



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

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon 的商标和商业外观不得用于任何非 Amazon 的商品或服务，也不得以任何可能引起客户混淆、贬低或诋毁 Amazon 的方式使用。所有非 Amazon 拥有的其他商标均为各自所有者的财产，这些所有者可能附属于 Amazon、与 Amazon 有关联或由 Amazon 赞助，也可能不是如此。

Table of Contents

解决方案概述	1
功能	2
优势	3
使用案例	4
概念和定义	5
架构概述	7
架构图	7
AWS Well-Architected 设计注意事项	9
卓越运营	9
安全性	9
可靠性	10
性能效率	10
成本优化	11
可持续性	11
架构详情	12
前端	12
负载测试 API	12
Web 控制台	12
MCP 服务器 (可选)	13
Backend	13
容器镜像管道	13
测试框架配置	14
测试基础架构	14
负载测试引擎	14
MCP 服务器	15
Amazon 基岩网关 AgentCore	15
DLT MCP 服务器 Lambda	15
身份验证集成	15
此解决方案中的 AWS 服务	16
AWS 上的分布式负载测试的工作原理	17
MCP 服务器工作流程 (可选)	19
设计注意事项	21
受支持的应用程序	21
测试类型	21

安排测试	23
并行测试	24
User management	24
身份提供者联合	24
区域部署	27
规划您的部署	28
成本	28
MCP 服务器额外费用 (可选)	29
ALB + ECS Fargate 托管 Web 控制台额外费用 (可选)	30
安全性	31
IAM 角色	31
亚马逊 CloudFront	31
Amazon API Gateway	31
AWS Fargate 安全组	32
Amazon VPC	32
网络压力测试	34
限制对公共用户界面的访问	34
MCP 服务器安全 (可选)	34
ALB + ECS Fargate 托管的 Web 控制台安全 (可选)	34
无头 (自带网络服务器) 安全 (可选)	35
Third-party 测试框架	35
支持的 AWS 区域	36
支持 MCP 服务器的 AWS 区域 (可选)	37
配额	38
此解决方案中的 AWS 服务的配额	39
AWS CloudFormation 配额	39
负载测试配额	39
并行测试	39
亚马逊 EC2 测试政策	39
亚马逊 CloudFront 负载测试政策	40
在部署后监控解决方案	40
设置 CloudWatch 警报	40
聘请专家	40
AWS 倒计时高级短期项目 , 用于在 AWS 上进行分布式负载测试	40
部署解决方案	43
部署流程概述	44

使用 AWS Launch Wizard 进行部署	44
使用 AWS 进行部署 CloudFormation	45
AWS CloudFormation 模板	45
启动堆栈 (默认 , CloudFront + S3 托管 Web 控制台)	45
启动堆栈 (ALB + ECS Fargate 托管的 Web 控制台)	48
启动堆栈 (Headless)	53
Multi-Region 部署	55
使用解决方案	58
创建测试场景	58
步骤 1 : 常规设置	58
步骤 2 : 场景配置	59
第 3 步 : 流量形状	61
步骤 4 : 审核并创建	63
运行测试场景	63
场景详细信息视图	63
测试执行工作流程	64
测试运行状态	64
使用实时数据进行监控	64
取消考试	66
浏览测试结果	66
测试运行摘要指标	66
测试运行表	67
基线比较	67
详细的测试结果	67
“错误”选项卡	69
“对象”选项卡	69
S3 结果结构	69
使用 CloudWatch 日志见解进行监控	70
使用已保存的查询	70
DLT-测试时间表	70
DLT-测试错误	71
DLT-任务失败	71
DLT-孤儿清理	72
CloudWatch 警报	72
订阅警报通知	72
OrphanCleanupFailure	73

MetricFilterCount	74
MCP 服务器集成	74
第 1 步：获取 MCP 端点和访问令牌	74
第 2 步：使用 MCP Inspector 进行测试	75
步骤 3：配置 AI 开发客户端	77
提示示例	80
问题排查	83
已知问题解决方案	83
ALB + ECS Fargate 部署问题	85
无头（自带 Web 服务器）部署问题	86
联系 AWS Support	86
创建工单	86
我们可提供哪些帮助？	86
附加信息	87
帮助我们更快地处理您的工单	87
立即解决或联系我们	87
更新此解决方案	88
使用 AWS Launch Wizard 进行更新	88
使用 AWS 进行更新 CloudFormation	88
对 v3.3.0 之前版本的更新进行故障排除	89
更新区域堆栈	90
AWS Systems Manager 应用程序管理器	90
卸载此解决方案	91
使用 AWS 管理控制台	91
AWS CloudFormation	91
AWS Launch Wizard	91
使用 AWS 命令行界面	91
手动删除保留的资源	92
删除 Amazon S3 存储桶	92
删除亚马逊 DynamoDB 表	92
删除 CloudWatch 日志组	93
删除 CloudWatch 仪表板	93
清理 ALB + ECS Fargate 部署后资源	94
堆栈删除后恢复 DLT 数据	94
背景	94
重要限制	95

恢复过程	95
避免堆栈漂移	100
无法恢复的东西	100
预防措施	101
开发人员指南	102
源代码	102
Maintenance	102
版本	102
容器镜像自定义	103
负载测试仪图片	103
Web 控制台镜像 (仅限 ALB + ECS Fargate 模板)	110
分布式负载测试 API	114
获取 /stack-info	115
获取 /场景	116
帖子/场景	116
选项/场景	118
获取 /scenarios/ {testID}	118
POST /scenarios/ {testID}	120
删除 /scenarios/ {testID}	121
选项 /scenarios/ {testID}	121
GET /scenarios/ {testID} /testruns	122
GET /scenarios/ {testID} /testruns/ {test} RunId	125
删除 /scenarios/ {testID} /testruns/ {test} RunId	127
GET /scenarios/ {testID} /baseline	128
PUT /scenarios/ {testID} /baseline	129
删除 /scenarios/ {testID} /baseline	130
获取 /tasks	131
选项 /任务	131
获取 /区域	132
选项 /区域	133
增加容器资源	133
创建新的任务定义修订版	133
更新 DynamoDB 表	134
MCP 工具规格	135
列出场景	135
获取场景详情	136

列出测试运行次数	137
get_test_run	138
get_latest_test_run	139
get_baseline_test_run	140
获取_测试_运行_工件	141
DLT CLI	142
关键功能	142
安装和详细参考	143
快速入门	143
Cron 表达式引用	144
接受的值	144
不支持的模式	145
预设图案	146
调度限制	146
参考	147
数据收集	147
贡献者	147
术语表	149
技术协议和格式	149
测试和数据库术语	150
AWS 和系统条款	151
负载测试条款	151
修订	152
版权声明	153
.....	cliv

自动对您的软件应用程序进行大规模测试

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 [JMeter](#)、[K6](#)、[Locust](#) 测试脚本或简单的 HTTP 端点配置自定义应用程序测试。有关捆绑框架的安全注意事项，请参阅 [Third-party 测试框架](#)。
- 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 [AWS 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。	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。	
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 AI-assisted 负载测试分析的可选 MCP 服务器。	MCP 服务器集成
了解可用的网络控制台托管选项 (CloudFront + S3、ALB + ECS Fargate 或无头)。	部署解决方案
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，(可选) 并使用 AWS Cloud Development Kit (AWS CDK) 部署解决方案。	GitHub repository

功能

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](#)。有关捆绑框架的安全注意事项，请参阅[Third-party 测试框架](#)。

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。

Multi-Region 负荷分布

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。

Real-Time 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。

灵活的 Web 控制台托管

有三个 Web 控制台托管选项可供选择：Amazon CloudFront + S3 (默认)、带有自定义域的 ALB + ECS Fargate 或无头 (自带网络服务器)。ALB + ECS Fargate 选项支持具有 VPC 阻止公共访问策略或受监管行业常见的零公共互联网暴露要求的环境。所有选项的后端和身份验证保持不变。有关更多信息，请参阅[部署解决方案](#)。

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-Assisted 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。

Real-World 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。

Cost-Efficient 基础设施

使用按使用付费定价的无服务器 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

Pre-Production 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 API、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。

Third-Party 服务测试

测试您的应用程序在各种负载条件下所依赖的第三方 API 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 JMeter 的测试，更高的并发度会增加 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 [amazon 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。

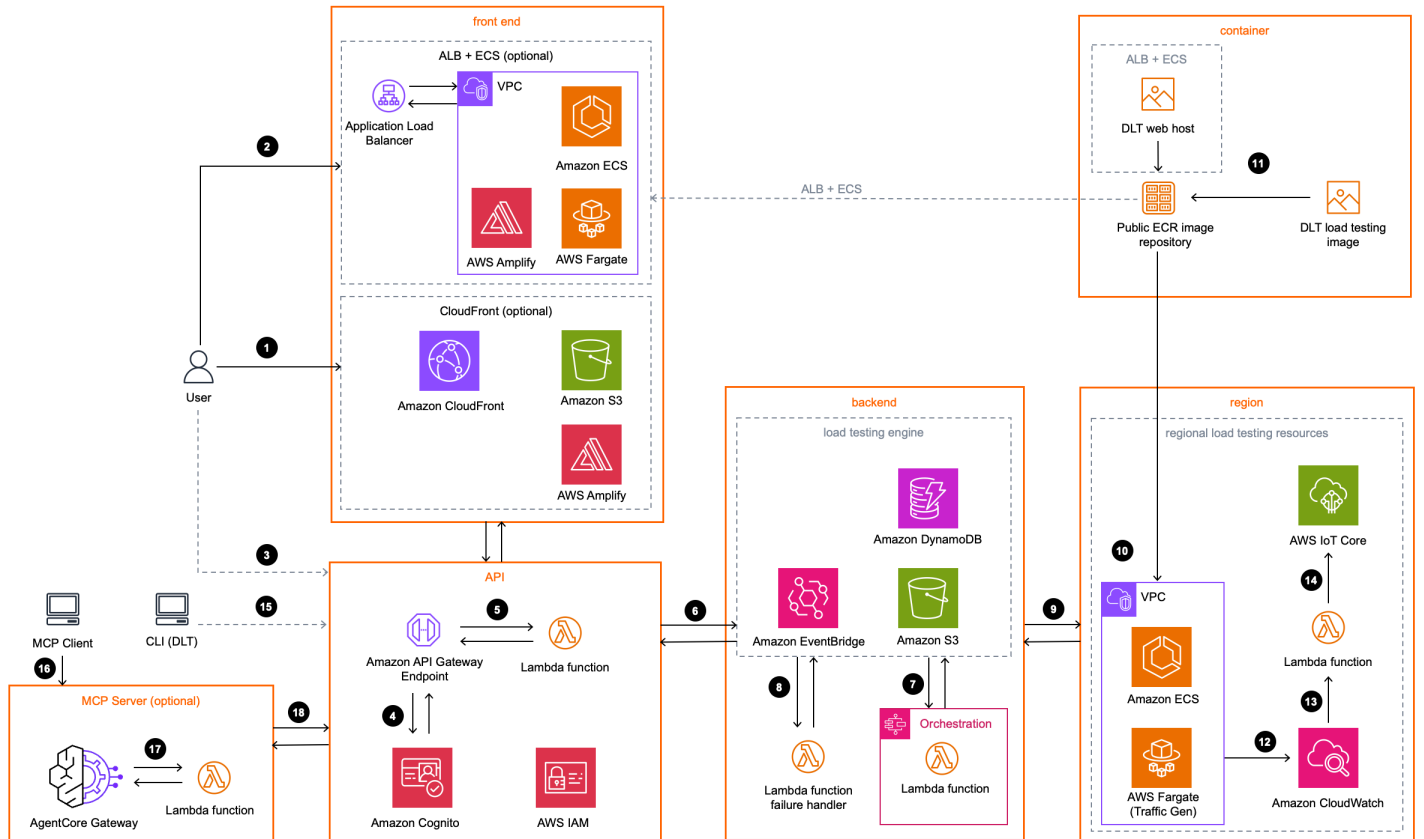
有关 AWS 术语的通用参考，请参阅 [AWS 词汇表](#)。

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 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. (CloudFront + S3 托管部署选项) 控制台用户通过[亚马逊访问 Web 控制台 CloudFront](#)，[亚马逊为托管在亚马逊简单存储服务 \(Amazon S3\) 存储桶中的 AWS Amplify 应用程序提供服务](#)。

2. (ALB + ECS Fargate 托管部署选项) 控制台用户通过[应用程序负载均衡器访问 Web 控制台](#)，该[负载均衡器](#)将流量路由到亚马逊[虚拟私有云](#) (亚马逊 VPC) 内在 AWS Fargate [上的亚马逊弹性容器服务](#) (亚马逊 EC S) 上运行的 AWS Amplify 应用程序。
3. (Headless 部署选项) 未部署任何公共前端。该解决方案将 Web 控制台作为可下载的 ZIP 文件提供到私有 Amazon S3 存储桶中。控制台用户可以从自托管 Web 服务器访问控制台。
4. 在初始配置期间，该解决方案会在 [Amazon Cognito](#) 用户池中创建默认管理员用户，并向您提供的电子邮件地址发送一封账户创建电子邮件。Cognito 用户池管理用户对 Web 控制台、REST API、CLI 和 MCP 服务器的访问权限。
5. [Amazon API Gateway](#) 调用 [AWS Lambda](#) 微服务，这些微服务提供管理测试数据和运行测试的业务逻辑。
6. 微服务与 Amazon S3、Amazon DynamoDB EventBridge 和亚马逊交互，以存储测试场景详细信息并管理测试计划。当您将测试安排在将来的某个时间或重复运行的时间间隔运行时，微服务会创建一个 EventBridge 调度程序计划，在计划的时间调用微服务。
7. 要运行测试，微服务会调用 [AWS Step Functions](#)，后者负责协调测试的执行。
8. EventBridge 规则将 Amazon ECS 任务和 Step Functions 失败事件路由到故障处理程序 Lambda 函数。
9. Step Functions 在你选择的每个 AWS 区域的 [AWS Fargate](#) 上启动[亚马逊弹性容器服务](#) (Amazon ECS) 任务。
10. 每项任务都在所选区域的[亚马逊虚拟私有云](#) (Amazon VPC) 中运行。
11. 负载测试容器使用安装了 [Taurus 测试自动化框架](#)的 [Amazon Linux 2023](#) 基础映像。Taurus 会执行你的 jMeter、K6、Locust 或单个 HTTP 端点测试。有关如何配置每个测试框架的详细信息，请参阅[测试框架配置](#)。ALB + ECS 选项将使用虚拟主机容器。容器镜像由 AWS 托管在[亚马逊弹性容器注册表](#) (Amazon ECR) 公共存储库中。
12. [每个 Fargate 任务都会将其每个区域的测试结果写入 Amazon S3，并向亚马逊发送日志。](#)
[CloudWatch](#)当所有区域都完成后，微服务将结果汇总到 DynamoDB 中。
13. 如果您启用实时数据选项，Lambda 函数将在测试期间接收来自 Amazon CloudWatch gate 任务的日志。
14. Lambda 函数将日志发布到部署主堆栈的区域 [AWS IoT Core](#) 中的一个主题。Web 控制台订阅主题以在测试运行时显示实时指标。
15. (可选 CLI 访问权限) 用户可以在本地安装 DLT 命令行界面 (CLI)，以便从其终端与解决方案进行交互。CLI 通过 Cognito 进行身份验证并直接调用 REST API，从而实现脚本化自动化和集成。CI/CD

Note

以下步骤描述了用于 AI-assisted 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。

16. MCP 客户端 (AI 开发工具) 连接到 [Amazon Bedrock 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。

17. 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 Gateway AgentCore y，网关将其发送回 MCP 客户端进行 AI-assisted 分析和见解。

18. 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 Framework Well-Architected k](#) 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 框架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 Access Management \(IAM\) 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

该解决方案创建的所有日志组都将日志保留 10 年，但 ALB + ECS Fargate 模板中的 Web 控制台和 WAF 日志组除外，它们保留了 1 年：

日志组	保留
AWS Step Functions、Lambda 运行时、ECS 测试任务、Cognito 用户池和 API Gateway 访问日志	10 岁
Web 控制台和 WAF (仅限 ALB + ECS Fargate 模板)	1 年
ECS 容器见解 (由 ECS 自动创建)	永不过期 (默认)

您可以在 Amazon 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。
- AgentCore Gateway 可作为分布式负载测试 API 的经济实惠 Lambda-based 代理，无需专用基础架构，并通过无服务器按请求付费定价降低成本。

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-assisted 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 API 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 API（亚马逊 API Gateway 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的 API。

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。

该解决方案支持三个 Web 控制台托管选项。所有选项的后端架构和 Cognito 身份验证都相同：

- CloudFront + S3 (默认) - 控制台托管在 Amazon S3 中，可通过亚马逊进行访问 CloudFront。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。
- ALB + ECS Fargate — 控制台在应用程序负载均衡器后面的 ECS Fargate 服务上运行，该服务具有客户提供的 ACM 证书和自定义域。AWS WAF Web ACL 部署在 ALB 前面，用于过滤常见的基于 Web 的攻击。此选项适用于 VPC 阻止公共访问 (BPA) 策略阻止来自公共 CloudFront 分配的流量的环境，或者对公共 Internet 暴露要求为零的组织。有关部署说明，请参阅[使用 ALB + ECS Fargate 进行部署](#)。
- He@@ adless (自带 Web 服务器) — 该解决方案仅部署后端，并提供 Web 控制台静态资产的可下载 zip 存档。您可以将控制台托管在自己的 Web 服务器上。此选项适用于需要完全网络隔离且没有面向公众的 AWS 终端节点的组织，或者需要在现有本地或企业基础设施上托管控制台的组织。有关部署说明，请参阅[使用无头模板进行部署 \(自带 Web 服务器\)](#)。

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 MCP-compatible 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 User-Defined 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，支持 JMeter、K6、Locust 和其他测试工具。AWS 将此图像

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。

测试框架配置

三个支持的测试框架是在解决方案生命周期的不同阶段配置的，以平衡图像大小和版本灵活性：

- Apache JMeter — 在堆栈部署期间暂存到您账户的 S3 存储桶中，并在运行 JMeter 或单个 HTTP 端点测试时在测试运行时提取。
- Grafana K6 — 直接从框架提供程序下载并在测试运行时提取，仅在 K6 测试运行时才提取。
- Locust — 在构建时安装到容器镜像中，在 Locust 测试运行之前一直处于空闲状态。

Note

K6 需要在测试运行时出站网络访问 Grafana K6 版本。限制出口将导致 K6 测试在下载步骤失败。

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[Multi-Region 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 ecr-checker AWS Lambda 函数、任务状态检查器 AWS Lambda 函数、任务运

行器 AWS Lambda 函数、任务取消器 AWS Lambda 函数和结果解析器 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启动实时数据发布者 Lambda 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：

Amazon 基岩网关 AgentCore

Amazon Bedrock 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，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 数据转换为代理可以轻松解释和分析的结构化 AI-friendly 格式。

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：

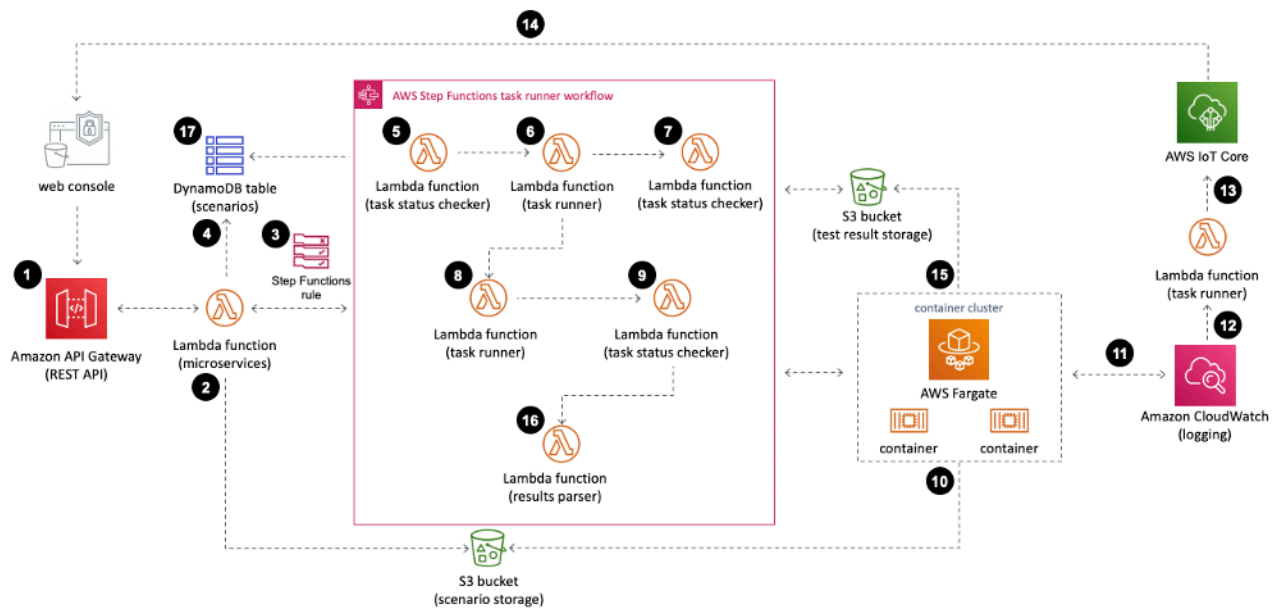
AWS 服务	说明
Amazon API Gateway	核心。在解决方案中托管 REST API 端点。
AWS CloudFormation	核心。管理解决方案基础架构的部署。
Amazon CloudFront	核心（仅限默认部署）。使用默认 CloudFront + S3 托管选项时，提供在 Amazon S3 中托管的网页内容。
Elastic Load Balancing（应用程序负载均衡器）	核心（仅限 ALB + ECS Fargate 部署）。使用 ALB + ECS Fargate 部署选项时为 Web 控制台提供服务。
AWS WAF	支持，可选（仅限 ALB + ECS Fargate 部署）。为 ALB 和 ECS 托管的 Web 控制台提供 Web 应用程序防火墙保护。
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 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 API 实现、测试结果解析和启动 workers/leader 任务提供逻辑。

AWS 服务	说明
AWS Step Functions	核心。为指定区域的 AWS Fargate 任务编排 Amazon ECS 容器的配置
AWS Amplify	支持。提供由 AWS Amplify 提供支持的部署控制台。
Amazon EventBridge	支持。将 Amazon ECS 任务和 Step Functions 失败事件路由到故障处理程序 Lambda 函数，并使用 EventBridge 计划程序安排测试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，并可视化展示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。

测试工作流程



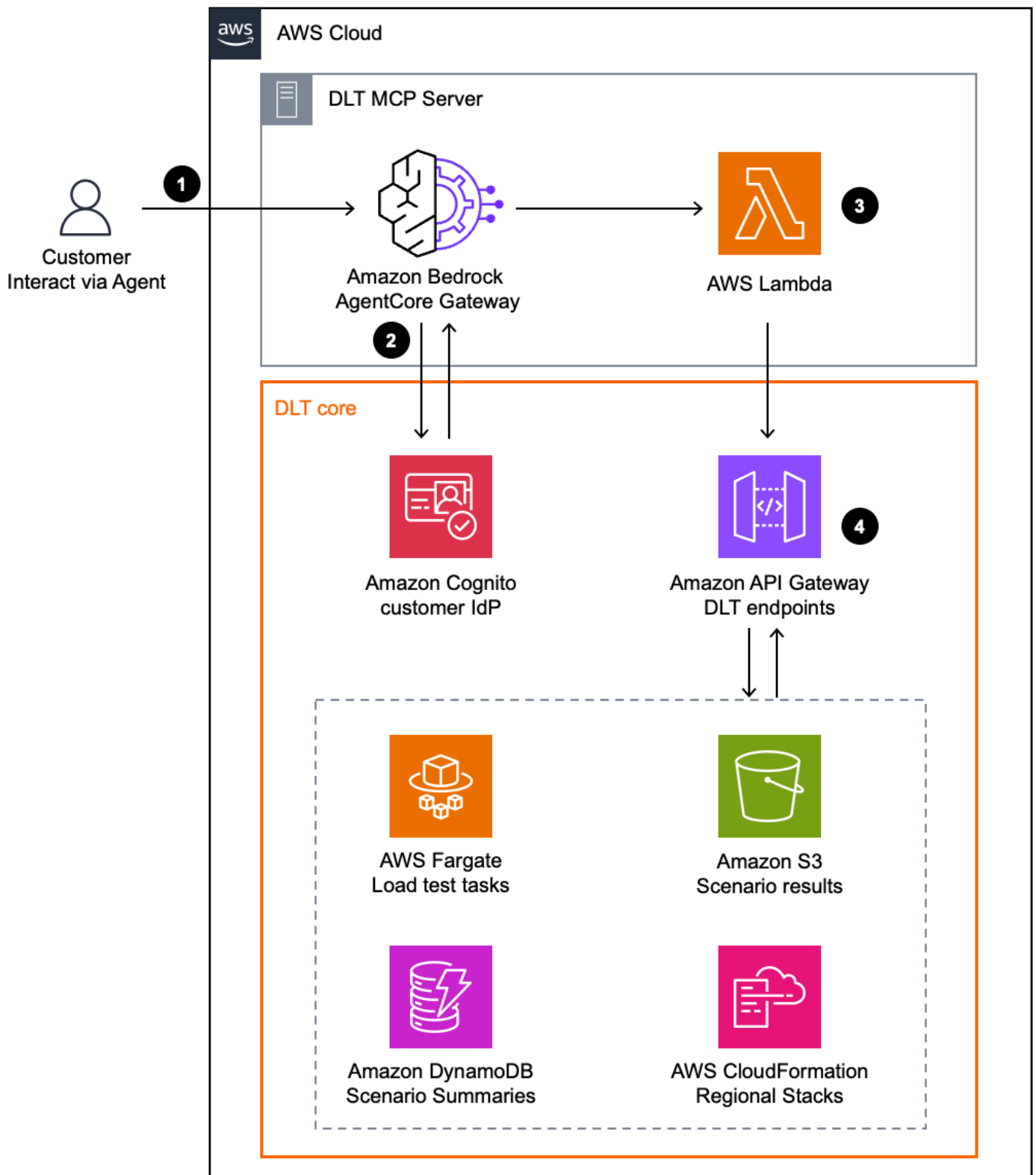
1. 您可以使用 Web 控制台向解决方案的 API 提交包含配置详细信息的测试场景。
2. 测试场景配置以 JSON 文件 `s3://<bucket-name>/test-scenarios/<$TEST_ID>/<$TEST_ID>.json` 的形式上传到亚马逊简单存储服务 (Amazon S3)。
3. AWS Step Functions 状态机使用测试 ID、任务计数、测试类型和文件类型作为 AWS Step Functions 状态机输入来运行。如果已安排测试，它将首先创建一个 Amazon 计划表，该 EventBridge 计划将在指定日期触发 AWS Step Functions。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，任务状态检查器 AWS 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. 任务状态检查器 AWS Lambda 函数检查所有 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 ID 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. 任务状态检查器 AWS 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 CloudWatch 中，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 函数对收到的数据进行结构化并将其发布到 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://dlite 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 服务器工作流程



1. 客户互动-客户通过 AI 代理与分布式负载测试解决方案进行交互。代理连接到 MCP 端点以请求访问负载测试数据。
2. 授权- AgentCore Gateway 会验证用户的 Amazon Cognito 身份验证令牌，以确保用户有权访问 DLT MCP 服务器。授权用户通过可用的代理工具获得对负载测试数据的只读访问权限。
3. 工具调用- AgentCore Gateway 将授权的 MCP 工具请求转发给 DLT MCP 工具 Lambda 函数。Lambda 函数实现了 AI 代理用来检索负载测试信息的工具。
4. Read-only API 访问 ——DLT MCP Tools Lambda 函数调用现有 DLT API Gateway 终端节点，从 DynamoDB、S3 和区域堆栈中检索测试数据。CloudFormation Lambda 函数提供了四个主要操作：
 - 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) 提供对测试数据的安全只读访问权限，以进行分析和见解。AI-powered

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。该解决方案支持使用 HTTP 或 HTTPS 协议的 API。

测试类型

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

Note

该解决方案无需修改即可将 JMeter、K6 和 Locust 作为第三方组件进行分发。有关安全注意事项、补丁选项和许可证信息，请参阅[Third-party 测试框架](#)。

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 标头和请求正文来测试 API。

在配置 HTTP 端点时，该解决方案会将您的配置转换为测试计划，该计划由捆绑的 Apache JMeter 二进制文件通过 Taurus 框架执行。简单的 HTTP 端点测试不接受测试档案，因此它们无法覆盖捆绑的 JMeter 二进制文件或插件。如果您需要使用已修补的 JMeter 运行 HTTP 端点测试，请改用 [jMeter 测试类型](#)。有关安全方面的考虑，请参阅 [Apache JMeter](#)。

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。

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，并可用于负载测试。

Note

如果您在 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 JMeter 输入文件和脚本文件位于 /home/user 目录中，而您引用 JMeter 脚本文件中的输入文件时，输入文件的路径必须为 /INPUT_FILES。如果改用 /home/user/INPUT_FILES，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。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。[当您创建新的 K6 测试时，Web 控制台会显示许可确认消息；有关许可和安全详细信息，请参阅 Grafana K6。](#)

Important

尽管您的 K6 脚本可以定义并发（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。

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
- 运行一次-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盘。

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功能。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](#)》。

身份提供者联合

该解决方案的 Amazon Cognito 用户池支持使用 SAML 2.0 或 OpenID Connect (OIDCIdPs) 协议与外部身份提供商 () 进行联合。联合允许用户使用其现有的公司或组织凭据而不是凭据登录 Web 控制台。Cognito-native 联邦用户获得的访问权限与直接在 Cognito 用户池中创建的用户相同。

该解决方案已经部署了 Cognito 用户池、域、应用程序客户端和托管用户界面。要启用联合，您只需要注册您的身份提供者并在现有应用程序客户端上将其启用即可。

如果您部署了可选的 MCP 服务器集成，则联合用户也可以使用相同的 Cognito 用户池凭据访问 MCP 服务器。

先决条件

在配置联合之前，您需要满足以下条件：

- 支持 SAML 2.0 或 OIDC 的外部身份提供商
- 管理员访问权限以配置外部 IdP（用于设置重定向 URI 或 ACS 网址）
- 该解决方案的 Cognito 用户池 ID（可在 CloudFormation 堆栈资源或 Amazon Cognito 控制台中找到）
- 该解决方案的 Cognito 域名前缀（可在 CloudFormation 堆栈输出中找到，也可在应用程序集成 > 域下的 Cognito 控制台中找到）

步骤 1：配置您的身份提供商

使用以下值配置外部身份提供商，使其可以与解决方案的 Cognito 用户池进行通信。

对于 SAML 身份提供商：

- SP 实体 ID：`urn:amazon:cognito:sp:_{UserPoolId}_`
- ACS 网址：`\https://<cognito-domain>.auth.<region>.amazoncognito.com/saml2/idpresponse`

对于 OIDC 身份提供商：

- 重定向 URI：`\https://<cognito-domain>.auth.<region>.amazoncognito.com/oauth2/idpresponse`

有关您的 IdP 需求的详细信息，请参阅 Amazon Cognito 开发[者指南中的将 SAML 身份提供商添加到用户池或将 OIDC 身份提供商添加到用户池](#)。

第 2 步：在 Cognito 中注册身份提供商

使用 Amazon Cognito 控制台将您的外部身份提供商添加到解决方案的现有 Cognito 用户池中。

有关分步说明，请参阅 Amazon Cognito [开发者指南中的通过第三方添加用户池登录](#)信息。

步骤 3：配置属性映射

在身份提供者的声明和 Cognito 用户池属性之间配置属性映射。至少，将来自外部提供商的用户电子邮件声明映射到 Cognito 属性 email。也可以考虑映射，name 或者您的身份提供商 nickname 是否提供映射。

有关说明，请参阅 Amazon Cognito 开发者指南中的 [为您的用户池指定身份提供商属性映射](#)。

步骤 4：在应用程序客户端上启用身份提供商

在 Amazon Cognito 控制台中，找到该解决方案创建的应用程序客户端，然后在托管 UI 设置下启用您的新身份提供商。

有关说明，请参阅 [Amazon Cognito 开发者指南中的配置用户池应用程序客户端](#)。

Note

该解决方案已经配置了应用程序客户端的回调和退出网址、OAuth 范围和托管界面域。您无需修改这些设置，只需在现有应用程序客户端上启用您的身份提供商即可。

Important

该解决方案故意从 CloudFormation 应用程序客户端配置中省略该 SupportedIdentityProviders 属性。这允许您在部署后添加身份提供商，而无需触发 CloudFormation 偏差检测。如果在模板中设置了此属性，则通过控制台或 CLI 进行的任何手动 IdP 更改都将在下一次堆栈更新时被覆盖，从而使应用程序客户端仅恢复到模板中列出的提供程序。

由于省略了此属性，因此 CloudFormation 不会跟踪或管理应用程序客户端上启用了哪些身份提供商。配置联合后，您负责管理应用程序客户端 SupportedIdentityProviders 上的内容。要监控未经授权的更改，请启用 [AWS CloudTrail](#) 日志并创建 A [amazon EventBridge 规则](#) 以提醒 CreateIdentityProvider 和针对解决方案的 Cognito 用户池的 UpdateUserPoolClient API 调用。

Note

- 添加外部身份提供商并不会取消现有 Cognito-native 用户使用其当前凭据登录的权限。

- 联邦用户受到与 Cognito 用户池相同的区域可用性限制。有关更多信息，请参阅[区域部署](#)。
- 在将联合登录推广到您的组织之前，先与一小群用户一起测试该登录。

禁用或删除默认 Cognito 用户

配置联合后，您可能需要禁用或删除在堆栈部署期间创建的默认用户。这是可选的 — 默认用户继续使用联合登录。

要禁用用户，请在 [Amazon Cognito 控制台中导航到该解决方案的 Cognito 用户池](#)，选择用户选项卡，选择用户，然后选择禁用用户访问权限。要删除用户，必须先将其禁用，然后选择删除用户。禁用用户会撤消其令牌并阻止登录，同时保留该帐户；删除会永久删除该帐户。

有关更多详细信息，请参阅 Amazon Cognito 开发者指南中的[管理和搜索用户账户](#)。

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 的地区部署此解决方案。有关各地区的最新可用服务，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。

成本

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 vCPU 和 4 GB 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 个 vCPU 和 4 GB 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 美元
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 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](#)。

ALB + ECS Fargate 托管 Web 控制台额外费用 (可选)

如果您选择 ALB + ECS Fargate 部署选项，则需要支付以下额外费用。定价适用于美国东部 (弗吉尼亚北部) 地区。

服务组件	Dimensions	成本 [美元]
Elastic Load Balancing	1 Application 负载均衡器	19.35 美元
AWS Fargate	2 个任务 (0.25 个 vCPU、0.5 GB 内存) 正在运行 24/7	17.77 美元
Amazon Virtual Private Cloud (VPC)	3 个接口终端节点，2 个可用区	43.21 美元
亚马逊 CloudWatch	10 个指标、1,000 个 GetMetricData 请求、1,000 个 API 请求、5 GB 日志摄取、1 个控制面板、10 个警报	6.54 美元
Amazon S3	0.1 GB S3 标准存储空间、100 个 PUT/COPY/POST/LIST 请求、10,000 个 GET/SELECT 请求	0.01 美元
AWS WAF	1 个 Web ACL、3 个托管规则组 (CRS、AnonymousIpList) AmazonIpReputation List、10,000 个请求 (请求成本可以忽略不计)	8.00 美元
估计的额外费用总额 :		每月 94.88 美元

价格可能会发生变化。有关完整详情，请参阅 [Elastic Load Balancing 定价](#)、[AWS Fargate 定价](#) 和 [AWS WAF 定价](#)。

安全性

当您在 AWS 基础设施上构建系统时，AWS 和您如何共同分担安全责任。此[责任共担模式](#)能够减轻您的运营负担，因为 AWS 负责运行、管理和控制各种组件，包括主机操作系统、虚拟化层和服务运行所在设施的物理安全性。有关 AWS 安全性的更多信息，请访问 [AWS 云安全性](#)。

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。

亚马逊 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，使用默认 API Gateway 终端节点而不是自定义域为负载测试功能提供 RESTful API。对于使用默认终端节点的边缘优化 API，API Gateway 使用 TLS-1-0 安全策略。有关更多信息，请参阅 Amazon [API Gateway 开发者指南中的使用 REST API](#)。

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 API 设置自定义域名](#)。

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 Inter-Domain 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。

Amazon VPC

VPC：基于 Amazon VPC 服务的虚拟私有云 (VPC) 为您提供 AWS 云中逻辑隔离的私有网络。

部署期间，您可以在 [AWS CloudFormation 参数](#) 中指定自己的 VPC。VPC 仅供产生负载的 ECS 任务使用；Web 控制台和 API 未部署在此 VPC 中。如果您未指定现有 VPC，则解决方案将使用所需的网络配置创建一个新 VPC。如果您选择使用现有 VPC，则它必须满足以下要求才能成功运行负载测试任务。

VPC 要求

下面列出了在 AWS 上使用 VPC 进行分布式负载测试的最低要求。

- VPC 必须包含至少两个可用区
- VPC 必须包含至少两个子网，每个子网位于单独的可用区中
- VPC 子网可以是公共子网也可以是私有子网，但它们必须使用相同的配置（公有或两者均为私有）
- VPC 必须提供对 ECR、CloudWatch 日志、S3 和物联网核心端点的访问权限。
- VPC 必须提供对负载测试所针对服务的访问权限。

Note

如果您没有符合这些条件的 VPC，则可以使用 VPC 向导快速创建 VPC。有关更多信息，请参阅 [创建 VPC](#)。

公有子网可以通过包括以下内容来满足这些要求：

- 连接到 VPC 的互联网网关
- 通往互联网网关的路由 (0.0.0. 0/0)

私有子网可以通过使用 NAT 网关或 VPC 终端节点来满足这些要求，如下所述。

选项 1：NAT 网关

- 在每个带有私有子网的可用区中部署 NAT 网关
- 配置路由表以路由互联网流量 (0.0.0. 0/0) 通过 NAT 网关

选项 2：VPC 终端节点

在您的 VPC 中创建以下 VPC 终端节点：

- 亚马逊 ECR API 终端节点：`com.amazonaws.<region>.ecr.api`
- Amazon ECR DKR 终端节点：`com.amazonaws.<region>.ecr.dkr`
- Amazon CloudWatch 日志终端节点：`com.amazonaws.<region>.logs`
- 亚马逊 S3 网关终端节点：`com.amazonaws.<region>.s3`
- AWS IoT Core 终端节点（如果使用实时数据图表，则为必填项）`com.amazonaws.<region>.iot.data`

其他 VPC 配置也可能起作用。

Important

连接到每个 VPC 终端节点接口的安全组必须允许来自 ECS 任务安全组的端口 443 上的入站 TCP 流量。

安全组配置

在部署期间，该解决方案将在您的 VPC 内创建一个安全组，以允许 ECS 集群中的任务进行以下流量：

- 所有出站流量
- 来自同一安全组中其他任务的端口 50000 上的入站流量，以促进工作人员和领导者任务之间的协调。

网络压力测试

根据 [Amazon EC2 测试政策](#)，您有责任使用此解决方案。该政策涵盖从 Amazon EC2 实例向其他 Amazon EC2 实例、AWS 服务或外部终端节点运行的大容量网络测试。这些测试有时被称为压力测试、负载测试或比赛日测试。查看该政策，了解网络压力测试和 DDoS 模拟（EC2 禁止使用，D [DoS 模拟测试政策单独涵盖这些模拟](#)）之间的区别，并注意 AWS 可能会在高流量下采用流量工程或整形。在运行大批量测试之前，请参阅政策页面了解当前阈值和指导。

限制对公共用户界面的访问

限制 Web 控制台访问权限的方法取决于您选择的部署选项。

默认 (CloudFront + S3) 部署 — 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，您可以将 AWS WAF Web ACL 与分配相关联。CloudFront 考虑使用 [AWS WAF 安全自动化解决方案](#)，该解决方案部署了一组预配置的 AWS WAF 规则，用于过滤常见的基于 Web 的攻击。默认 CloudFront + S3 模板不会自动部署 WAF 资源。

ALB + ECS Fargate 部署 — 该解决方案自动在 ALB 前面部署 AWS WAF Web ACL，其中包含托管规则，可提供针对常见基于 Web 的攻击的基准保护。您可以自定义 WAF 规则以满足您的特定安全要求，包括添加 IP-based 允许或阻止列表、地理限制、速率限制或其他 AWS 托管规则组。有关修改 WAF 配置的说明，请参阅部署说明中的 [WAF 集成](#) 部分。

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 Amazon Bedrock 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。

ALB + ECS Fargate 托管的 Web 控制台安全（可选）

如果您选择 ALB + ECS Fargate 部署选项，则需要考虑以下安全注意事项：

- **VPC 区块公共访问兼容性** — ALB + ECS Fargate 选项专为 VPC 阻止公共访问 (BPA) 策略阻止来自公共分配的流量的环境而设计。CloudFront ALB 可以作为内部负载均衡器部署在您的 VPC 中，只能通过您的公司网络、VPN 或 AWS 进行访问 PrivateLink，满足零公共互联网暴露要求。

- ACM 证书管理 — ALB 使用 ACM 证书进行 HTTPS 终止。您有责任确保证书保持有效并在到期前续订。ACM 会自动续订其管理的证书，但必须手动续订导入的证书。有关更多信息，请参阅 AWS Certificate Manager 用户指南中的[托管证书续订](#)。
- AWS WAF 保护 — 默认情况下，使用 ALB + ECS Fargate 模板部署 WAF。有关详细信息，请参阅[限制对公共用户界面的访问](#)。

无头（自带网络服务器）安全（可选）

如果您选择无头部署选项并在自己的 Web 服务器上托管 Web 控制台，则需要考虑以下安全注意事项：

- HTTPS 配置 — 我们强烈建议您在您的网络服务器上配置 HTTPS。
- 访问控制-您负责在您的 Web 服务器上实施访问控制、防火墙规则和网络安全。
- 安全加强-将贵组织的安全强化标准应用于 Web 服务器，包括修补、监控和入侵检测。

Third-party 测试框架

AWS 上的分布式负载测试捆绑了三个第三方测试框架：Apache JMeter、Grafana K6 和 Locust。在[AWS 责任共担模式](#)下，您负责在运行负载测试之前评估这些框架及其捆绑版本是否符合贵组织的安全要求。该解决方案无需修改即可分发每个框架，并在构建时和运行时使用 SHA512 校验和验证捆绑的二进制文件。

有关每个框架何时安装以及如何配置的详细信息，请参阅[测试框架配置](#)。

Apache JMeter

捆绑版本的 Apache JMeter 存在已知的安全漏洞，如果不破坏与 Taurus 测试自动化框架和解决方案所依赖的 JMeter 插件生态系统的兼容性，就无法在外部完全修补这些漏洞。在运行负载测试之前，请查看[Apache JMeter 安全公告](#)，并评估它们是否会给您带来安全漏洞。

Note

对于单个 HTTP 端点测试类型，Apache JMeter 也在幕后运行。在 Web 控制台中配置 URL、方法、标头和正文负载时，该解决方案会生成 JMeter 测试计划并使用捆绑的 JMeter 二进制文件执行该计划。因此，本节中描述的 JMeter 安全注意事项也适用于单个 HTTP 端点测试。

如果您需要修补版本的 JMeter，则有两种选择。这两个选项都需要测试存档，并且仅适用于 JMeter 测试类型：

- 提供已修补的 JMeter 二进制文件 — 在测试存档中加入已修补的 JMeter 二进制文件。该解决方案使用您的二进制文件代替捆绑版本。
- 覆盖单个插件 JAR-使用插件覆盖机制将特定易受攻击的插件 JAR 替换为已修补的版本。有关更多信息，请参阅 [JMeter 测试](#)。

单个 HTTP 端点测试类型不接受测试档案，因此无法覆盖捆绑的 JMeter 二进制文件或插件。如果您需要使用已修补的 JMeter 运行 HTTP 端点测试，请使用 jMeter 测试类型并提供 jMeter 脚本 (.jmx) 或包含已修补的 JMeter 二进制文件或插件 JAR 的 .zip 存档。

Grafana K6

K6 是根据 [AGPL-3.0 许可](#) 发布的。当您创建新的 K6 测试时，Web 控制台会显示许可证确认消息。在本解决方案发布时，在捆绑版的 K6 中未发现任何已知的安全漏洞。该解决方案不会持续监控 K6 是否存在新漏洞；您有责任在使用 K6 的整个过程中根据自己的安全要求对其进行评估。

蝗虫

在本解决方案发布时，在捆绑版本的 Locust 中未发现任何已知的安全漏洞。该解决方案不会持续监控 Locust 是否存在新漏洞；您有责任在使用 Locust 的整个过程中根据您的安全要求对其进行评估。

支持的 AWS 区域

AWS 上的分布式负载测试适用于以下 AWS 区域：

Note

要按地区了解本解决方案中使用的 AWS 服务的最新可用性，请参阅 [AWS 区域服务列表](#)。

区域名称	区域代码
美国东部 (弗吉尼亚州北部)	us-east-1
美国东部 (俄亥俄州)	us-east-2
美国西部 (加利福尼亚北部)	us-west-1

区域名称	区域代码
美国西部 (俄勒冈州)	us-west-2
加拿大 (中部)	ca-central-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
欧洲地区 (巴黎)	eu-west-3
欧洲地区 (斯德哥尔摩)	eu-north-1
亚太地区 (孟买)	ap-south-1
亚太地区 (东京)	ap-northeast-1
亚太地区 (首尔)	ap-northeast-2
亚太地区 (新加坡)	ap-southeast-1
亚太地区 (悉尼)	ap-southeast-2
南美洲 (圣保罗)	sa-east-1

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 AgentCore 网关的 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
美国东部 (俄亥俄州)	us-east-2
美国西部 (俄勒冈州)	us-west-2

区域名称	区域代码
加拿大 (中部)	ca-central-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
欧洲地区 (巴黎)	eu-west-3
欧洲地区 (斯德哥尔摩)	eu-north-1
亚太地区 (孟买)	ap-south-1
亚太地区 (东京)	ap-northeast-1
亚太地区 (首尔)	ap-northeast-2
亚太地区 (新加坡)	ap-southeast-1
亚太地区 (悉尼)	ap-southeast-2
南美洲 (圣保罗)	sa-east-1

要了解按地区划分的 AgentCore 网关的最新可用性，请参阅 AWS 一般参考中的 [Amazon Bedrock AgentCore 终端节点和配额](#)。

Tip

如果您想使用 MCP 服务器，但需要从不支持 AgentCore 网关的 AWS 区域发送负载测试流量，则可以在支持的网 AgentCore 关区域部署主中心堆栈（启用 MCP 服务器），然后在要生成测试流量的 AWS 区域 [Multi-Region 部署区域堆栈](#)（参见 [部署](#)）。这允许您使用 MCP 服务器进行 AI-assisted 分析，同时仍从所需区域运行负载测试。

配额

服务配额（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。

此解决方案中的 AWS 服务的配额

请确保[此解决方案中实施的每项服务](#)都有足够的配额。有关更多信息，请参阅 [AWS 服务配额](#)。

通过以下链接转到相关服务的页面。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，请改为在 PDF 中查看[服务端点和配额](#)页面的信息。

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](#)。通过了解这些配额，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

您有责任根据 [Amazon EC2 测试政策运行负载测试](#)。查看政策，了解当前的指导方针，了解哪些内容符合网络压力测试资格、允许的流量以及在运行大容量测试之前何时联系 AWS。请注意，AWS 可能会在高总流量下应用流量工程或整形。

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。

聘请专家

AWS 倒计时高级短期项目，用于在 AWS 上进行分布式负载测试

我们的 AWS 工程师在性能测试基础知识、脚本开发和结果分析方面提供专家指导。[立即注册](#)。

概述

AWS Countdown Premium (CDP) 短期项目为组织大规模进行性能测试提供专家指导。通过协作的“自己动手”模式，AWS 工程师提供战略监督和技术专业知识，同时您的团队负责执行。AWS 专家工程师可在注册后一周内到场，无需签订长期合同。

服务模型

CDP 工程师与您的团队合作，在整个性能测试实施过程中提供指导和监督。这种放手的方法可确保您在建立内部能力的同时获得专家指导。该服务非常适合具有现有测试能力、需要专业的 AWS 专业知识才能在 AWS 上有效实施分布式负载测试的组织。

CDP 工程师提供什么

CDP 工程师将指导您完成性能测试基础知识和 AWS 架构上的分布式负载测试。他们提供有关 JMeter、K6 和 Locust 脚本结构和测试脚本开发的指导，协助 CloudFormation 模板部署，并根据性能优化建议评估测试结果。Support 包括资源利用率分析、最佳实践调整以及从初始设置到结果分析的端到端指导，从而为您的团队传授知识。

客户责任

您的团队负责应用程序级配置、测试脚本开发和测试场景验证。您对实际的测试执行和操作负责，包括性能测试事件之前、期间和之后的所有测试活动。

主要优势

CDP 短期合作通过专家监督、针对您的工作量的情境指导、性能优化建议、更快的问题解决、最佳实践调整和全面的支持，降低风险，同时保持团队的所有权和能力发展。

支持的架构

AWS 上的分布式负载测试支持利用 AWS 上的分布式负载测试解决方案大规模测试 Web 应用程序、API、微服务和无服务器架构。测试功能远远超出了这些常见用例，还包括数据库、TCP/UDP 协议、LDAP 目录、SMTP 邮件服务器以及许多其他需要在负载下进行性能验证的系统和协议。

开始使用

对在 AWS 上进行分布式负载测试的 CDP 短期合作感兴趣的组织可以直接通过[此处](#)的 AWS 网站注册，然后为您的重点领域选择“用例实施”。

超出范围

CDP 不提供自定义测试脚本开发（仅提供指导）、管理测试执行操作或创建自定义动手实验室或研讨会。On-site 支持也超出了范围。

部署解决方案

该解决方案提供三个 Web 控制台托管选项。所有选项的后端架构和 Amazon Cognito 身份验证模型都是相同的。

托管选项	说明	何时使用
CloudFront + S3 (默认)	Web 控制台托管在 Amazon S3 中，并通过亚马逊提供服务 CloudFront。	建议用于大多数部署。提供全局边缘缓存，无需其他基础架构。
ALB + ECS Fargate	Web 控制台在 Application Load Balancer 后面的 ECS Fargate 服务上运行，该服务具有客户提供的 ACM 证书和自定义域。AWS WAF Web ACL 部署在 ALB 前面，用于过滤常见的基于 Web 的攻击。	在 VPC 阻止公共访问 (BPA) 策略阻止来自公共 CloudFront 分配的流量、您的组织要求零公共 Internet 暴露时、需要为 Web 控制台使用自定义域名时，或者需要使用可自定义规则保护 Web 应用程序防火墙时，使用此选项。常见于受监管行业（金融、医疗保健、政府）和合规要求严格的环境。
Headless (自带网络服务器)	该解决方案仅部署后端，并提供可下载的 Web 控制台静态资产压缩包，以便在您自己的 Web 服务器上托管。	当您想在 AWS 以外的现有基础设施上托管 Web 控制台、公司政策要求在托管 Web 服务器上托管，或者您的组织要求完全网络隔离，且无论身份验证机制如何，都不得使用面向公众的终端节点时，请使用。

[AWS Launch Wizard](#) 是此解决方案的推荐部署方法。它提供了以下功能：

- 指导式配置体验，每个步骤都有详细的帮助面板
- 用于监控所有部署运行状况的集中式页面
- 指示何时拥有最新版本的解决方案可供部署或升级

或者，您可以使用 [AWS CloudFormation 模板](#) 直接部署解决方案。

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

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。通过此调查收集的数据归 AWS 所有，数据收集受 [AWS 隐私声明](#) 的约束。

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。

使用 AWS Launch Wizard 进行部署

该解决方案采用使用 AWS Launch Wizard 的指导部署流程。按照以下步骤将 AWS 上的分布式负载测试部署到您的账户。

1. 登录 AWS 管理控制台并选择下面的按钮开始部署过程。

Launch solution

2. 如果该解决方案有多个部署模式可用，请选择最适合您的用例的部署模式。
3. 选择要部署的版本。建议使用最新版本。
4. 选择“启动部署向导”按钮。

然后，您将按照一系列步骤收集部署解决方案所需的信息。配置所需资源大约需要 15 分钟。

从“部署”[列表中选择您的部署](#)以查看其状态。

使用 AWS 进行部署 CloudFormation

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。

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这些模板：

[View template](#)

上的 [distributed-load-testing-testing-on-template](#)-使用此模板启动带有默认 + S3 CloudFront 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。

[View template](#)

布式负载测试-on-aws-alb-ecs.template-使用此模板启动解决方案，将托管在 ECS Fargate 上的 Web 控制台托管在 ALB 后面。有关部署说明，请参阅[使用 ALB + ECS Fargate 进行部署](#)。

[View template](#)

headless.template.template-使用此模板仅启动解决方案后端，Web 控制台资产可供下载。有关部署说明，请参阅[使用无头模板进行部署（自带 Web 服务器）](#)。

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（默认，CloudFront + S3 托管 Web 控制台）

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

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。

- 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。有关各地区的最新可用服务，请参阅 [AWS 区域服务列表](#)。

- 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。
- 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。
- 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	说明
管理员姓名	<需要输入>	初始解决方案管理员的用户名。
管理员电子邮件	<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。
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

参数	默认值	说明
		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。
Auto-update 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。
部署可选 MCP 服务器	No	部署可选的远程 MCP 服务器，使用 AgentCore 网关将 AI 应用程序连接到 AWS 上的分布式负载测试。

- 选择 Next(下一步)。
- 在配置堆栈选项页面上，请选择下一步。
- 在审核页面上，审核并确认设置。选中确认模板将创建 AWS Identity and Access Management (AWS IAM) 资源的复选框。
- 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。

启动堆栈 (ALB + ECS Fargate 托管的 Web 控制台)

先决条件

在部署 ALB + ECS Fargate 模板之前，您必须完成以下操作：

1. ACM 证书 — 在您将部署堆栈的同一区域的 [AWS Certificate Manager \(ACM\) 中申请或导入证书](#)。SSL/TLS 证书必须包含您计划用于 Web 控制台的域名。
2. 域名所有权-您必须拥有或控制将在 ConsoleDomainName 参数中指定的域名。部署后，您将需要能够为此域创建 DNS 记录。

启动 堆栈

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，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。有关各地区的最新可用服务，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。除了默认模板中的标准参数外，此模板还使用以下参数。

参数	默认值	说明
控制台域名	<需要输入>	Web 控制台的自定义域名 (例如，dlt.example.com)。此域必须与您在先决条件中创建的 ACM 证书相匹配。
ACM 证书 ARN	<需要输入>	域名的 ACM 证书的 ARN (例如)。arn:aws:acm:us-east-1:123456789012:certificate/abcd1234-ef56-gh78-ij90-klmnopqrstuv 必须与堆栈位于同一区域。
管理员姓名	<需要输入>	初始解决方案管理员的用户名。
管理员电子邮件	<需要输入>	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	说明
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。
Auto-update 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	说明
Web 控制台镜像 URI	<Optional input>	来自 Amazon ECR 私有注册表的自定义 Web 控制台容器镜像的 URI (例如 , 123456789012.dkr.ecr.us-east-1.amazonaws.com/my-web-console:latest)。如果为空 , 则使用默认的公共镜像。有关更多信息 , 请参阅 Web 控制台镜像 。
部署 WAF	Yes	使用 AWS 托管规则在 ALB 前面部署 AWS WAF Web ACL , 以过滤常见的基于 Web 的攻击。设置 No 为可禁用 WAF 部署。
部署可选 MCP 服务器	No	部署可选的远程 MCP 服务器 , 使用 AgentCore 网关将 AI 应用程序连接到 AWS 上的分布式负载测试。

- 选择 Next(下一步)。
- 在配置堆栈选项页面上 , 请选择下一步。
- 在审核页面上 , 审核并确认设置。选中确认模板将创建 AWS Identity and Access Management (AWS IAM) 资源的复选框。
- 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。

Post-deployment 配置

堆栈创建完成后 , 您必须配置 DNS 以将您的自定义域指向 ALB。

1. 导航到 CloudFormation 堆栈的“输出”选项卡并复制该 ALB DnsName 值。
2. 在您的 DNS 提供商中，创建以下记录之一：
 - CNAME 记录 — 将您的域名映射到 ALB 的 DNS 名称。适用于子域名（例如，dlt.example.com）。
 - 别名记录（Route 53）— 如果您使用 Amazon Route 53，则可以创建指向 ALB 的别名记录。对于区域顶点域（例如 example.com），这是必需的，并且可以避免 CNAME 查询费用。

在 Route 53 控制台中，创建启用别名的 A 记录，选择应用程序别名和 Classic Load Balancer，选择区域，然后选择 ALB。

3. 等待 DNS 传播完成。您可以通过以下方式进行验证：

```
$ dig dlt.example.com
```

4. 访问 Web 控制台，网址为 `https://<your-domain>`（例如，`https://dlt.example.com`）。

Web 控制台 URL 也作为 consoleUrl 在“CloudFormation 输出”选项卡中提供。

Note

DNS 传播可能需要几分钟到 48 小时不等，具体取决于您的 DNS 提供商和 TTL 设置。

WAF 集成（可选）

默认托管规则组（核心规则集、Amazon IP 信誉列表和匿名 IP 列表）为大多数部署提供基准保护。您可以通过将“部署 WAF” CloudFormation 参数设置为来禁用 WAF 部署。除非您有特定的安全要求，否则无需进行额外的 WAF 配置。

如果您需要自定义 WAF 规则，可以按如下方式修改配置：

1. 在与您的部署相同的区域中打开 [AWS WAF 控制台](#)。
2. 选择解决方案创建的 Web ACL。Web ACL 名称可以在 CloudFormation 堆栈资源中找到。
3. 选择“规则”以查看、添加、删除或修改 Web ACL 中的规则。
4. 您可以根据自己的安全需求添加其他 AWS 托管规则组、自定义规则或基于费率的规则。

有关更多信息，请参阅 [AWS WAF 开发人员指南](#) 中的 AWS WAF。

启动堆栈 (Headless)

启动 堆栈

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

A blue button with rounded corners and a white border, containing the text "Launch solution" in white. The button is positioned above a thin blue horizontal line.

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，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。有关各地区的最新可用服务，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。此模板使用与[默认模板](#)相同的参数（管理员姓名、管理员电子邮件、VPC/subnet 配置、出口 CIDR、Auto-update 容器映像和部署可选 MCP 服务器）。不需要额外参数。
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. 在审核页面上，审核并确认设置。选中确认模板将创建 AWS Identity and Access Management (AWS 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。

Post-deployment 配置

堆栈创建完成后，您可以使用 [DLT CLI 与解决方案进行交互](#)，[直接从终端访问 REST API](#)，或者在自己的 Web 服务器上托管 Web 控制台。

要托管 Web 控制台，请执行以下操作：

1. 导航到 CloudFormation 堆栈的“输出”选项卡。
2. 选择 ConsoleAssetsBucket 输出链接以在控制台中打开 S3 存储桶。
3. 从控制台存储桶下载 dlt-web-console.zip 文件。
4. 提取 zip 内容并将网站托管在您的 Web 服务器上。
5. 将 Amazon Cognito 配置为允许您的网络服务器的网址：
 - a. 打开 [Amazon Cognito 控制台](#)，然后选择堆栈创建的用户池。
 - b. 选择应用程序客户端，然后选择应用程序客户端。应用程序客户端 ID 可以在 CloudFormation 堆栈的“输出”选项卡中找到 CognitoAppClientID。
 - c. 在“登录页面”部分中，选择“编辑”。
 - d. 在“允许的回传 URL”下，添加您的 Web 服务器的 URL 和带有尾部斜杠的 URL（例如 <https://dlt.example.com> 和 <https://dlt.example.com/>）。
 - e. 在“允许的注销 URL”下，添加相同的 URL。
 - f. 选择保存更改。

有关更多信息，请参阅 Amazon Cognito 开发者指南中的 [应用程序客户端 Application-specific 设置](#)。

Important

如果您托管 Web 控制台，则负责在 Web 服务器上配置 HTTPS、访问控制和安全加固。我们强烈建议将 HTTPS 用于生产用途。

Note

Web 控制台资产包括一个带有 API 端点网址和 Cognito 设置的配置文件。这些值是在堆栈创建过程中预先配置的。如果将资源移动到其他服务器，请确保配置文件保持不变。

Multi-Region 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，它会在场景 S3 存储桶中创建一个区域 CloudFormation 模板。此模板的网址列在主堆栈的 CloudFormation 输出中，密钥为“regionalCFTemplate”。

要运行多区域测试，您必须在要运行测试的每个区域部署区域 CloudFormation 模板。

Note

每个 AWS 账户在每个区域只能使用一个区域堆栈。此外，区域堆栈不能与主堆栈在同一个区域中使用。

您可以按如下方式安装区域模板：

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 [AWS 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	说明
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

参数	默认值	说明
		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。
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。

- 选择 Next(下一步)。
- 在配置堆栈选项页面上，请选择下一步。
- 在审核页面上，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。
- 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。

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。有关可接受的 cron 格式、支持的模式和约束条件的详细信息，请参阅开发者指南中的[Cron 表达式参考](#)。

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，该函数创建配置为在指定日期运行的 Amazon EventBridge 计划表。

- 对于一次性测试（运行一次），EventBridge 计划程序计划在指定的日期和时间调用 `Lambda::InvokeFunction` 函数，由该函数执行测试。
- 对于重复测试（按计划运行），计划 EventBridge 程序计划会立即按照 cron 或速率表达式定义的节奏调用 `Lambda::InvokeFunction` 函数，直到到期日。

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）。

Note

所有四种测试类型都依赖于第三方组件。该解决方案通过 Taurus 测试自动化框架运行测试，该框架根据测试类型执行 jMeter、K6 或 Locust；单个 HTTP 端点测试将转换为 jMeter 测试计划，由捆绑的 Apache JMeter 运行。在创建测试之前，请查看[Third-party 测试框架](#)以了解安全注意事项、许可证信息和修补选项。

HTTP 端点配置

选择“单个 HTTP 端点”后，该解决方案会根据您的配置生成 JMeter 测试计划，并使用捆绑的 Apache JMeter 二进制文件执行该计划。配置以下设置：

HTTP 终端节点 (必填)

输入您要测试的终端节点的完整 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)

在您的请求中添加自定义 HTTP 标头。常见示例包括：

- 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，您可以在 .zip 存档的 /plugins 文件夹中包含自定义插件。

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。

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。

Multi-region 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。建议的限制基于每个任务 2 个 vCPU 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. 在导航窗格的“容器见解”下，选择“性能监控”。
 - b. 在性能监控页面的左下拉菜单中，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。
- 每个任务的 vCPU-分配给每个任务的虚拟 CPU 数量（默认值：2）。
- DLT 任务限制-根据您的账户的 Fargate 按需 vCPU 配额可以创建的最大任务数。新账户的配额通常较低；请在 Service Quotas 控制台中验证您当前的配额，并在需要时申请增加配额。
- 可用的 DLT 任务-该区域当前可用的任务数，计算方法是您的 DLT 任务限制减去运行 Fargate 任务时已在使用的 vCPU。

要增加每个任务的可用任务或 vCPU 数量，请参阅开发者指南。

测试时长

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。任务状态表显示每个区域的实时信息。

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-每个任务的虚拟用户数
- 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 区域配置容器（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。

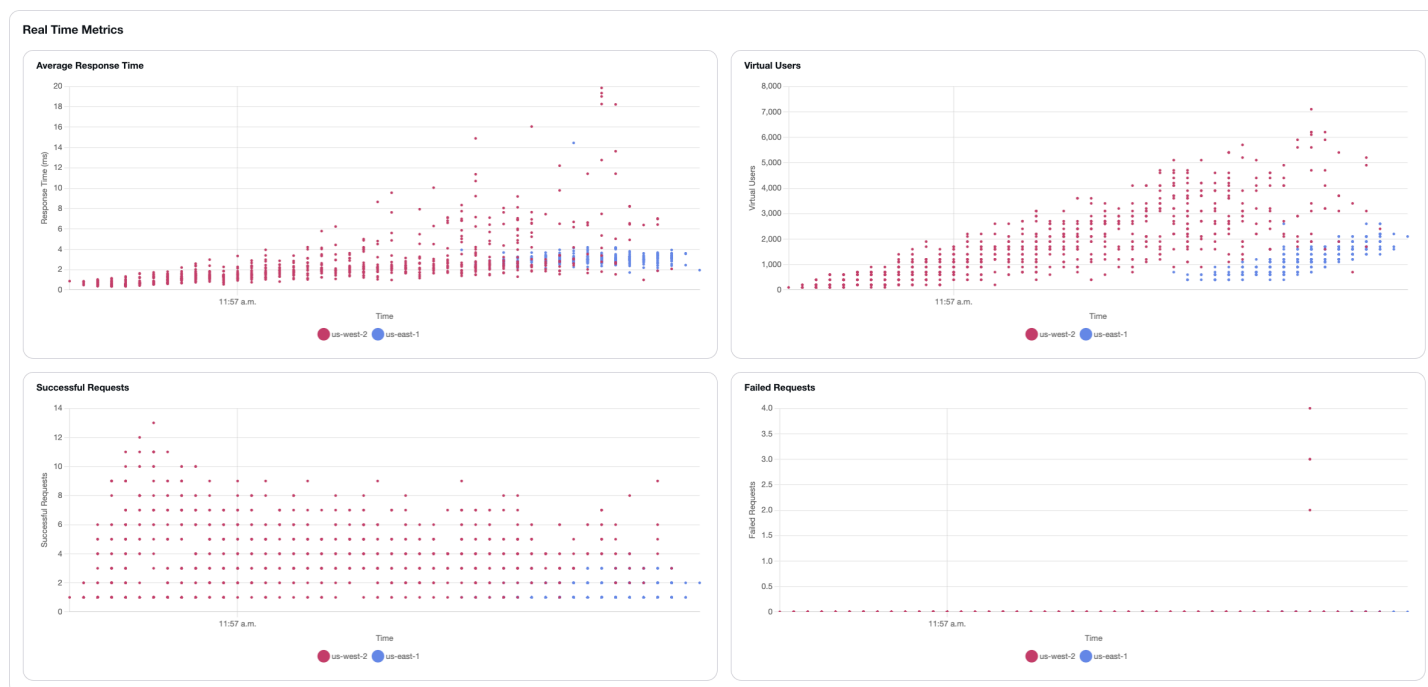
测试运行状态

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-测试运行遇到了错误。
- 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）。以秒为单位 Y-axis 显示响应时间，X-axis 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。

Multi-region 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. `microservicesAPI` 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。您可以：

- 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。

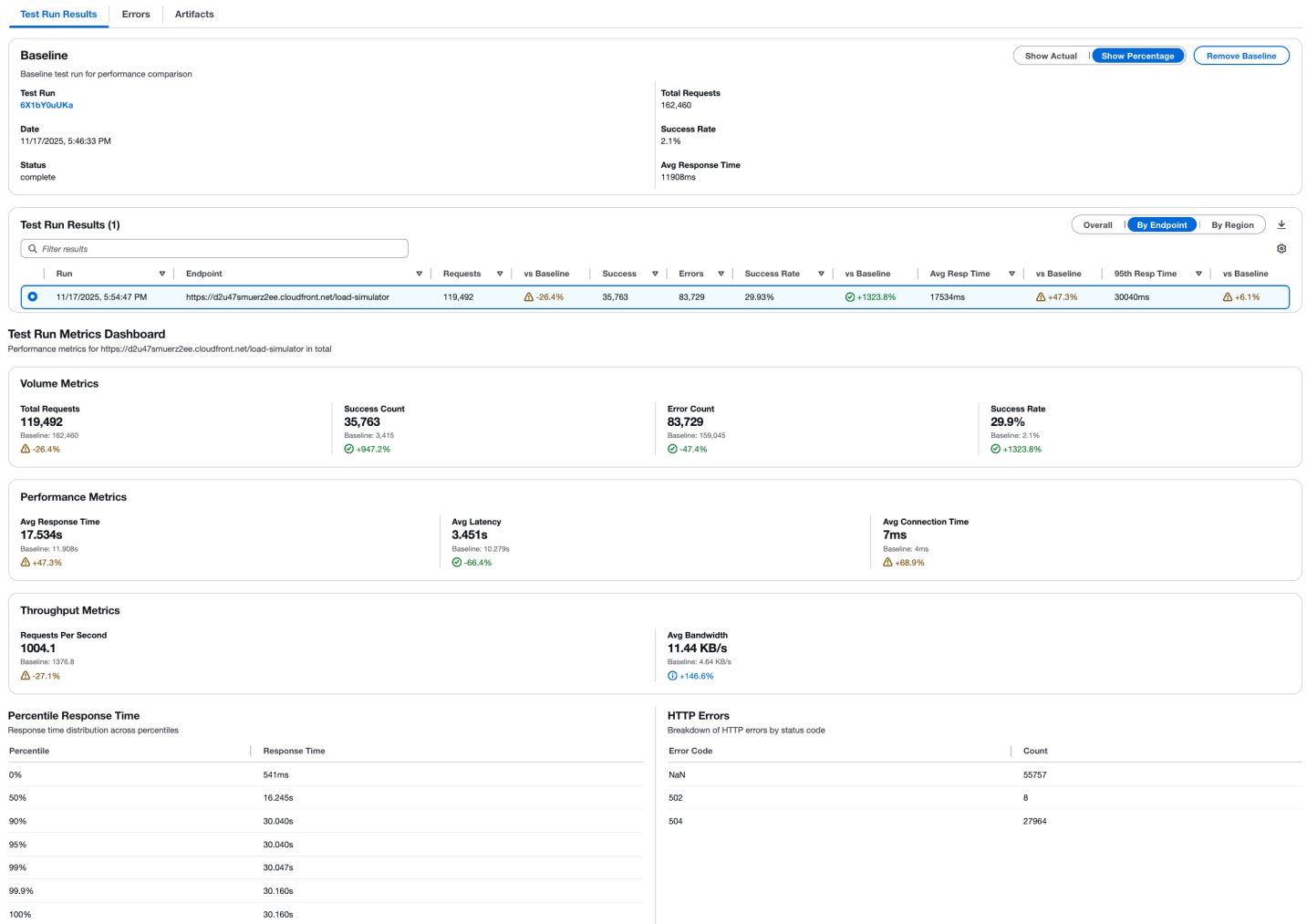
基线比较

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。
- 您可以随时更改或删除基线。

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
- 按区域-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。

“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 Taurus 用户手册中的[生成测试报告](#)。

使用 CloudWatch 日志见解进行监控

此解决方案部署了四个 CloudWatch Logs Insights 保存的查询，这些查询显示在 [CloudWatch Logs Insights 控制台](#) 的“已保存查询”这些查询无需您编写自定义查询即可查看测试编排、错误、任务失败和孤儿清理。

每个查询名称都包含堆栈名称和区域，以便在多个部署中进行识别。例如：DLT - Test Timeline [my-stack us-east-1]。

使用已保存的查询

要运行已保存的查询，请执行以下操作：

1. 打开“[CloudWatch 日志见解](#)”控制台。
2. 在左侧面板中，展开“已保存的查询”。
3. 选择前缀为 DLT - 查询。
4. 如果查询包含占位符值 REPLACE_WITH_TEST_RUN_ID，例如，请将其替换为测试结果中的实际测试运行 ID。
5. 选择运行查询。

DLT-测试时间表

显示跨所有编排 Lambda 函数运行的单个测试的完整生命周期。使用此查询可以跟踪从测试创建到完成的事件顺序。

属性	值
字段	@timestamp , logEvent, message, region, error

属性	值
筛选条件	testRunId = "REPLACE_WITH_TEST_RUN_ID"
排序	@timestamp asc
限制	500

REPLACE_WITH_TEST_RUN_ID替换为您要调查的测试运行 ID。

DLT-测试错误

显示在 Lambda 函数和 ECS 任务之间运行的单个测试的所有 ERROR-level 条目。使用此查询可以确定测试失败或产生意外结果时的根本原因。

属性	值
字段	@timestamp , logEvent, message, region, taskId, error
筛选条件	testRunId = "REPLACE_WITH_TEST_RUN_ID" and level = "ERROR"
排序	@timestamp asc

REPLACE_WITH_TEST_RUN_ID替换为您要调查的测试运行 ID。

DLT-任务失败

显示各个 ECS 任务失败以及停止代码和故障分类。使用此查询来了解测试期间特定的 Fargate 任务停止的原因。

属性	值
字段	@timestamp , testId, testRunId , region, taskArn, stopCode, exitCode, stopCategory , stoppedReason
筛选条件	logEvent = "TASK_FAILURE_DETECTED"

属性	值
排序	@timestamp desc
限制	50

此查询不需要测试运行 ID，它会显示所有测试运行中最近的所有任务失败。

DLT-孤儿清理

显示孤立 ECS 服务检测的历史记录。使用此查询来验证每小时的孤儿清理过程是否在查找和删除废弃的服务。

属性	值
字段	@timestamp , logEvent, message, region, cluster, orphanCount , orphanTestIds
筛选条件	logEvent = "ORPHAN_DETECTED"
排序	@timestamp desc
限制	50

CloudWatch 警报

该解决方案部署了两个 CloudWatch 警报，用于监控需要注意的操作条件。默认情况下，这些警报没有配置任何通知操作。我们建议为每个警报订阅 Amazon SNS 主题，以便操作员在出现问题时立即收到通知。

订阅警报通知

要在警报触发时接收通知，请执行以下操作：

1. 打开[CloudWatch 警报控制台](#)。
2. 搜索以堆栈名称为前缀的警报（例如my-stack-OrphanCleanupFailure）。
3. 选择闹钟并选择编辑。

4. 在“通知”下，选择“添加通知”。
5. 使用您的首选通知终端节点（电子邮件、短信或 Lambda）选择或创建 SNS 主题。
6. 选择更新警报。

对每个闹钟重复此操作。

OrphanCleanupFailure

属性	值
告警名称	{StackName}-OrphanCleanupFailure
指标	OrphanCleanupFailures 在distributed-load-testing 命名空间中
Threshold	>= 5 分钟内有 1 次失败
处理缺失的数据	突破

此警报监控的内容：该解决方案使用三层防御来防止 ECS 服务失控：

- 第 1 层：自动错误处理 — 测试编排工作流程包括每一步的错误处理。如果在配置、稳定或执行过程中出现任何故障，工作流程会自动触发清理以耗尽和删除 ECS 服务。
- 第 2 层：执行失败检测 — 如果编排工作流程本身意外退出（例如，由于超时或内部错误绕过了正常的错误处理），EventBridge 则规则会检测失败并独立触发测试中涉及的每个区域的清理。
- 第 3 层：每小时清理孤儿 — 每小时运行一次定时进程，扫描与任何活动测试无关的 ECS 服务，然后强制删除它们。这是万不得已的安全网——如果第 1 层和第 2 层都出现故障，泄露的服务仍将在一小时内删除。如果孤儿清理过程本身失败，则会触发此警报。

为何重要：孤立 ECS Fargate 服务继续运行并产生费用，但在 DLT 控制台中看不见。如果没有通知订阅，运营商只有在账单上出现意外费用时才会发现问题。

推荐响应：当此警报触发时，导航到 [Amazon ECS 控制台](#)，找出 DLT 集群中与正在运行的测试不对应的服务，然后手动将其删除。

MetricFilterCount

属性	值
告警名称	{StackName}-MetricFilterCount-Alarm
指标	MetricFilterCount 在distributed-load-testing 命名空间中
Threshold	>= 90
处理缺失的数据	未突破

此警报监控的内容：该解决方案在 ECS 日志组上动态创建 CloudWatch 指标筛选器，以支持测试执行期间的实时指标。AWS 将每个日志组限制为 100 个指标筛选条件。当使用量达到该限制的 90% 时，将触发此警报。

为何重要：如果达到限制，新的负载测试运行将失败。

推荐回应：删除不再需要的测试场景。删除测试场景后，该解决方案会删除关联的指标筛选条件并释放容量以进行新的测试。

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，可通过 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、Kiro CLI、Cline 和 Amazon Quick 的设置说明。

第 1 步：获取 MCP 端点和访问令牌

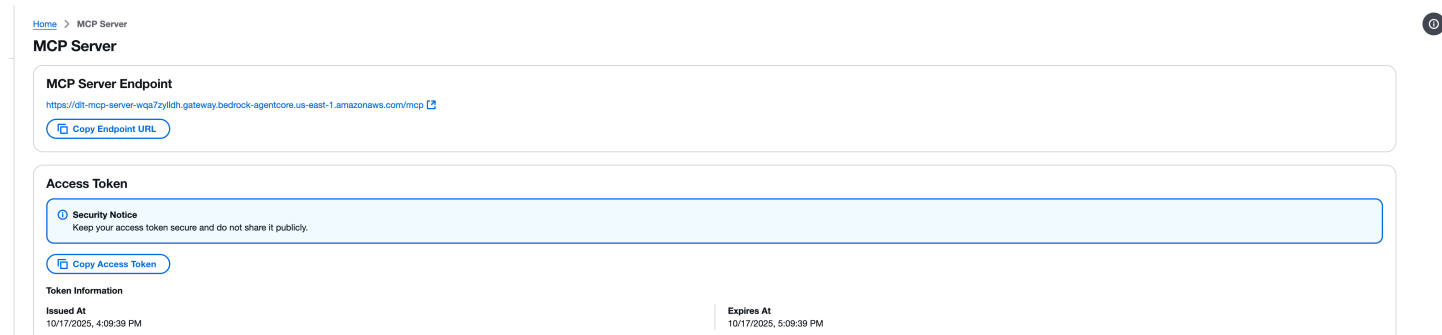
在配置任何 MCP 客户端之前，您需要从 DLT Web 控制台检索 MCP 服务器端点和访问令牌。

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://{gateway-id}.gateway.bedrock-agentcore.{region}.amazonaws.com/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。

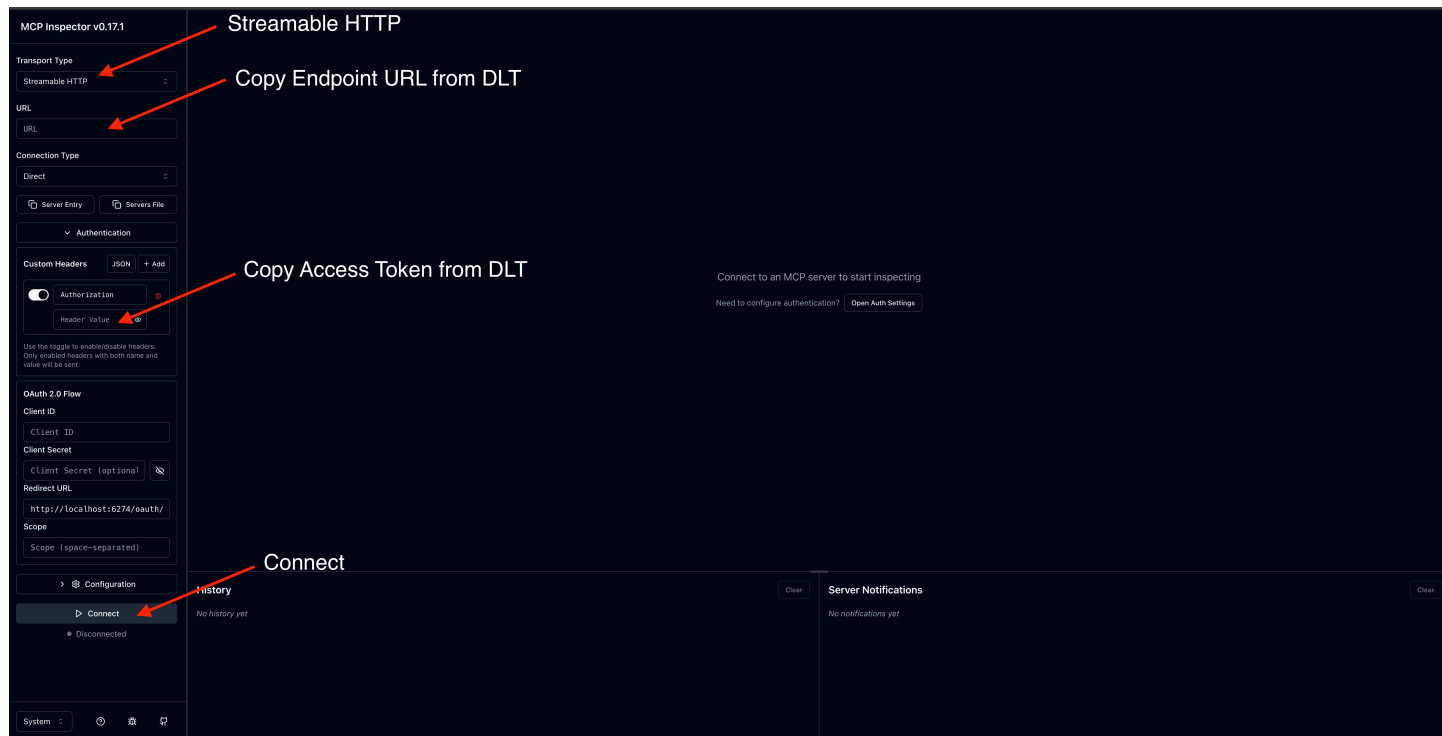
安装并启动 MCP Inspector

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

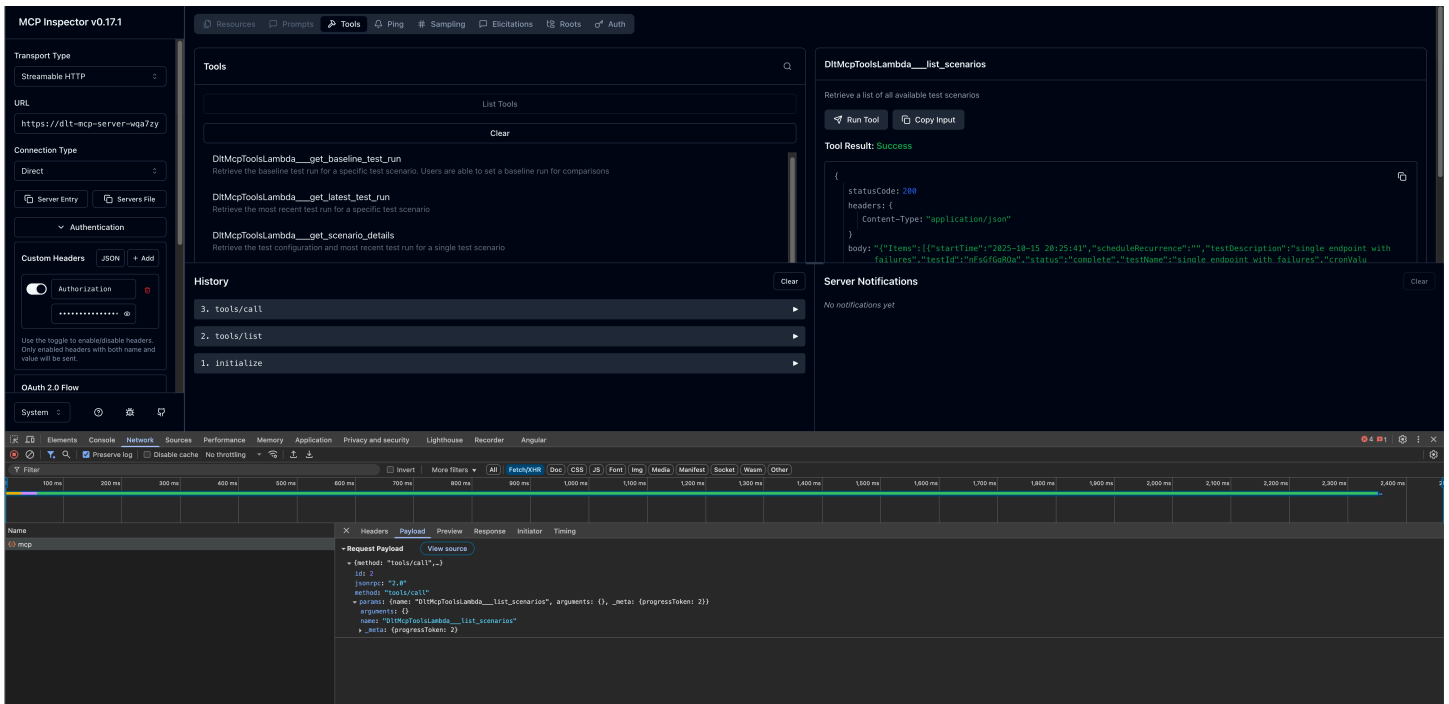
1. 在 MCP Inspector 界面中，输入你的 MCP 服务器端点 URL。
2. 添加带有访问令牌的授权标头。
3. 选择 Connect 以建立连接。



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. 选择 Invoke 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。

Kiro CLI

Kiro CLI (前身为 Amazon Q Developer CLI) 通过集成 MCP 服务器提供命令行 AI-assisted 开发权限。

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 Kiro CLI 文档中的[模型上下文协议 \(MCP\)](#)。
2. 添加您的 DLT MCP 服务器配置：

```
{
  "mcpServers": {
    "dlt-mcp": {
      "type": "http",
      "url": "https://<gateway-id>.gateway.bedrock-agentcore.<region>.amazonaws.com/mcp",
      "headers": {
        "Authorization": "Bearer <access-token>"
      }
    }
  }
}
```

```
    }  
  }  
}  
}
```

将<gateway-id>和<region>替换为您的 MCP 服务器端点 URL 中的值<access-token>，以及您在步骤 1 中复制的值。

验证配置

1. 在终端中，键入`kiro-cli`以启动 Kiro CLI。
2. 键入`/mcp`以查看所有可用的 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。

克莱恩

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 MCP 服务器。
2. 更新 `cline_mcp_settings.json` 文件：

```
{  
  "mcpServers": {  
    "dlt-mcp": {  
      "type": "streamableHttp",  
      "url": "https://<gateway-id>.gateway.bedrock-agentcore.<region>.amazonaws.com/  
mcp",  
      "headers": {  
        "Authorization": "Bearer <access-token>"  
      }  
    }  
  }  
}
```

将<gateway-id>和<region>替换为您的 MCP 服务器端点 URL 中的值<access-token>，以及您在步骤 1 中复制的值。

3. 保存配置文件。
4. 重新启动 Cline 以应用更改。

Amazon Quick

Amazon Quick (前身为 Amazon Quick 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 Quick 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 (计算机对机器) 结尾的应用程序客户端，然后将其选中。
8. 在应用程序客户端详细信息页面上，复制客户端 ID。要显示客户机密钥，请选择显示客户机密钥，然后复制该值。
9. 返回用户池并从“域”选项卡中获取用户池域。
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 Quick 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：Service-based 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 Quick 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 ID、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].

问题排查

[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.
```

- 解决方法：

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

- 解决方法：

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

- 解决方法：

如果您选择了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 [EC S 服务就可以通过从 `aws-load-testing-on-aws-load-testing-on-aws-load-testing-on-load-testing-on-load-solutions/distributed` testing 下载解决方案 ECR 映像来配置任务](#) 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。任务定义更新后，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

• 解决方法：

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。在两个 VPC 中配置相应的路由表。请参阅 [VPC 对等互连](#) 文档。
5. Transit Gateway：对于涉及多个 VPC 的更复杂的联网场景，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

• 解决方法：

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。

ALB + ECS Fargate 部署问题

问题：ACM 证书验证停留在“待验证”状态

- 解决方法：

如果您使用 DNS 验证请求公有 ACM 证书，则必须将 ACM 提供的别名记录添加到您的 DNS 配置中。导航到 [ACM 控制台](#)，展开证书详细信息，然后将 DNS 验证记录添加到您的域名的 DNS 提供商。如果您使用了电子邮件验证，请查看与该域关联的电子邮件地址以获取来自 AWS 的验证电子邮件。有关更多信息，请参阅 AW [S Certificate Manager 用户指南中的 DNS 验证](#)。

问题：部署后，Web 控制台返回“502 网关错误”或“503 服务暂时不可用”错误

- 解决方法：

在 [EC2 控制台](#) 中查看 ALB 目标组运行状况检查。如果 ECS Fargate 任务显示为运行状况不佳，请在 [ECS 控制台](#) 中验证任务是否正在运行，并检查任务日志中 CloudWatch 是否存在错误。确保附加到 ECS 任务的安全组允许来自 ALB 安全组的入站流量。

问题：已配置 DNS 但自定义域无法解析到 Web 控制台

- 解决方法：

验证您的 DNS 提供商中是否正确配置了 CNAME 记录，将您的自定义域（例如 `console.example.com`）映射到输出中的 ALB DNS 名称。CloudFormation DNS 传播最多可能需要 48 小时，具体取决于您的 DNS 提供商和 TTL 设置。您可以使用 `dig console.example.com CNAME` 或验证 DNS 记录 `nslookup console.example.com`。

问题：Web 控制台在部署后返回“403 禁止”错误

- 解决方法：

部署在 ALB 前面的 AWS WAF Web ACL 可能阻止了合法请求。打开 [AWS WAF 控制台](#)，选择与 ALB 关联的 Web ACL，然后选中“采样请求”选项卡，确定哪条规则阻塞了流量。您可以修改 WAF 规则以允许被屏蔽的请求。例如，如果托管规则组产生误报，则可以将特定规则操作设置为“计数”而不是“阻

止”，以便在仍记录流量的同时允许流量。有关更多信息，请参阅 [AWS WAF 开发人员指南中的测试和调整您的 AWS WAF 保护](#)。

无头（自带 Web 服务器）部署问题

问题：Web 控制台在连接到 API 时显示 CORS 错误

• 解决方法：

Cross-Origin 当您的域上托管的 Web 控制台尝试调用解决方案的 API Gateway 端点时，就会发生资源共享 (CORS) 错误。请确保您的网络服务器通过 HTTPS 为控制台提供服务，因为 API Gateway 端点需要 HTTPS。验证您的 Web 服务器的源域是否与 API Gateway CORS 设置中配置的允许来源相匹配。如果您使用的是自定义域，则可能需要更新 API Gateway CORS 配置以包含您的域。

问题：Web 控制台加载但身份验证失败或重定向不正确

• 解决方法：

Web 控制台资产包括一个包含 Cognito 用户池设置和 API 端点网址的配置文件。确认在解压缩期间未修改此配置文件。请确保您的网络服务器通过 HTTPS 为控制台提供服务，因为 Cognito 要求回传网址使用 HTTPS。检查 Cognito 应用程序客户端回传 URL 是否包含您的 Web 服务器的域名。

联系 AWS Support

如果您有 [AWS Business Support+](#)、[AWS Enterprise Support](#) 或 [统一运营](#)，则可以使用 AWS Support Center 获取有关此解决方案的专家帮助。以下部分提供了说明。

创建工单

1. 登录 [支持中心](#)。
2. 选择创建工单。

我们可提供哪些帮助？

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. 对于严重性，选择与您的使用案例最匹配的选项。
5. 当您填写完服务、类别和严重性信息后，界面中会填入常见故障排除问题的链接。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。

附加信息

1. 对于主题，输入可概括您的问题的文本。
2. 对于描述，请详细描述问题，包括此产品的名称和您使用的版本，例如以下示例：AWS 上的分布式负载测试 vX.Y.Z.
3. 选择附加文件。
4. 附上 AWS Support 处理请求所需的信息。

帮助我们更快地处理您的工单

1. 输入请求的信息。
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. 查看立即解决中的解决方案。
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，输入请求的信息，然后选择提交。

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。要更新到最新版本，请根据您的原始部署方法参考相应的部分：[AWS Launch Wizard](#) 或 [AWS CloudFormation](#)。

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。

使用 AWS Launch Wizard 进行更新

控制台会在部署版本下拉列表中自动显示解决方案的最新可用版本。如果您之前部署过该解决方案，请按照以下步骤将您的部署更新到最新版本。

1. 转到 [Launch Wizard 部署](#)。
2. 选择要更新的部署。
3. 选择操作，然后选择更新部署版本。
4. 从可用的部署版本中选择最新版本。
5. 查看配置。
6. 对每个步骤进行必要的更改。
7. 确认更新。

使用 AWS 进行更新 CloudFormation

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。

1. 登录 [CloudFormation 控制台](#)，选择您的现有 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. 在审核页面上，审核并确认设置。
9. 选中确认模板可创建 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 3.3.0 或更高版本进行更新，请通过 AWS Launch Wizard 或 AWS 按照标准更新程序进行更新。CloudFormation](#)

1. 下载 [distributed-load-testing-on-aws.template](#)。
2. 打开模板并导航到Conditions:并查找DLTCommonResourcesAppRegistryCondition。
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true值更改为false：

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

5. 按照使用 [AWS 更新中的步骤，使用自定义模板更新](#)您的堆栈 CloudFormation。
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。

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。在您部署了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”执行以下操作：

- 从中央位置监控资源、跨堆栈和 AWS 账户部署的资源成本以及日志。
- 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。

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 (AWS CLI) Line CLI 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多个保留资源。如果您有数据要保留，AWS Solutions 不会自动删除存储和日志资源。因此，请参阅以下有关如何删除 S3 存储桶、DynamoDB 表 CloudWatch、日志组和控制面板的信息。CloudWatch

Important

在卸载解决方案之前，请确保已先删除所有区域堆栈。我们还建议在删除主堆栈之前通过 DLT Web 控制台删除所有测试场景。这样可以确保正确清理运行时创建的资源，例如 CloudWatch 仪表盘。

使用 AWS 管理控制台

AWS CloudFormation

1. 登录 [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。

AWS Launch Wizard

1. 登录 AWS Launch Wizard 控制台。
2. 在 [Launch Wizard 部署](#) 页面上，选择此解决方案的部署。
3. 选择操作，然后选择删除。
4. 确认删除操作。

使用 AWS 命令行界面

确定 AWS 命令行界面 (AWS 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，请运行以下命令。

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

手动删除保留的资源

删除堆栈后，请参阅以下章节，了解如何删除剩余资源。

删除 Amazon S3 存储桶

如果您决定删除 AWS CloudFormation 堆栈以防止意外丢失数据，则此解决方案配置为保留解决方案创建的 Amazon S3 存储桶。卸载解决方案后，如果您不需要保留数据，则可以手动删除这些 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. 在导航窗格中，选择存储桶。
3. 找到解决方案创建的 S3 存储桶。
4. 选择一个 S3 存储桶，然后选择清空。
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。

重复这些步骤，直到删除所有解决方案 S3 存储桶。

要使用 AWS CLI 删除 S3 存储桶，请对每个存储桶运行以下命令：

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

删除亚马逊 DynamoDB 表

如果您决定删除 AWS CloudFormation 堆栈以防止意外丢失数据，则此解决方案配置为保留解决方案创建的 DynamoDB 表。卸载解决方案后，如果您不需要保留数据，则可以手动删除这些表。按照以下步骤删除 DynamoDB 表。

1. 登录 [Amazon DynamoDB 控制台](#)。
2. 在导航窗格中，选择表。
3. 找到解决方案创建的表。
4. 选择一个表并选择“删除”。

5. 输入删除并选择删除表以确认删除。

重复这些步骤，直到删除所有解决方案表。

要使用 AWS CLI 删除 DynamoDB 表，请运行以下命令：

```
$ aws dynamodb delete-table --table-name <table-name>
```

删除 CloudWatch 日志组

如果您决定删除 AWS CloudFormation 堆栈，则此解决方案会保留 CloudWatch 日志组，以防止意外丢失数据。卸载解决方案后，如果您不需要保留数据，则可以手动删除日志组。按照以下步骤删除 CloudWatch 日志组。

1. 登录 [Amazon CloudWatch 控制台](#)。
2. 在导航窗格中，选择日志组。
3. 找到解决方案创建的日志组。
4. 选择要删除的日志组。
5. 选择操作，然后选择删除。

重复这些步骤，直到删除所有解决方案日志组。

要使用 AWS CLI 删除日志组，请运行以下命令：

```
$ aws logs describe-log-groups --log-group-name-prefix <stack-name> --query  
'logGroups[].logGroupName' --output text  
$ aws logs delete-log-group --log-group-name <log-group-name>
```

Note

如果您部署了区域堆栈，请在部署了区域堆栈的每个区域重复此过程。

删除 CloudWatch 仪表板

此解决方案在每次运行负载测试时都会创建一个 CloudWatch 仪表板。这些仪表板遵循命名模式 `EcsLoadTesting-<testId>-<region>`。当您通过 DLT 控制台删除测试时，它们会被清除，但是如果在先删除所有测试的情况下删除堆栈，它们将保留下来。

卸载解决方案后，如果这些仪表板尚未清理，则可以手动将其删除。按照以下步骤删除 CloudWatch 仪表板。

1. 登录 [Amazon CloudWatch 控制台](#)。
2. 在导航窗格中，选择控制面板。
3. 在筛选器字段中，输入 EcsLoadTesting 以查找解决方案的仪表板。
4. 选择要删除的仪表板。
5. 选择删除。

要使用 AWS CLI 删除控制面板，请运行以下命令：

```
$ aws cloudwatch list-dashboards --dashboard-name-prefix EcsLoadTesting --query
'DashboardEntries[].DashboardName' --output text
$ aws cloudwatch delete-dashboards --dashboard-names <dashboard-name-1> <dashboard-
name-2>
```

清理 ALB + ECS Fargate 部署后资源

如果您选择了 ALB + ECS Fargate 部署选项，请删除将您的自定义域映射到 DNS 提供商中的 ALB DNS 名称的 CNAME 或 Route 53 别名记录。如果不再需要 ACM 证书，也可以从 [ACM](#) 控制台中将其删除。删除 CloudFormation 堆栈后，部署在 ALB 前面的 AWS WAF Web ACL 会自动删除。

堆栈删除后恢复 DLT 数据

本节介绍如何在删除原始堆栈后将测试场景和运行历史恢复到新的 DLT 堆栈中。当客户的 S3 存储桶和 DynamoDB 表已保留（默认行为）并且客户希望将它们重新连接到新的部署时，它适用。

背景

删除 DLT 堆栈后，由于数据保护政策，以下资源将保留在客户的 AWS 账户中：

资源	包含	命名
场景 S3 存储桶	测试脚本 (JMeter/K6/Locust)、每个任务的结果 XML 文件、JMeter 框架资产	Auto-generated (例如, dltstack-scenarios-bucket-abc123)

资源	包含	命名
日志 S3 存储桶	所有 DLT 存储桶的 S3 访问日志	Auto-generated
控制台 S3 存储桶	Web 用户界面静态资产	Auto-generated
场景 DynamoDB 表	测试定义、配置、调度规则	Auto-generated (例如, DLTStack-Scenarios Table-xyz789)
历史记录 DynamoDB 表	测试运行结果、持续时间、成功率	Auto-generated (例如, DLTStack-HistoryTable-xyz789)
CloudWatch 日志组	Lambda 和 ECS 执行日志 (保留期为 10 年)	按函数命名

所有资源名称均由自动生成。CloudFormation 它们不可预测，也无法在部署新堆栈时指定为参数。

重要限制

1. DLT CloudFormation 模板不接受现有 DynamoDB 表或 S3 存储桶的参数。新的部署总是会创建新的资源。
2. 直接修改堆栈外的 CloudFormation-managed 资源会导致堆栈偏移。所有数据迁移都必须在数据级别进行，而不是在基础架构级别进行。
3. 解决方案 UUID 是在每次新部署时随机生成的。新堆栈的 UUID 将与原始堆栈不同。
4. 默认情况下，DynamoDB 表启用 Point-in-Time 了恢复 (PITR)，提供过去 35 天的连续备份。

恢复过程

步骤 1：确定保留的资源

在部署新堆栈之前，请从已删除的堆栈中找到保留的资源。

```
# List retained DynamoDB tables (look for DLT naming patterns)
```

```
aws dynamodb list-tables --query "TableNames[?contains(@, 'ScenariosTable') ||
contains(@, 'HistoryTable')]"

# List retained S3 buckets (look for DLT naming patterns)
aws s3api list-buckets --query "Buckets[?contains(Name, 'dltttestrunnerstoragedlts') ||
contains(Name, 'dltconsoleresourcesdlttda')]"
```

如果您在删除之前记下了 CloudFormation 堆栈输出，请使用这些确切的名称。相关产出是：

- ScenariosBucket— S3 存储桶名称
- ScenariosTable— DynamoDB 场景表名称

对于未作为堆栈输出公开的资源（例如历史表），`list-stack-resources`请在删除之前使用来获取物理资源 ID。此命令列出堆栈中的所有资源，无论其是否有相应的输出。

步骤 2：部署新的 DLT 堆栈

使用相同的 CloudFormation 模板版本（或更高版本）部署全新 DLT 堆栈。使用与原始部署相同的参数（VPC 设置、管理员电子邮件等）。

```
aws cloudformation create-stack \
  --stack-name distributed-load-testing-new \
  --template-url https://s3.amazonaws.com/solutions-reference/distributed-load-testing-
on-aws/latest/distributed-load-testing-on-aws.template \
  --parameters ParameterKey=AdminName,ParameterValue=<admin-name> \
  ParameterKey=AdminEmail,ParameterValue=<admin-email> \
  --capabilities CAPABILITY_IAM CAPABILITY_NAMED_IAM CAPABILITY_AUTO_EXPAND
```

等待堆栈达到CREATE_COMPLETE状态。

步骤 3：确定新堆栈的资源

从新堆栈的输出中检索资源名称。

```
aws cloudformation describe-stacks \
  --stack-name distributed-load-testing-new \
  --query "Stacks[0].Outputs"
```

请注意以下各项的值：

- ScenariosBucket（新的存储桶名称）

- ScenariosTable (新表名)

对于任何未作为堆栈输出公开的资源 (例如历史表) , 请使用 `list-stack-resources` 其逻辑 ID 进行查找 :

```
aws cloudformation list-stack-resources \  
  --stack-name distributed-load-testing-new \  
  --query "StackResourceSummaries[?contains(LogicalResourceId, 'HistoryTable')].  
{LogicalId:LogicalResourceId,PhysicalId:PhysicalResourceId}"
```

此命令列出了堆栈管理的所有资源。您可以按逻辑 ID 子字符串筛选以查找任何资源。

步骤 4 : 迁移 DynamoDB 数据

从保留的表中导出数据并将其导入到新表中。使用 DynamoDB 数据导出或扫描和写入操作。

选项 A : 使用 Dynamo Export/Import DB (建议用于大型数据集)

对于包含数千个项目的表, 使用 DynamoDB 导出到 S3, 然后再进行导入 :

```
# Export from retained table to S3  
aws dynamodb export-table-to-point-in-time \  
  --table-arn arn:aws:dynamodb:<region>:<account>:table/<old-scenarios-table> \  
  --s3-bucket <temporary-export-bucket> \  
  --s3-prefix dlt-migration/scenarios/ \  
  --export-format DYNAMODB_JSON  
  
# Wait for export to complete, then import into new table  
aws dynamodb import-table \  
  --s3-bucket-source S3Bucket=<temporary-export-bucket>,S3KeyPrefix=dlt-migration/  
scenarios/ \  
  --table-creation-parameters '{"TableName": "<new-scenarios-table>", "KeySchema":  
[{"AttributeName": "testId", "KeyType": "HASH"}], "AttributeDefinitions":  
[{"AttributeName": "testId", "AttributeType": "S"}], "BillingMode": "PAY_PER_REQUEST"}' \  
  --input-format DYNAMODB_JSON
```

Note

`import-table` API 会创建一个新表。由于新堆栈已经创建了目标表, 因此请改用选项 B, 或者先删除新的空表, 然后通过导入重新创建它 (这将导致堆栈偏移, 请参阅下面的警告)。

选项 B：扫描和 BatchWrite（推荐）

这种方法直接从保留的表中读取并写入新堆栈的表，而不会导致堆栈偏移。它需要 Python 3 和 boto3 库。

```
# Install boto3 if not already available
pip install boto3
```

```
#!/usr/bin/env python3
"""Migrate DynamoDB items directly from retained tables to new stack tables."""

import boto3

dynamodb = boto3.resource("dynamodb", region_name="<region>")

def migrate_table(source_table_name, target_table_name):
    source = dynamodb.Table(source_table_name)
    target = dynamodb.Table(target_table_name)

    # Scan all items from the retained table (handles pagination automatically)
    items = []
    response = source.scan()
    items.extend(response["Items"])
    while "LastEvaluatedKey" in response:
        response = source.scan(ExclusiveStartKey=response["LastEvaluatedKey"])
        items.extend(response["Items"])

    print(f"Scanned {len(items)} items from {source_table_name}")

    # Write items to the new table using batch_writer (handles batching automatically)
    with target.batch_writer() as batch:
        for item in items:
            batch.put_item(Item=item)

    print(f"Wrote {len(items)} items to {target_table_name}")

# Migrate scenarios
migrate_table("<old-scenarios-table>", "<new-scenarios-table>")

# Migrate history
migrate_table("<old-history-table>", "<new-history-table>")
```

对于大型表（成千上万个项目），可以考虑使用并行扫描段来加快读取阶段。

选项 C : 从 PITR 恢复 (如果在 35 天的时间范围内)

如果在过去 35 天内删除了堆栈，并且启用了 PITR (默认情况下)，则可以将表恢复到某个时间点。但是，PITR 还原会创建一个名称不同的新表，因此您仍然需要使用选项 B 将数据复制到堆栈管理的表中。

```
# Restore to a new temporary table
aws dynamodb restore-table-to-point-in-time \
  --source-table-name <old-scenarios-table> \
  --target-table-name dlt-scenarios-restored \
  --restore-date-time <timestamp-before-deletion>

# Then scan from dlt-scenarios-restored and batch-write into the new stack's table
```

步骤 5 : 迁移 S3 数据

将测试脚本和结果文件从保留的存储桶复制到新堆栈的存储桶。

```
# Copy all objects from the old scenarios bucket to the new one
aws s3 sync \
  s3://<old-scenarios-bucket> \
  s3://<new-scenarios-bucket> \
  --source-region <region> \
  --region <region>
```

这复制了：

- 测试脚本 (JMeter .jmx 文件、K6 脚本、Locust 文件、ZIP 存档)
- 按测试 ID 和区域组织的测试结果 XML 文件
- JMeter 框架资产和插件

步骤 6 : 验证迁移

1. 使用新堆栈的 URL (可在堆WebConsole栈输出中找到) 登录 DLT Web 控制台。
2. 确认所有测试场景都显示在场景列表中。
3. 打开各个场景并验证测试运行历史记录是否可见。
4. 确认可以从场景详细信息页面下载测试脚本。
5. 运行一个小型测试，验证新堆栈的端到端功能。

步骤 7：清理保留的资源

确认所有数据均已成功迁移后，删除保留的旧资源以避免持续的存储成本。

```
# Delete old DynamoDB tables
aws dynamodb delete-table --table-name <old-scenarios-table>
aws dynamodb delete-table --table-name <old-history-table>

# Empty and delete old S3 buckets
aws s3 rm s3://<old-scenarios-bucket> --recursive
aws s3api delete-bucket --bucket <old-scenarios-bucket>

aws s3 rm s3://<old-logs-bucket> --recursive
aws s3api delete-bucket --bucket <old-logs-bucket>

aws s3 rm s3://<old-console-bucket> --recursive
aws s3api delete-bucket --bucket <old-console-bucket>
```

只有在确认新堆栈已完全运行且所有历史数据完好无损后才执行此步骤。

避免堆栈漂移

当资源的实际状态与 CloudFormation 预期的状态不同时，就会发生堆栈偏移。为避免在此恢复过程中出现偏差，请执行以下操作：

- 请勿直接重命名或修改新堆栈的 DynamoDB 表或 S3 存储桶。
- 请勿使用 `import-table` 来替换新堆栈的表（这需要先删除 CloudFormation-managed 表）。
- 请勿在之外修改 IAM 策略、存储桶策略或表设置 CloudFormation。
- 请仅使用数据级操作：PutItem、BatchWriteItems、sync、s3 cp。
- 请务必将数据写入 CloudFormation 创建的表和存储桶。向 DynamoDB 表添加项目或向 S3 存储桶添加对象并不构成偏差。

将数据写入 CloudFormation-managed 表和存储桶是安全的，因为会 CloudFormation 跟踪资源配置（表架构、计费模式、加密设置），而不是数据内容。

无法恢复的东西

删除堆栈后，以下内容将丢失，且无法通过此过程恢复：

Item	Reason
Cognito 用户帐户	用户池随堆栈一起删除。用户必须重新注册。
活动 EventBridge 日程安排	计划测试规则已删除。在 DLT 用户界面中手动重新创建计划。
CloudWatch 仪表板	Per-test 仪表板 (EcsLoadTesting*) 已删除。新的运行将创建新的仪表板。
解决方案 UUID	将生成一个新的随机 UUID。这仅影响运营指标的相关性。
区域堆栈协会	必须重新部署区域堆栈并将其与新的中心堆栈重新关联。

预防措施

要简化 future 恢复方案，请执行以下操作：

1. 在任何删除之前记录堆栈输出。保存 ScenariosBucketScenariosTable、和历史表的名称。
2. 启用 DynamoDB PITR (在 DLT 中默认启用)。验证它是否保持活动状态。
3. 启用 S3 版本控制 (默认情况下在场景存储桶上启用)。这样可以防止意外删除对象。
4. 考虑在场景和历史表上启用 DynamoDB 删除保护，以此作为额外的保护措施：

```
aws dynamodb update-table \
  --table-name <scenarios-table> \
  --deletion-protection-enabled
```

5. 使用 AWS Backup 创建 DynamoDB 表的定时备份，以便在 35 天 PITR 窗口之后长期保留。

开发人员指南

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 (CVE)。如果可能，团队会发布带有相同版本标签的修补镜像，以便在不破坏与已发布解决方案版本的兼容性的前提下解析 CVE。

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

负载测试仪图片

此解决方案使用由 AWS 管理的公用 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，或者编辑此容器以满足您的需求。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 Amazon 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/ecslister.py
RUN chmod 755 /bzt-configs/ecscroller.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

#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 `ecslister.py` 脚本，而领导任务将运行 `ecscontroller.py` 脚本。该 `ecslister.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。

Web 控制台镜像 (仅限 ALB + ECS Fargate 模板)

ALB + ECS Fargate 模板在 ECS Fargate 上将 Web 控制台作为容器运行。默认情况下，该解决方案会从 AWS 管理的公共 Amazon ECR 存储库中提取 Web 控制台映像。

在限制访问公共容器注册表的环境中 (例如，无法访问互联网的 VPC 或拥有 ECR 公共访问策略的账户)，您必须将公共映像镜像到私有 Amazon ECR 存储库，并在 Web 控制台映像 URI 参数中提供私有映像 URI。CloudFormation 您也可以使用这种方法自定义 Web 控制台映像以满足您的需求。

将公共映像镜像到私有 ECR 存储库

要将 Web 控制台映像镜像到私有 ECR 存储库，请执行以下操作：

1. 向公共 ECR 注册表进行身份验证：

```
$ aws ecr-public get-login-password --region us-east-1 | docker login --username AWS --password-stdin public.ecr.aws
```

2. 提取公共 Web 控制台镜像：

```
$ docker pull public.ecr.aws/aws-solutions/distributed-load-testing-on-aws-web-console:<version>
```

3. 在您的账户中创建私有 ECR 存储库（如果尚不存在）：

```
$ aws ecr create-repository --repository-name <your-repo-name> --region <region> 2>/dev/null || true
```

4. 向您的私有 ECR 注册表进行身份验证：

```
$ aws ecr get-login-password --region <region> | docker login --username AWS --password-stdin <account-id>.dkr.ecr.<region>.amazonaws.com
```

5. 标记图像并将其推送到您的私有存储库：

```
$ docker tag public.ecr.aws/aws-solutions/distributed-load-testing-on-aws-web-console:<version> \
  <account-id>.dkr.ecr.<region>.amazonaws.com/<your-repo-name>:<version>
$ docker push <account-id>.dkr.ecr.<region>.amazonaws.com/<your-repo-name>:<version>
```

6. 启动 ALB + ECS Fargate 堆栈时，在 Web 控制台镜像 URI 参数中输入私有镜像 URI：

```
<account-id>.dkr.ecr.<region>.amazonaws.com/<your-repo-name>:<version>
```

自定义 Web 控制台镜像

您也可以自定义 Web 控制台镜像。图像资源位于[GitHub 存储库](#)中的 `deployment/ecr/distributed-load-testing-on-aws-web-console` 目录中。编辑容器文件以满足您的需求，在本地构建映像，将其推送到您的私有 ECR 存储库，然后在 Web 控制台映像 URI 参数中提供 URI。

Web 控制台容器使用 Nginx 为静态 Web 应用程序提供服务。启动时，入口点脚本会从 Amazon S3 下载 Web 控制台资产，并将其提取到 Nginx 文档根目录中。

以下示例显示了容器文件。

```
FROM public.ecr.aws/nginx/nginx:alpine

# Install AWS CLI for S3 operations
RUN apk upgrade --no-cache zlib libpng && \
    apk add --no-cache aws-cli

# Copy nginx configuration
COPY nginx.conf /etc/nginx/nginx.conf

# Copy entrypoint script
COPY entrypoint.sh /usr/local/bin/entrypoint.sh
RUN chmod +x /usr/local/bin/entrypoint.sh

EXPOSE 80

ENTRYPOINT ["/usr/local/bin/entrypoint.sh"]
```

入口点脚本从 S3 下载 Web 控制台资产并启动 Nginx。

```
#!/bin/sh
set -e

S3_URI="s3://${S3_BUCKET}/${S3_KEY}"

echo "[$(date -Iseconds)] Downloading from $S3_URI"

# AWS CLI has built-in retry with exponential backoff (standard mode)
AWS_MAX_ATTEMPTS=5 aws s3 cp "$S3_URI" /tmp/web-app.zip

echo "[$(date -Iseconds)] Download successful"

# Extract and cleanup
unzip -o /tmp/web-app.zip -d /usr/share/nginx/html/
rm -f /tmp/web-app.zip

# Start Nginx
exec nginx -g 'daemon off;'
```

Nginx 配置为单页应用程序 (SPA) 提供对 ALB 的运行状况检查支持、gzip 压缩、静态资产缓存和安全标头。

```
events {
```

```
worker_connections 1024;
}

http {
    include /etc/nginx/mime.types;
    default_type application/octet-stream;

    gzip on;
    gzip_types text/plain text/css application/json application/javascript text/xml
    application/xml;

    server {
        listen 80;
        root /usr/share/nginx/html;
        index index.html;

        # Health check endpoint for ALB
        location /healthz {
            return 200 'OK';
        }

        # SPA routing
        location / {
            try_files $uri $uri/ /index.html;
        }

        # Cache static assets for 1 year
        location ~* \.(js|css|png|jpg|jpeg|gif|ico|svg|woff|woff2)$ {
            expires 1y;
            add_header Cache-Control "public, immutable";
        }

        # No cache for index.html (SPA entry point)
        location = /index.html {
            add_header Cache-Control "no-cache, no-store, must-revalidate";
        }

        # No cache for runtime config
        location = /aws-exports.json {
            add_header Cache-Control "no-store, no-cache, must-revalidate";
        }

        # Security headers
        add_header X-Frame-Options "SAMEORIGIN" always;
```

```
    add_header X-Content-Type-Options "nosniff" always;
    add_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always;
}
}
```

有关更多信息，请参阅《[亚马逊 ECR 用户指南](#)》中的“[将 Docker 镜像推送到亚马逊 ECR 私有存储库](#)”。

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 API 来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/ {test} RunId](#)
- [删除 /scenarios/ {testID} /testruns/ {test} RunId](#)

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

说明

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	说明
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

说明

该GET /scenarios操作允许您检索测试场景列表。

响应

Name	说明
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

说明

该POST /scenarios操作允许您创建或安排测试场景。

请求正文

Name	说明
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"]）
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	说明
testId	测试的唯一 ID
testName	测试的名称
status	测试的状态

选项/场景

说明

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

获取 /scenarios/ {testID}

说明

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

是否必需：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查询参数的查询参数。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	说明
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	上次测试结束的时间和日期

Name	说明
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	用于运行测试的任务 ID 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 <code>history=false</code>)
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
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}

说明

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

是否必需：是

响应

Name	说明
status	测试的状态

删除 /scenarios/ {testID}

说明

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

是否必需：是

响应

Name	说明
status	测试的状态

选项 /scenarios/ {testID}

说明

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	说明
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
startTime	上次测试开始的时间和日期
endTime	上次测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	用于运行测试的任务 ID 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

说明

该 GET /scenarios/{testId}/testruns 操作会检索特定测试场景的测试运行 ID，可以选择按时间范围进行筛选。当 latest=true 时，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
- 是否必需：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 (含)。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	说明
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）
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=eyJ0ZXN0SWQiOiJzZVFVeTEyTETtMIiwic3RhcnRUaW1lIjo1MjAyNC0wMS

GET /scenarios/ {testID} /testruns/ {test} RunId

说明

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
- 是否必需：是

testRunId

- 具体的测试运行 ID
- 类型：字符串
- 是否必需：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	说明
testId	测试的唯一 ID (例如 , seQUy12LKL)
testRunId	特定的测试运行 ID (例如 , 2DEwHIItEne)

Name	说明
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)
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-无效的 testID 或测试 RunId
- 404-未找到测试运行
- 500-服务器内部错误

删除 /scenarios/ {testID} /testruns/ {test} RunId

说明

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

是否必需：是

testRunId

- 要删除的特定测试运行 ID

类型：字符串

是否必需：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-无效的 testID 或测试 RunId
- 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

说明

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

是否必需：是

data

- 如果返回完整的基线测试运行数据true，否则只返回测试 RunId

类型：布尔值

默认值：false

必需：否

响应

200-成功

何时data=false (默认) :

Name	说明
testId	测试场景 ID (例如 , seQUy12LKL)

Name	说明
baselineTestRunId	基准测试运行 ID (例如 , 2DEwHIItEne)

什么时候data=true :

Name	说明
testId	测试场景 ID (例如 , seQUy12LKL)
baselineTestRunId	基准测试运行 ID (例如 , 2DEwHIItEne)
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

说明

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

是否必需 : 是

请求正文

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

响应

200-成功

Name	说明
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, 2DEwHIte)

错误响应

- 400-无效的 testID 或测试 RunId
- 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

说明

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线值。

请求参数

testId

- 测试场景 ID

类型：字符串

是否必需：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

说明

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	说明
tasks	用于运行测试的任务 ID 列表

选项 /任务

说明

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	说明
taskIds	用于运行测试的任务 ID 列表

获取 /区域

说明

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	说明
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

说明

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	说明
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

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 Hard/Soft 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>`-DLTTestRunnerStorageDLTScenariosTable-<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 TaskDefinition 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

说明

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。

端点

GET /scenarios

参数

无

响应

Name	说明
testId	测试场景的唯一标识符

Name	说明
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	测试场景描述

获取场景详情

说明

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。

端点

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

是否必需：是

响应

Name	说明
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

Name	说明
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

说明

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

端点

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

或者

```
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

是否必需：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	说明
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

说明

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。

端点

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

是否必需：是

test_run_id

- 特定测试运行的唯一标识符

类型：字符串

是否必需：是

响应

Name	说明
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、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

说明

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

端点

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

是否必需：是

响应

Name	说明
results	与格式相同的最新测试运行数据 <code>get_test_run</code>

get_baseline_test_run

说明

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。

端点

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

是否必需：是

响应

Name	说明
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

说明

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。

端点

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

是否必需：是

test_run_id

- 特定测试运行的唯一标识符

类型：字符串

是否必需：是

响应

Name	说明
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。无需修改底层 API 即可支持 MCP 功能。

DLT CLI

DLT CLI 允许您直接从终端与 AWS REST API 上的分布式负载测试进行交互。它支持负载测试的脚本自动化，允许您列出、检查和启动测试场景，并作为可重复工作流程的一部分查询运行结果。

CLI 的目标是与 DLT Web 控制台相同的 IAM-authorized API Gateway，因此基于浏览器的界面中可用的任何操作也可以使用相同的底层 API 和权限模型从命令行执行。

CLI 源代码和完整文档位于解决方案 GitHub 存储库的 `source/cli` 目录下。要找到它，请导航到 [AWS 存储库上的分布式负载测试](#) 并打开 `source/cli`。该目录 `README.md` 中的是安装、配置、身份验证和完整命令参考的权威参考。

关键功能

命令组	说明
<code>configure</code>	使用您的 DLT 堆栈设置以交互方式或通过导入文件来配置 CLI <code>aws-exports.json</code> 。
<code>login</code>	使用 DLT 服务进行身份验证并获取调用 API 所需的凭据。
<code>logout</code>	从本地计算机中移除存储的凭据。
<code>token</code>	检查并输出身份验证令牌和凭证到期状态。
<code>scenarios</code>	列出、查看详细信息并启动（重新运行）现有的测试场景。
<code>runs</code>	查询测试运行结果、查看运行详细信息、与基准进行比较以及下载运行构件。

CLI 支持三种身份验证模式：基于浏览器的登录（通过 Cognito 托管用户界面的 PKCE 流程）、无头 SRP（安全远程密码）身份验证和直接 IAM 凭证身份验证。有关每种模式的完整设置详细信息，`README.md` 请参阅 [GitHub 存储库 `source/cli` 目录](#) 中的。

安装和详细参考

CLI 以可移植 Node.js 包的形式分发，可以直接运行，也可以使用存储库从源代码构建。要开始使用，请导航到 [AWS 存储库上的分布式负载测试](#) GitHub 并打开该source/cli目录。该目录中的README.md文件包含：

- 安装说明（便携式捆绑包和源代码构建）
- 所有三种身份验证模式的配置和身份验证设置
- 包含用法示例的完整命令参考

快速入门

Note

本快速入门假设 DLT 堆栈已部署。

选项 1：便携式捆绑包（推荐）

最快的入门方法是便携式捆绑包。这不需要克隆存储库或任何构建步骤：

```
# Download the portable bundle
curl -sLo /usr/local/bin/dlt \
  https://raw.githubusercontent.com/aws-solutions/distributed-load-testing-on-aws/main/
  deployment/cli/dlt-cli.mjs

# Make it executable
chmod +x /usr/local/bin/dlt

# Point the CLI at your deployed DLT stack
dlt configure

# Authenticate with the DLT service
dlt login

# Run a command (for example, list test scenarios)
dlt scenarios list
```

选项 2：从源代码构建

如果您更喜欢从源代码构建，请先克隆[存储库](#)。该项目使用 npm 工作空间，因此请从存储库根目录运行以下命令：

```
# Install all workspace dependencies
npm install

# Build the CLI
npm run build -w source/cli

# Link the dlt command into your PATH
npm link -w source/cli

# Point the CLI at your deployed DLT stack
dlt configure

# Authenticate with the DLT service
dlt login

# Run a command (for example, list test scenarios)
dlt scenarios list
```

Cron 表达式引用

此解决方案使用标准 5 字段 Linux cron 格式的子集来制定重复的测试计划。cron 表达式由五个以空格分隔的字段组成。

```
##### minute (0-59)
# ##### hour (0-23, *, */N, or comma list)
# # ##### day of month (1-31 or *)
# # # ##### month (1-12 or *)
# # # # ## day of week (0-6, *, range, or list)
# # # # #
0 9 * * 1-5
```

接受的值

下表描述了每个字段接受的内容。

字段	接受的值	示例
分钟	从 0 到 59 之间的单个值。	0, 30, 45
小时	(每小时)、步长值、0 到 23 之间的单个值或以逗号分隔的值列表。	,, 9, 9, 17
日期	* (每天) 或从 1 到 31 的单个值。	*, 1, 15, 31
Month	* (每个月) 或从 1 到 12 的单个值。	*, 1, 6, 12
星期几	* (每天)、0 到 6 之间的单个值、使用连字符的范围或逗号分隔的列表。	*, 0, 1-5, 0, 6

星期值使用以下映射：0 = 星期日，1 = 星期一，2 = 星期二，3 = 星期三，4 = 星期四，5 = 星期五，6 = 星期六。

不支持的模式

以下模式是有效的 Linux cron 语法，但此解决方案不支持。

模式	示例	Reason
分钟步长值	*/15 * * * *	最小调度间隔为一小时。
会议纪要清单	0,30 * * * *	最小调度间隔为一小时。
月份的日期范围	0 9 1-15 * *	仅接受单日值或通配符。
月份的日期清单	0 9 1,15 * *	仅接受单日值或通配符。
月份范围	0 9 * 3-9 *	仅接受单月值或通配符。
月份清单	0 9 1 1,6,12 *	仅接受单月值或通配符。
问号	0 9 ? * ?	问号字符在标准 Linux cron 中无效。请改用 *。

预设图案

Web 控制台提供以下预设模式，您可以直接选择这些模式。

模式名称	Expression	说明
每小时	0 * * * *	每小时在第 0 分钟运行。
每天上午 9:00	0 9 * * *	在所选时区每天上午 9:00 运行一次。
工作日上午 8:00	0 8 * * 1-5	周一至周五上午 8:00 运行。
每周日下午 5 点	0 17 * * 0	每周日下午 5:00 运行。
每月 1 日上午 11 点	0 11 1 * *	在每个月的第一天上午 11:00 运行。

调度限制

- 预定测试运行之间的最小间隔为一小时。系统会验证连续运行之间的间隔是否长于估计的测试持续时间。
- 所有定期计划都需要一个到期日。测试将在到期日期 (UTC) 结束后运行。
- 计划时区决定了 cron 何时触发。夏令时过渡是自动处理的。如果由于夏令时向前弹跳过渡而导致不存在预定时间，则会跳过该事件。
- 系统会在 Web 控制台和 API 上验证 cron 表达式。如果表达式与接受的格式不匹配，则无法创建测试。

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
- Beomseok Lee
- 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
- 布拉德·洪
- Colin McCoy

- 钟大卫
- 张佳丽

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

客户有责任对本文档中的信息进行单独评测。本文档：(a) 仅供参考，(b) 代表 AWS 当前的产品和实践，如有更改，恕不另行通知，以及 (c) AWS 及其关联公司、供应商或许可方未做出任何承诺或保证。AWS 产品或服务“按原样”提供，不附带任何形式的担保、陈述或条件，无论是明示还是暗示。AWS 对其客户的责任和责任由 AWS 协议控制，本文档不是 AWS 与其客户之间任何协议的一部分，也未对其进行修改。

AWS 上的分布式负载测试是根据 Apache [软件基金会提供的 Apache 许可版本 2.0 的](#)条款进行许可的。

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。