新硬限制以匹配或小于您的任务内存分配。

d. 选择更新。

6. 在 “创建新修订版本” 页面上，选择 “创建”。
7. 成功创建任务定义后，记录完整的任务定义 ARN，包括版本号。例如：[replaceable]<stackName>`-EcsTaskDefinition-*<system-generated-random-Hash>*:[可替换] <system-generated-versionNumber>。

更新 DynamoDB 表

创建新的任务定义修订版后，必须更新解决方案的 DynamoDB 表，以便 future 的测试运行使用新的任务定义。对要使用更新后的任务定义的每个 AWS 区域重复以下步骤：

1. 导航到 DynamoDB 控制台。
2. 在左侧导航窗格中，选择 “表” 下的 “浏览项目”。
3. 选择与此解决方案关联的 scenarios-table DynamoDB 表。例如，[replaceable]<stackName>`-DLTTest RunnerStorage DLTScenarios 表-*<system-generated-random-Hash>*。
4. 选择与您在其中创建新任务定义修订版的区域相对应的项目。例如，region-[replaceable]<region-name>`。
5. 在项目编辑器中，找到 TaskD efinition 属性并使用您在上一节中记录的完整任务定义 ARN（包括版本号）更新其值。
6. 选择保存更改。

> **Note**
>
> 更新的任务定义将仅用于新的测试运行。当前正在运行或计划的所有测试都将继续使用先前的任务定义。

MCP 工具规格

分布式负载测试解决方案公开了一组 MCP 工具，这些工具使 AI 代理能够与测试场景和结果进行交互。这些工具提供了与 AI 代理处理信息的方式一致的高级抽象功能，使他们能够专注于分析和见解，而不是详细的 API 合同。

> **Note**
>
> 所有 MCP 工具都提供对解决方案数据的只读访问权限。不支持通过 MCP 接口修改测试场景或配置。

列出场景

描述

该`list_scenarios`工具检索包含基本元数据的所有可用测试场景的列表。

终端节点

```
GET /scenarios
```

Parameters

无

响应

Name	描述
`testId`	测试场景的唯一标识符
`testName`	测试场景的名称
`status`	测试场景的当前状态
`startTime`	测试的创建时间或上次运行的时间
`testDescription`	测试场景的描述

获取场景详情

描述

该`get_scenario_details`工具检索单个测试场景的测试配置和最近的测试运行。

终端节点

```
GET /scenarios/<test_id>?history=false&results=false
```

请求参数

`test_id`

- 测试场景的唯一标识符

  类型：字符串

  必需：是

响应

Name	描述
`testTaskConfigs`	每个区域的任务配置
`testScenario`	测试定义和参数
`status`	当前测试状态
`startTime`	测试开始时间戳
`endTime`	测试结束时间戳（如果已完成）

列出测试运行次数

描述

该`list_test_runs`工具检索特定测试场景的测试运行列表，按最新到最旧的顺序排列。最多返回 30 个结果。

终端节点

```
GET /scenarios/<testid>/testruns/?limit=<limit>
```

或

```
GET /scenarios/<testid>/testruns/?
limit=30&start_date=<start_date>&end_date=<end_date>
```

请求参数

`test_id`

- 测试场景的唯一标识符

  类型：字符串

  必需：是

`limit`

- 要返回的最大测试运行次数

  类型：整数

  默认值：20

  最大值：30

  必需：否

`start_date`

- 用于筛选从特定日期开始的运行的 ISO 8601 时间戳

  类型：字符串（日期时间格式）

  必需：否

`end_date`

- 用于筛选特定日期之前运行的 ISO 8601 时间戳

  类型：字符串（日期时间格式）

  必需：否

响应

Name	描述
`testRuns`	包含每次运行的性能指标和百分位数的测试运行摘要数组

get_test_run

描述

该`get_test_run`工具检索单次测试的详细结果，包括区域和端点细分。

终端节点

```
GET /scenarios/<testid>/testruns/<testrunid>
```

请求参数

`test_id`

- 测试场景的唯一标识符

  类型：字符串

  必需：是

`test_run_id`

- 特定测试运行的唯一标识符

  类型：字符串

  必需：是

响应

Name	描述
`results`	完整的测试运行数据，包括区域结果明细、端点特定指标、性能百分位数（p50、p90、p95、

Name	描述
	p99）、成功和失败计数、响应时间和延迟，以及用于运行的测试配置

get_latest_test_run

描述

该get_latest_test_run工具检索特定测试场景的最新测试运行。

终端节点

```
GET /scenarios/<testid>/testruns/?limit=1
```

> **Note**
>
> 使用全局二级索引 (GSI) 按时间对结果进行排序，确保返回最新的测试运行。

请求参数

`test_id`

- 测试场景的唯一标识符

  类型：字符串

  必需：是

响应

Name	描述
`results`	最新测试运行数据，其格式与相同 `get_test_run`

get_baseline_test_run

描述

该get_baseline_test_run工具检索特定测试场景的基准测试运行。该基准用于性能比较目的。

终端节点

```
GET /scenarios/<test_id>/baseline
```

请求参数

test_id

- 测试场景的唯一标识符

  类型：字符串

  必需：是

响应

Name	描述
baselineData	用于比较目的的基准测试运行数据，包括指定基准运行的所有指标和配置

获取_测试_运行_工件

描述

该get_test_run_artifacts工具检索 Amazon S3 存储桶信息，用于访问测试项目，包括日志、错误文件和结果。

终端节点

```
GET /scenarios/<testid>/testruns/<testrunid>
```

请求参数

`test_id`

- 测试场景的唯一标识符

  类型：字符串

  必需：是

`test_run_id`

- 特定测试运行的唯一标识符

  类型：字符串

  必需：是

响应

Name	描述
`bucketName`	存储工件的 S3 存储桶名称
`testRunPath`	当前工件存储的路径前缀（版本 4.0+）
`testScenarioPath`	旧版工件存储的路径前缀（4.0 版之前）

> **Note**
>
> 所有 MCP 工具都利用现有的 API 端点。无需对底层 APIs 进行修改即可支持 MCP 功能。

参考

本节包括有关数据收集的信息、指向相关资源的指针以及为该解决方案做出贡献的构建者列表。

数据收集

此解决方案向 AWS 发送有关该解决方案使用情况的运营指标（“数据”）。我们使用这些数据来更好地了解客户如何使用此解决方案以及相关服务和产品。AWS 对这些数据的收集受 AWS 隐私声明的约束。

贡献者

- 汤姆·南丁格尔
- 费尔南多丁格勒
- 李范锡
- George Lenz
- 艾琳 McGill
- 迪米特里·洛佩兹
- Kamyar Ziabari
- Bassem Wanis
- 加维特·辛格
- Nikhil Reddy
- 西蒙·克罗尔
- Ahern Knox
- 伊恩·唐纳德
- 欧文·布雷迪
- 吉姆·塔里奥
- Thyag Ramachandran
- 杨琴
- 詹姆斯·王

术语表

本词汇表定义了《AWS 分布式负载测试实施指南》中使用的首字母缩略词和缩写词。

技术协议和格式

AGPL

Affero 通用公共许可证。K6 使用的开源软件许可证。

API

应用程序编程接口。一组协议和工具，用于构建软件应用程序和实现不同系统之间的通信。

CLI

命令行界面。一种基于文本的界面，用于与软件和操作系统进行交互。

CORS

跨源资源共享。一项安全功能，允许或限制在一个源上运行的 Web 应用程序访问来自其他来源的资源。

CSV

以逗号分隔的值。一种用于以纯文本形式存储表格数据的文件格式，通常用于数据导出。

gRPC

gRPC 远程过程调用。用于远程过程调用的高性能、开源框架。

HTTP

超文本传输协议。用于在万维网上传输数据的基础协议。

HTTPS

HTTP 安全。HTTP 的扩展，它使用加密技术通过网络进行安全通信。

JSON

JavaScript 物体标记。一种轻量级的数据交换格式，便于人类读写，机器也易于解析和生成。

JWT

JSON 网络令牌。一种紧凑、网址安全的方式，用于表示要在双方之间转移以进行身份验证和授权的声明。

OAuth

打开授权。访问委派的开放标准，通常用于基于令牌的身份验证和授权。

REST

代表性状态转移。一种架构风格，用于使用无状态通信和标准 HTTP 方法设计联网应用程序。

SSE

服务器发送的事件。一种服务器推送技术，使客户端能够通过 HTTP 连接接收来自服务器的自动更新。

UI

用户界面。用户与软件应用程序交互时使用的视觉元素和控件。

URL

统一资源定位器。用于访问互联网资源的地址。

XML

可扩展标记语言。一种标记语言，它定义了以人类可读和机器可读的格式对文档进行编码的规则。

测试和数据库术语

FTP

文件传输协议。一种用于在客户端和服务器之间传输文件的标准网络协议。

GSI

全球二级索引。一项 DynamoDB 功能，允许您使用备用密钥查询数据。

JDBC

Java 数据库连接。一种 Java API，用于连接和执行与数据库的查询。

JMS

Java 消息服务。用于在两个或多个客户端之间发送消息的 Java API。

TPS

每秒交易次数。衡量系统在一秒钟内可以处理的交易数量的指标。

AWS 和系统条款

进行筛选

Amazon 资源名称。AWS 资源的唯一标识符，用于指定 AWS 服务中的资源。

ISO

国际标准化组织。一个独立的非政府组织，负责制定国际标准。本指南中引用了 ISO 8601 时间戳格式。

SLA

服务等级协议。服务提供商与客户之间的承诺，用于定义预期的服务级别。

UUID

通用唯一标识符。一个 128 位的数字，用于唯一标识计算机系统中的信息。

vCPU

虚拟中央处理器。分配给虚拟机或容器的虚拟处理器，代表物理 CPU 处理能力的一部分。

负载测试条款

并发

每个任务的并发虚拟用户数。此参数控制每个 Fargate 任务在负载测试期间生成多少模拟用户。

区域堆栈

部署在 AWS 区域中的 CloudFormation 堆栈，用于为多区域负载测试提供测试基础设施。

任务计数

为运行测试场景而启动的 Fargate 容器（任务）的数量。生成的总负载等于任务数乘以并发数。

测试场景

配置的负载测试，包括测试类型、目标端点、任务计数、并发度、持续时间和其他参数。

修订

访问我们 GitHub 存储库中的 Changelog.md，跟踪特定版本的改进和修复。

版权声明

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。