Guide de mise en œuvre

Tests de charge distribués sur AWS

Tests de charge distribués sur AWS: Guide de mise en œuvre

Table of Contents

Automatisez les tests de vos applications logicielles à grande échelle

Date de publication : novembre 2025

Les tests de charge distribués sur AWS vous aident à automatiser les tests de performance de vos applications logicielles à grande échelle afin d'identifier les goulots d'étranglement avant de publier votre application. Cette solution simule des milliers d'utilisateurs connectés qui génèrent des requêtes HTTP à un rythme soutenu sans qu'il soit nécessaire de configurer des serveurs.

Cette solution s'appuie sur Amazon Elastic Container Service (Amazon ECS) sur AWS Fargate pour déployer des conteneurs qui exécutent vos simulations de tests de charge et offre les fonctionnalités suivantes :

- Déployez Amazon ECS sur des conteneurs Fargate AWS qui s'exécutent indépendamment pour tester la capacité de charge de votre application.
- Simulez des dizaines de milliers d'utilisateurs simultanés dans plusieurs régions AWS en générant des demandes à un rythme soutenu.
- Personnalisez les tests de vos applications à l'aide JMeterde K6, de scripts de test Locust ou d'une simple configuration de point de terminaison HTTP.
- Planifiez les tests de charge pour qu'ils s'exécutent immédiatement, à une date et à une heure futures, ou selon un calendrier récurrent.
- Exécutez plusieurs tests de charge simultanément dans différents scénarios et régions.

Ce guide de mise en œuvre fournit une vue d'ensemble de la solution de test de charge distribué sur AWS, de son architecture de référence et de ses composants, des considérations relatives à la planification du déploiement et des étapes de configuration pour le déploiement de la solution sur le cloud Amazon Web Services (AWS). Il inclut des liens vers un CloudFormation modèle AWS qui lance et configure les services AWS nécessaires au déploiement de cette solution en utilisant les meilleures pratiques d'AWS en matière de sécurité et de disponibilité.

Le public visé par l'utilisation des fonctionnalités et des capacités de cette solution dans leur environnement comprend les architectes d'infrastructure informatique, les administrateurs et les DevOps professionnels ayant une expérience pratique de l'architecture dans le cloud AWS.

Utilisez ce tableau de navigation pour trouver rapidement les réponses aux questions suivantes :

Si tu veux...	Lisez.
Connaissez le coût de fonctionnement de cette solution. Le coût d'exécution de cette solution dans la région de l'est des États-Unis (Virginie du Nord) est estimé à 30,90 dollars américains par mois pour les ressources AWS.	Coût
Comprenez les considérations de sécurité liées à cette solution.	Sécurité
Sachez comment planifier les quotas pour cette solution.	Quotas
Découvrez quelles régions AWS prennent en charge cette solution.	Régions AWS prises en charge
Découvrez le serveur MCP en option pour l'analyse des tests de charge assistée par l'IA.	Intégration au serveur MCP
Consultez ou téléchargez le CloudFormation modèle AWS inclus dans cette solution pour déployer automatiquement les ressources d'infrastructure (la « pile ») de cette solution.	CloudFormation Modèle AWS
Accédez au code source et utilisez éventuellement l'AWS Cloud Development Kit (AWS CDK) pour déployer la solution.	GitHub référentiel

Fonctionnalités

La solution fournit les fonctionnalités suivantes :

Support de plusieurs frameworks de test

Supporte JMeter les scripts de test K6 et Locust, ainsi que les tests simples des points de terminaison HTTP sans nécessiter de scripts personnalisés. Pour plus d'informations, reportez-vous à la section Types de tests dans la section Détails de l'architecture.

Simulation d'une charge utilisateur élevée

Simule des dizaines de milliers d'utilisateurs virtuels simultanés pour tester le stress de votre application dans des conditions de charge réalistes.

Répartition de la charge entre plusieurs régions

Distribue des tests de charge dans plusieurs régions AWS afin de simuler le trafic utilisateur géographiquement distribué et d'évaluer les performances globales.

Planification flexible des tests

Planifie les tests pour qu'ils s'exécutent immédiatement, à une date et à une heure futures spécifiques, ou selon un calendrier récurrent à l'aide d'expressions cron pour les tests de régression automatisés.

Surveillance en temps réel

Fournit un streaming de données en direct en option pour suivre la progression des tests à l'aide de mesures en temps réel, notamment les temps de réponse, le nombre d'utilisateurs virtuels et les taux de réussite des demandes.

Résultats de tests complets

Affiche les résultats de test détaillés avec des mesures de performance, des percentiles (p50, p90, p95, p99), une analyse des erreurs et des artefacts téléchargeables pour une analyse hors ligne.

Comparaison de référence

Désigne les essais de référence pour la comparaison des performances afin de suivre les améliorations ou les régressions au fil du temps.

Flexibilité des terminaux

Teste n'importe quel point de terminaison HTTP ou HTTPS dans les régions AWS, les environnements sur site ou d'autres fournisseurs de cloud.

Console Web intuitive

Fournit une console Web pour créer, gérer et surveiller des tests sans interaction avec la ligne de commande.

Analyse assistée par IA (en option)

S'intègre aux outils de développement d'IA via le serveur MCP (Model Context Protocol) pour une analyse intelligente des données de test de charge.

Support de plusieurs protocoles

Supporte divers protocoles, notamment HTTP, HTTPS WebSocket, JDBC, JMS, FTP et gRPC via des scripts de test personnalisés.

Avantages

La solution offre les avantages suivants :

Tests de performance complets

Prend en charge les tests de charge, les tests de stress et les tests d'endurance pour évaluer de manière approfondie les performances des applications dans diverses conditions.

Détection précoce des problèmes

Identifie les goulots d'étranglement liés aux performances, les fuites de mémoire et les problèmes d'évolutivité avant le déploiement en production, réduisant ainsi le risque de pannes.

Simulation d'utilisation dans le monde réel

Simule avec précision le comportement réel des utilisateurs et les modèles de trafic pour valider les performances des applications dans des conditions réalistes.

Informations exploitables sur les Performances

Fournit des mesures détaillées, des percentiles et une analyse des erreurs pour comprendre le comportement des applications et guider les efforts d'optimisation.

Flux de travail de test automatisés

Permet des tests planifiés et récurrents pour une surveillance continue des performances et des tests de régression sans intervention manuelle.

Infrastructure rentable

Utilise des conteneurs AWS Fargate sans serveur assortis de tarifs, éliminant ainsi le besoin d' pay-per-useune infrastructure de test dédiée et de frais d'abonnement permanents.

Déploiement rapide des tests

Déploie et fait évoluer l'infrastructure de test en quelques minutes sans provisionner ni gérer de serveurs.

Interrogation aisée des résultats des tests

S'intègre aux outils de développement d'intelligence artificielle via un serveur MCP (Model Context Protocol) en option, permettant des requêtes en langage naturel et une analyse intelligente des données de test de charge pour des informations et un dépannage plus rapides.

Cas d’utilisation

Validation de pré-production

Testez les applications Web et mobiles dans des conditions de charge similaires à celles de la production avant de lancer une nouvelle version afin de valider les performances et d'identifier les problèmes.

Planification des capacités

Déterminez le nombre maximum d'utilisateurs simultanés que votre application peut prendre en charge avec l'infrastructure actuelle et déterminez quand une mise à l'échelle est requise.

Vérification de la charge maximale

Vérifiez que votre infrastructure peut gérer les pics de charge, les pics de trafic saisonniers ou les pics de demande inattendus sans dégradation des performances.

Optimisation des performances

Identifiez les obstacles aux performances tels que la lenteur des requêtes de base de données, l'exécution inefficace du code, la latence du réseau ou les contraintes de ressources.

Tests de régression

Planifiez des tests de charge récurrents pour détecter les régressions de performances induites par les nouveaux déploiements de code ou les modifications de l'infrastructure.

Évaluation globale des performances

Évaluez les performances des applications dans plusieurs régions géographiques afin de garantir une expérience utilisateur cohérente à un public mondial.

Test de charge des API

Testez les points de terminaison REST APIs, GraphQL ou les microservices pour valider les temps de réponse, le débit et les taux d'erreur sous charge.

Intégration du pipeline CI/CD

Intégrez des tests de performance automatisés dans les pipelines d'intégration et de déploiement continus pour détecter les problèmes de performance dès le début du cycle de développement.

Tests de services tiers

Testez les performances et la fiabilité de tiers APIs ou de services dont dépend votre application dans différentes conditions de charge.

Concepts et définitions

Cette section décrit les concepts clés et définit la terminologie spécifique à cette solution :

scénario

Définition du test, y compris le nom, la description, le nombre de tâches, la simultanéité, la région AWS, la montée en puissance, le maintien, le type de test, la date de planification et les configurations de récurrence.

nombre de tâches

Nombre de conteneurs qui seront lancés dans le cluster Fargate pour exécuter le scénario de test. Aucune tâche supplémentaire ne sera créée une fois que la limite du compte pour les ressources Fargate aura été atteinte. Toutefois, les tâches déjà en cours se poursuivront.

concurrency

La simultanéité (nombre d'utilisateurs virtuels simultanés par tâche). La simultanéité recommandée sur la base des paramètres par défaut est de 200. La simultanéité est limitée par le processeur et la mémoire. Pour les tests basés sur Apache JMeter, une simultanéité plus élevée augmente la mémoire utilisée par la JVM sur la tâche ECS. La définition de tâche ECS par défaut crée des tâches avec 4 Go de mémoire. Il est recommandé de commencer par des valeurs de simultanéité plus faibles pour une tâche et de surveiller les CloudWatch métriques ECS pour le cluster de tâches. Reportez-vous aux métriques d'utilisation du cluster Amazon ECS.

montée en puissance

Période pendant laquelle il faut passer progressivement de zéro au niveau de simultanéité cible.

maintenez pour

Période nécessaire pour maintenir le niveau de simultanéité cible une fois la montée en puissance terminée.

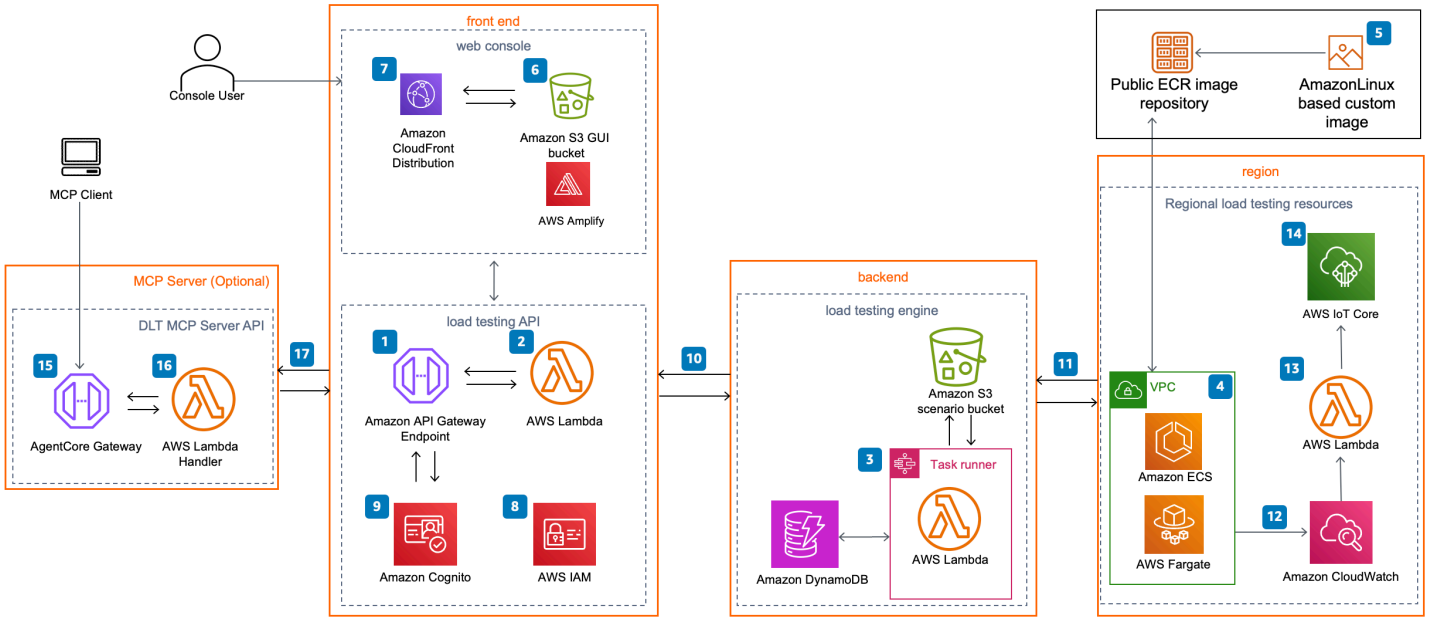
Pour une référence générale des termes AWS, consultez le glossaire AWS.

Présentation de l’architecture

Diagramme d’architecture

Le déploiement de cette solution avec les paramètres par défaut déploie les composants suivants dans votre compte AWS.

Tests de charge distribués sur l'architecture AWS sur AWS



> **Note**
>
> Les CloudFormation ressources AWS sont créées à partir des constructions du kit AWS Cloud Development Kit (AWS CDK).

Le flux de processus de haut niveau pour les composants de solution déployés avec le CloudFormation modèle AWS est le suivant :

1. Une API de testeur de charge distribué utilise Amazon API Gateway pour appeler les microservices de la solution (fonctions AWS Lambda).
2. Les microservices fournissent la logique métier permettant de gérer les données de test et d'exécuter les tests.

3. Ces microservices interagissent avec Amazon Simple Storage Service (Amazon S3), Amazon DynamoDB et AWS Step Functions pour stocker les détails et les résultats des scénarios de test et pour orchestrer l'exécution des tests.
4. Une topologie de réseau Amazon Virtual Private Cloud (Amazon VPC) est déployée et contient les conteneurs Amazon Elastic Container Service (Amazon ECS) de la solution exécutés sur AWS Fargate.
5. Les conteneurs utilisent une image de base Amazon Linux 2023 sur laquelle le framework de test de charge Taurus est installé. Taurus est un framework d'automatisation des tests open source qui prend en charge K6 JMeter, Locust et d'autres outils de test. L'image du conteneur est conforme à l'Open Container Initiative (OCI) et est hébergée par AWS dans un référentiel public Amazon Elastic Container Registry (Amazon ECR). Pour plus d'informations, reportez-vous à la section Personnalisation de l'image du conteneur.
6. Une console Web alimentée par AWS Amplify se déploie dans un compartiment S3 configuré pour l'hébergement Web statique.
7. Amazon CloudFront fournit un accès public sécurisé au contenu du bucket du site Web de la solution.
8. Lors de la configuration initiale, la solution crée un rôle d'administrateur par défaut (rôle IAM) et envoie une invitation d'accès à une adresse e-mail utilisateur spécifiée par le client.
9. Un groupe d'utilisateurs Amazon Cognito gère l'accès des utilisateurs à la console, à l'API du testeur de charge distribué et au serveur MCP.
10. Après avoir déployé cette solution, vous pouvez utiliser la console Web ou APIs créer et exécuter des scénarios de test qui définissent une série de tâches.
11. Les microservices utilisent ce scénario de test pour exécuter des tâches ECS sur Fargate dans les régions spécifiées.
12. Une fois le test terminé, la solution stocke les résultats dans S3 et DynamoDB et enregistre les résultats sur Amazon. CloudWatch
13. Si vous activez l'option Live Data, la solution envoie les CloudWatch journaux des tâches Fargate à une fonction Lambda pendant le test pour chaque région où le test est exécuté.
14. La fonction Lambda publie les données dans le sujet correspondant dans AWS IoT Core dans la région où le stack principal a été déployé. La console Web s'abonne au sujet et affiche les données en temps réel pendant le test.

> **Note**
>
> Les étapes suivantes décrivent l'intégration optionnelle du serveur MCP pour l'analyse des tests de charge assistée par l'IA. Ce composant n'est déployé que si vous sélectionnez l'option Serveur MCP lors du déploiement de la solution.

15. Un client MCP (outil de développement d'IA) se connecte au point de terminaison AWS AgentCore Gateway pour accéder aux données de la solution de test de charge distribué via le protocole Model Context. AgentCore Gateway valide le jeton d'authentification Cognito de l'utilisateur pour garantir un accès autorisé au serveur MCP.
16. Une fois l'authentification réussie, AgentCore Gateway transmet la demande d'outil MCP à la fonction Lambda du serveur DLT MCP. La fonction Lambda renvoie les données structurées à AgentCore Gateway, qui les renvoie au client MCP pour une analyse et des informations assistées par l'IA.
17. La fonction Lambda traite la demande et interroge les ressources AWS appropriées (tables DynamoDB, compartiments S3 ou CloudWatch journaux) pour récupérer les données de test de charge demandées.

Considérations relatives à la conception d'AWS Well-Architected

Cette solution utilise les meilleures pratiques de l'AWS Well-Architected Framework, qui aide les clients à concevoir et à exploiter des charges de travail fiables, sécurisées, efficaces et rentables dans le cloud.

Cette section décrit comment les principes de conception et les meilleures pratiques du Well-Architected Framework profitent à cette solution.

Excellence opérationnelle

Cette section décrit comment nous avons conçu cette solution en utilisant les principes et les meilleures pratiques du pilier de l'excellence opérationnelle.

- Toutes les ressources sont définies comme une infrastructure sous forme de code à l'aide de CloudFormation modèles AWS générés à partir de constructions AWS CDK.

- La solution applique les métriques CloudWatch à différentes étapes afin de fournir de l'observabilité dans les fonctions Lambda, les tâches ECS, les compartiments S3 et les autres composants de la solution.

Sécurité

Cette section décrit comment nous avons conçu cette solution en utilisant les principes et les meilleures pratiques du pilier de sécurité.

- Cognito authentifie et autorise les utilisateurs de la console Web et les demandes d'API.
- Toutes les communications interservices utilisent des rôles AWS Identity and Access Management (IAM) avec le moins de privilèges d'accès, contenant uniquement les autorisations minimales requises.
- L'ensemble du stockage de données, y compris les compartiments S3 et les tables DynamoDB, crypte les données au repos à l'aide de clés gérées par AWS.
- La journalisation, le suivi et le versionnement sont activés le cas échéant à des fins d'audit et de conformité.
- L'accès au réseau est privé par défaut et les points de terminaison VPC sont activés lorsqu'ils sont disponibles pour maintenir le trafic au sein du réseau AWS.

> **Note**
>
> La solution crée plusieurs groupes de CloudWatch journaux avec des périodes de conservation différentes en fonction du volume de journaux et des considérations de coût :

Type de journal	Période de conservation
Informations sur les conteneurs ECS	1 jour
Step Functions, journaux personnalisés ECS, journaux d'accès à API Gateway	1 an
Journaux d'exécution Lambda	2 ans
Journaux d'exécution d'API Gateway	N'expire jamais

Vous pouvez modifier ces périodes de conservation dans la CloudWatch console en fonction de vos besoins.

Fiabilité

Cette section décrit comment nous avons conçu cette solution en utilisant les principes et les meilleures pratiques du pilier de fiabilité.

- La solution utilise les services sans serveur AWS dans la mesure du possible (exemples : Lambda, API Gateway, Amazon S3, AWS Step Functions, Amazon DynamoDB et AWS Fargate) pour garantir une haute disponibilité et une reprise en cas de panne de service.
- Tous les traitements informatiques utilisent les fonctions Lambda ou Amazon ECS sur AWS Fargate.
- Les données sont stockées dans DynamoDB et Amazon S3, elles sont donc conservées par défaut dans plusieurs zones de disponibilité.

Efficacité des performances

Cette section décrit comment nous avons conçu cette solution en utilisant les principes et les meilleures pratiques du pilier de l'efficacité des performances.

- La solution utilise une architecture sans serveur capable d'évoluer horizontalement selon les besoins.
- La solution peut être lancée dans toutes les régions qui prennent en charge les services AWS de cette solution, telles que : AWS Lambda, Amazon API Gateway, Amazon S3, AWS Step Functions, Amazon DynamoDB, Amazon ECS, AWS Fargate et Amazon Cognito.
- La solution utilise des services gérés dans l'ensemble afin de réduire la charge opérationnelle liée au provisionnement et à la gestion des ressources.
- La solution est automatiquement testée et déployée quotidiennement pour garantir la cohérence au fur et à mesure de l'évolution des services AWS. Elle est également revue par des architectes de solutions et des experts en la matière pour identifier les domaines à expérimenter et à améliorer.

Optimisation des coûts

Cette section décrit comment nous avons conçu cette solution en utilisant les principes et les meilleures pratiques du pilier d'optimisation des coûts.

- La solution utilise une architecture sans serveur ; par conséquent, les clients ne sont facturés que pour ce qu'ils utilisent.
- Amazon DynamoDB adapte la capacité à la demande, de sorte que vous ne payez que pour la capacité que vous utilisez.
- AWS ECS sur AWS Fargate vous permet de payer uniquement pour les ressources de calcul que vous utilisez, sans frais initiaux.
- AWS AgentCore Gateway sert de proxy économique basé sur Lambda pour l'API de test de charge distribué, éliminant ainsi le besoin d'une infrastructure dédiée et réduisant les coûts grâce à une tarification sans serveur pay-per-request.

Durabilité

Cette section décrit comment nous avons conçu cette solution en utilisant les principes et les meilleures pratiques du pilier du développement durable.

- La solution utilise des services gérés sans serveur pour minimiser l'impact environnemental des services principaux par rapport aux services sur site fonctionnant en continu.
- Les services sans serveur vous permettent d'augmenter ou de diminuer selon vos besoins.

Détails de l'architecture

Cette section décrit les composants et les services AWS qui constituent cette solution ainsi que les détails de l'architecture sur la manière dont ces composants fonctionnent ensemble.

La solution de test de charge distribué sur AWS comprend trois composants de haut niveau : un front-end, un backend et un serveur MCP en option.

Extrémité avant

Le front-end fournit les interfaces permettant d'interagir avec la solution et inclut :

- Une API de test de charge pour un accès programmatique
- Une console Web pour créer, planifier et exécuter des tests de performance
- Un serveur MCP en option pour l'analyse assistée par l'IA des résultats de test et des erreurs

API de test de charge

Les tests de charge distribués sur AWS configurent Amazon API Gateway pour héberger l' RESTful API de la solution. Les utilisateurs peuvent interagir avec le système de test de charge en toute sécurité via la console Web, RESTful l'API et le serveur MCP en option inclus. L'API fait office de « porte d'entrée » pour accéder aux données de test stockées dans Amazon DynamoDB. Vous pouvez également utiliser le APIs pour accéder à toutes les fonctionnalités étendues que vous intégrez à la solution.

Cette solution tire parti des fonctionnalités d'authentification des utilisateurs des groupes d'utilisateurs Amazon Cognito. Après avoir authentifié un utilisateur avec succès, Amazon Cognito émet un jeton Web JSON qui est utilisé pour permettre à la console d'envoyer des demandes aux solutions (points APIs de terminaison Amazon API Gateway). Les requêtes HTTPS sont envoyées par la console au APIs avec l'en-tête d'autorisation qui inclut le jeton.

Sur la base de la demande, API Gateway invoque la fonction AWS Lambda appropriée pour effectuer les tâches nécessaires sur les données stockées dans les tables DynamoDB, stocker les scénarios de test sous forme d'objets JSON dans Amazon S3, récupérer les images des métriques CloudWatch Amazon et soumettre des scénarios de test à la machine d'état AWS Step Functions.

Pour plus d'informations sur l'API de la solution, reportez-vous à la section API de test de charge distribuée de ce guide.

console Web

Cette solution inclut une console Web que vous pouvez utiliser pour configurer et exécuter des tests, surveiller les tests en cours et afficher les résultats détaillés des tests. La console est une application ReactJS construite avec Cloudscape, un système de conception open source permettant de créer des applications Web intuitives. La console est hébergée dans Amazon S3 et accessible via Amazon CloudFront. L'application utilise AWS Amplify pour s'intégrer à Amazon Cognito afin d'authentifier les utilisateurs. La console Web contient également une option permettant d'afficher les données en temps réel pour un test en cours, dans laquelle elle s'abonne à la rubrique correspondante dans AWS IoT Core.

L'URL de la console Web est le nom de domaine de CloudFront distribution qui se trouve dans les CloudFormation sorties sous forme de console. Après avoir lancé le CloudFormation modèle, vous recevrez également un e-mail contenant l'URL de la console Web et le mot de passe à usage unique pour vous y connecter.

Serveur MCP (facultatif)

Le serveur MCP (Model Context Protocol) en option fournit une interface supplémentaire permettant aux outils de développement d'IA d'accéder aux données de test de charge et de les analyser par le biais d'interactions en langage naturel. Ce composant n'est déployé que si vous sélectionnez l'option Serveur MCP lors du déploiement de la solution.

Le serveur MCP permet aux agents d'intelligence artificielle de consulter les résultats des tests, d'analyser les indicateurs de performance et d'obtenir des informations sur vos données de test de charge à l'aide d'outils tels qu'Amazon Q, Claude et d'autres assistants d'IA compatibles avec MCP. Pour des informations détaillées sur l'architecture et la configuration du serveur MCP, reportez-vous à la section Serveur MCP dans cette section.

Backend

Le backend se compose d'un pipeline d'images de conteneur et d'un moteur de test de charge que vous utilisez pour générer de la charge pour les tests. Vous interagissez avec le backend par le biais du front-end. En outre, les tâches Amazon ECS on AWS Fargate lancées pour chaque test sont associées à un identifiant de test (ID) unique. Ces étiquettes d'identification de test peuvent être utilisées pour vous aider à contrôler les coûts de cette solution. Pour plus d'informations, reportez-vous aux balises de répartition des coûts définies par l'utilisateur dans le guide de l'utilisateur d'AWS Billing and Cost Management.

Pipeline d'images de conteneurs

Cette solution utilise une image de conteneur créée avec Amazon Linux 2023 comme image de base avec le framework de test de charge Taurus installé. Taurus est un framework d'automatisation des tests open source qui prend en charge K6 JMeter, Locust et d'autres outils de test. AWS héberge cette image dans un référentiel public Amazon Elastic Container Registry (Amazon ECR). La solution utilise cette image pour exécuter des tâches dans le cluster Amazon ECS on AWS Fargate.

Pour plus d'informations, reportez-vous à la section Personnalisation des images du conteneur de ce guide.

Infrastructure de test

Outre le CloudFormation modèle principal, la solution fournit un modèle régional pour lancer les ressources nécessaires à l'exécution de tests dans plusieurs régions. La solution stocke ce modèle dans Amazon S3 et fournit un lien vers celui-ci dans la console Web. Chaque stack régional inclut un VPC, un cluster AWS Fargate et une fonction Lambda pour le traitement des données en temps réel.

Pour plus d'informations sur le déploiement d'une infrastructure de test dans des régions supplémentaires, reportez-vous à la section Déploiement multirégional de ce guide.

Moteur d'essai de charge

La solution de test de charge distribué utilise Amazon Elastic Container Service (Amazon ECS) et AWS Fargate pour simuler des milliers d'utilisateurs simultanés dans plusieurs régions, générant des requêtes HTTP à un rythme soutenu.

Vous définissez les paramètres de test à l'aide de la console Web incluse. La solution utilise ces paramètres pour générer un scénario de test JSON et le stocker dans Amazon S3. Pour plus d'informations sur les scripts de test et les paramètres de test, reportez-vous à la section Types de tests de cette section.

Une machine d'état AWS Step Functions exécute et surveille les tâches Amazon ECS dans un cluster AWS Fargate. La machine d'état AWS Step Functions inclut une fonction AWS Lambda ecr-checker, une fonction AWS Lambda, une fonction AWS Lambda de task-status-checker lancement de tâches, une fonction d'annulation de tâches AWS Lambda et une fonction AWS Lambda d'analyseur de résultats. Pour plus d'informations sur le flux de travail, reportez-vous à la section Test du flux de travail de ce guide. Pour plus d'informations sur les résultats des tests, reportez-vous à la section Résultats des tests de ce guide. Pour plus d'informations sur le flux de travail d'annulation des tests, reportez-vous à la section sur le flux de travail d'annulation des tests de ce guide.

Si vous sélectionnez des données en temps réel, la solution lance une fonction real-time-data-publisher Lambda dans chaque région à partir CloudWatch des journaux correspondant aux tâches Fargate de cette région. La solution traite et publie ensuite les données dans une rubrique d'AWS IoT Core dans la région où vous avez lancé le stack principal. Pour plus d'informations, reportez-vous à la section Données en temps réel de ce guide.

Serveur MCP

L'intégration optionnelle du serveur MCP (Model Context Protocol) permet aux agents d'intelligence artificielle d'accéder à vos données de test de charge et de les analyser par le biais d'interactions en langage naturel. Ce composant n'est déployé que si vous sélectionnez l'option Serveur MCP lors du déploiement de la solution.

Le serveur MCP fait office de pont entre les outils de développement d'IA et votre déploiement DLT, fournissant une interface standardisée pour une analyse intelligente des résultats des tests de performance. L'architecture intègre plusieurs services AWS afin de créer une interface sécurisée et évolutive pour les interactions avec les agents d'intelligence artificielle :

AgentCore Passerelle AWS

AWS AgentCore Gateway est un service entièrement géré qui fournit un hébergement standardisé et une gestion des protocoles pour les serveurs MCP. Dans cette solution, AgentCore Gateway sert de point de terminaison public auquel les agents d'IA se connectent lorsqu'ils demandent l'accès à vos données de test de charge.

Le service gère toutes les communications du protocole MCP, y compris la découverte d'outils, la validation des jetons d'authentification et le routage des demandes. AgentCore Gateway fonctionne comme un service mutualisé doté de protections de sécurité intégrées contre les menaces courantes qui pèsent sur les terminaux publics, tout en validant les signatures et les revendications des jetons Cognito pour chaque demande.

Serveur DLT MCP Lambda

La fonction Lambda du serveur DLT MCP est un composant sans serveur personnalisé qui traite les demandes MCP des agents d'intelligence artificielle et les traduit en requêtes relatives à vos ressources DLT.

Cette fonction Lambda agit comme la couche intelligente de l'intégration MCP, en récupérant les résultats des tests à partir des tables DynamoDB, en accédant aux artefacts de performance stockés

dans des compartiments S3 et en interrogeant les journaux pour obtenir des informations d'exécution détaillées. CloudWatch La fonction Lambda implémente des modèles d'accès en lecture seule et transforme les données DLT brutes en formats structurés et adaptés à l'IA que les agents peuvent facilement interpréter et analyser.

Intégration de l'authentification

Le système d'authentification tire parti de l'infrastructure existante de votre pool d'utilisateurs Cognito pour maintenir des contrôles d'accès cohérents à la fois sur la console Web et sur les interfaces du serveur MCP.

Cette intégration utilise une authentification basée sur des jetons OAuth 2.0. Les utilisateurs s'authentifient une fois via le processus de connexion Cognito et reçoivent des jetons qui fonctionnent à la fois pour les interactions avec l'interface utilisateur et pour l'accès au serveur MCP. Le système maintient les mêmes limites d'autorisation et les mêmes contrôles d'accès que l'interface Web, garantissant que les utilisateurs ne peuvent accéder que par le biais d'agents d'intelligence artificielle aux mêmes données de test de charge auxquelles ils peuvent accéder via la console.

Services AWS inclus dans cette solution

Les services AWS suivants sont inclus dans cette solution :

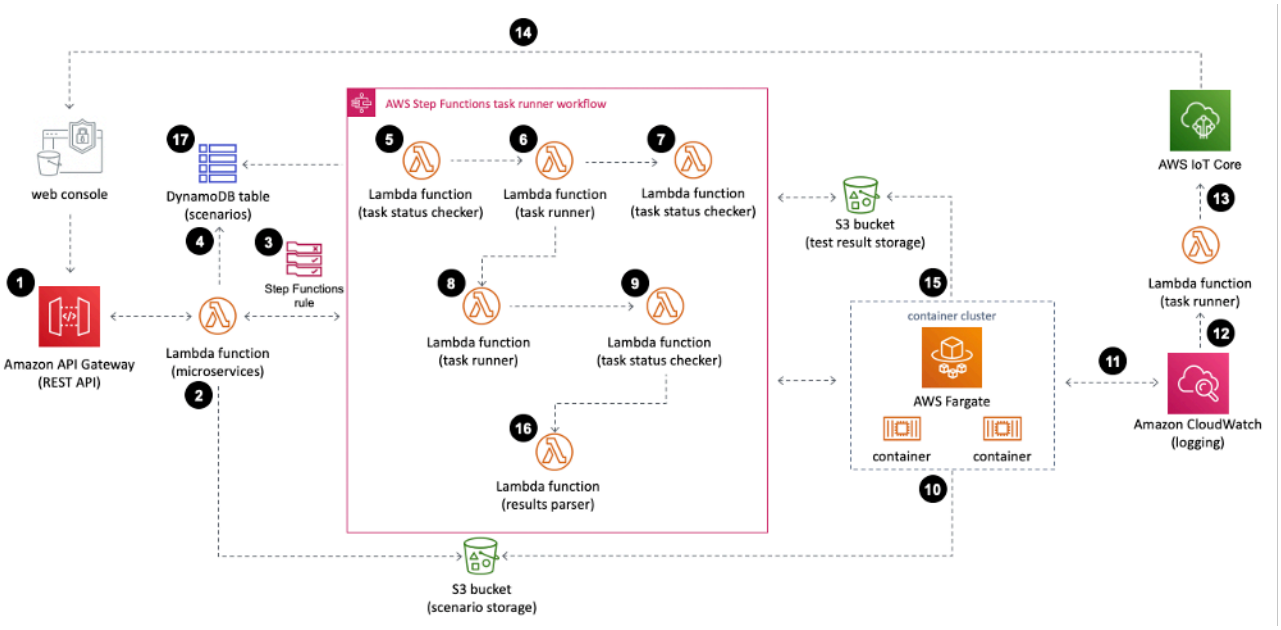
Service AWS	Description
Amazon API Gateway	Noyau. Héberge les points de terminaison de l'API REST dans la solution.
AWS CloudFormation	Noyau. Gère les déploiements de l'infrastructure de la solution.
Amazon CloudFront	Noyau. Diffuse le contenu Web hébergé dans Amazon S3.
Amazon CloudWatch	Noyau. Stocke les journaux et les indicateurs de la solution.
Amazon Cognito	Noyau. Gère la gestion des utilisateurs et l'authentification pour l'API.
Amazon DynamoDB	Noyau. Stocke les informations de déploiement et teste les détails et les résultats des scénarios.

Service AWS	Description
Amazon Elastic Container Service	Noyau. Déploie et gère des tâches Amazon ECS indépendantes sur des conteneurs AWS Fargate.
AWS Fargate	Noyau. Héberge les conteneurs Amazon ECS de la solution
AWS Identity and Access Management	Noyau. Gère la gestion des rôles et des autorisations des utilisateurs.
AWS Lambda	Noyau. Fournit une logique pour la APIs mise en œuvre, l'analyse des résultats des tests et le lancement de workers/leader tâches.
AWS Step Functions	Noyau. Orchestre le provisionnement des conteneurs Amazon ECS sur les tâches AWS Fargate dans les régions spécifiées
AWS Amplify	Soutenir. Fournit une console Web alimentée par AWS Amplify.
CloudWatch Événements Amazon	Soutenir. Planifie les tests pour qu'ils commencent automatiquement à une date spécifiée ou à des dates récurrentes.
Amazon Elastic Container Registry	Soutenir. Héberge l'image du conteneur dans un référentiel ECR public.
Noyau d'AWS IoT	Soutenir. Permet de visualiser les données en temps réel pour un test en cours en vous abonnant à la rubrique correspondante dans AWS IoT Core.
AWS Systems Manager	Soutenir. Assure la surveillance des ressources au niveau de l'applica tion et la visualisation des opérations sur les ressources et des données de coûts.
Amazon S3	Soutenir. Héberge le contenu Web statique, les journaux, les métriques et les données de test.
Amazon Virtual Private Cloud	Soutenir. Contient les conteneurs Amazon ECS de la solution exécutés sur AWS Fargate.
Amazon Bedrock AgentCore	Support, facultatif. Héberge le serveur MCP (Remote Model Context Protocol) optionnel de la solution pour l'intégration de l'agent AI à l'API.

Comment fonctionnent les tests de charge distribués sur AWS

La répartition détaillée suivante montre les étapes nécessaires à l'exécution d'un scénario de test.

Flux de travail de test



1. Vous utilisez la console Web pour soumettre un scénario de test incluant les détails de configuration à l'API de la solution.
2. La configuration du scénario de test est téléchargée sur Amazon Simple Storage Service (Amazon S3) sous forme de fichier `s3://<bucket-name>/test-scenarios/<$TEST_ID>/<$TEST_ID>.json` JSON ().
3. Une machine d'état AWS Step Functions s'exécute en utilisant l'ID de test, le nombre de tâches, le type de test et le type de fichier comme entrée de la machine d'état AWS Step Functions. Si le test est planifié, il créera d'abord une règle d' CloudWatch événements, qui déclenchera AWS Step Functions à la date spécifiée. Pour plus de détails sur le flux de travail de planification, reportez-vous à la section Processus de planification des tests de ce guide.
4. Les détails de configuration sont stockés dans le tableau des scénarios Amazon DynamoDB.
5. Dans le flux de travail du lanceur de tâches AWS Step Functions, la fonction task-status-checker AWS Lambda vérifie si les tâches Amazon Elastic Container Service (Amazon ECS) sont déjà en cours d'exécution pour le même ID de test. Si des tâches portant le même ID de test sont détectées en cours d'exécution, cela provoque une erreur. Si aucune tâche Amazon ECS n'est

exécutée dans le cluster AWS Fargate, la fonction renvoie l'ID de test, le nombre de tâches et le type de test.

6. La fonction AWS Lambda d'exécution de tâches obtient les détails des tâches de l'étape précédente et exécute les tâches de travail Amazon ECS dans le cluster AWS Fargate. L'API Amazon ECS utilise l' RunTask action pour exécuter les tâches de travail. Ces tâches de travail sont lancées, puis attendent un message de démarrage de la tâche principale pour commencer le test. L' RunTask action est limitée à 10 tâches par définition. Si le nombre de tâches est supérieur à 10, la définition des tâches sera exécutée plusieurs fois jusqu'à ce que toutes les tâches de travail aient été démarrées. La fonction génère également un préfixe pour distinguer le test en cours dans la fonction AWS Lambda d'analyse des résultats.
7. La fonction task-status-checker AWS Lambda vérifie si toutes les tâches de travail Amazon ECS sont exécutées avec le même ID de test. Si les tâches sont toujours en cours de provisionnement, il attend une minute et vérifie à nouveau. Une fois que toutes les tâches Amazon ECS sont exécutées, il renvoie l'ID de test, le nombre de tâches, le type de test, toutes les tâches IDs et le préfixe et les transmet à la fonction de lancement de tâches.
8. La fonction de lancement de tâches AWS Lambda s'exécute à nouveau, en lançant cette fois une seule tâche Amazon ECS qui servira de nœud principal. Cette tâche ECS envoie un message de test de démarrage à chacune des tâches de travail afin de démarrer les tests simultanément.
9. La fonction task-status-checker AWS Lambda vérifie à nouveau si les tâches Amazon ECS sont exécutées avec le même ID de test. Si les tâches sont toujours en cours d'exécution, il attend une minute et vérifie à nouveau. Lorsqu'aucune tâche Amazon ECS n'est en cours d'exécution, il renvoie l'ID de test, le nombre de tâches, le type de test et le préfixe.
10. Lorsque la fonction de lancement de tâches AWS Lambda exécute les tâches Amazon ECS dans le cluster AWS Fargate, chaque tâche télécharge la configuration de test depuis Amazon S3 et lance le test.
11. Une fois les tests exécutés, le temps de réponse moyen, le nombre d'utilisateurs simultanés, le nombre de demandes réussies et le nombre de demandes ayant échoué pour chaque tâche sont enregistrés sur Amazon CloudWatch et peuvent être consultés dans un CloudWatch tableau de bord.
12. Si vous avez inclus des données en temps réel dans le test, la solution filtre les résultats du test en temps réel à CloudWatch l'aide d'un filtre d'abonnement. La solution transmet ensuite les données à une fonction Lambda.
13. La fonction Lambda structure ensuite les données reçues et les publie dans une rubrique AWS IoT Core.

14 La console Web s'abonne à la rubrique AWS IoT Core du test et reçoit les données publiées dans cette rubrique pour représenter graphiquement les données en temps réel pendant l'exécution du test.

15 Lorsque le test est terminé, les images du conteneur exportent un rapport détaillé sous forme de fichier XML vers Amazon S3. Un UUID est attribué à chaque fichier pour son nom de fichier. Par exemple, s3://dlte-bucket/test-scenarios/ <$TEST_ID> /results/ <$UUID> .json.

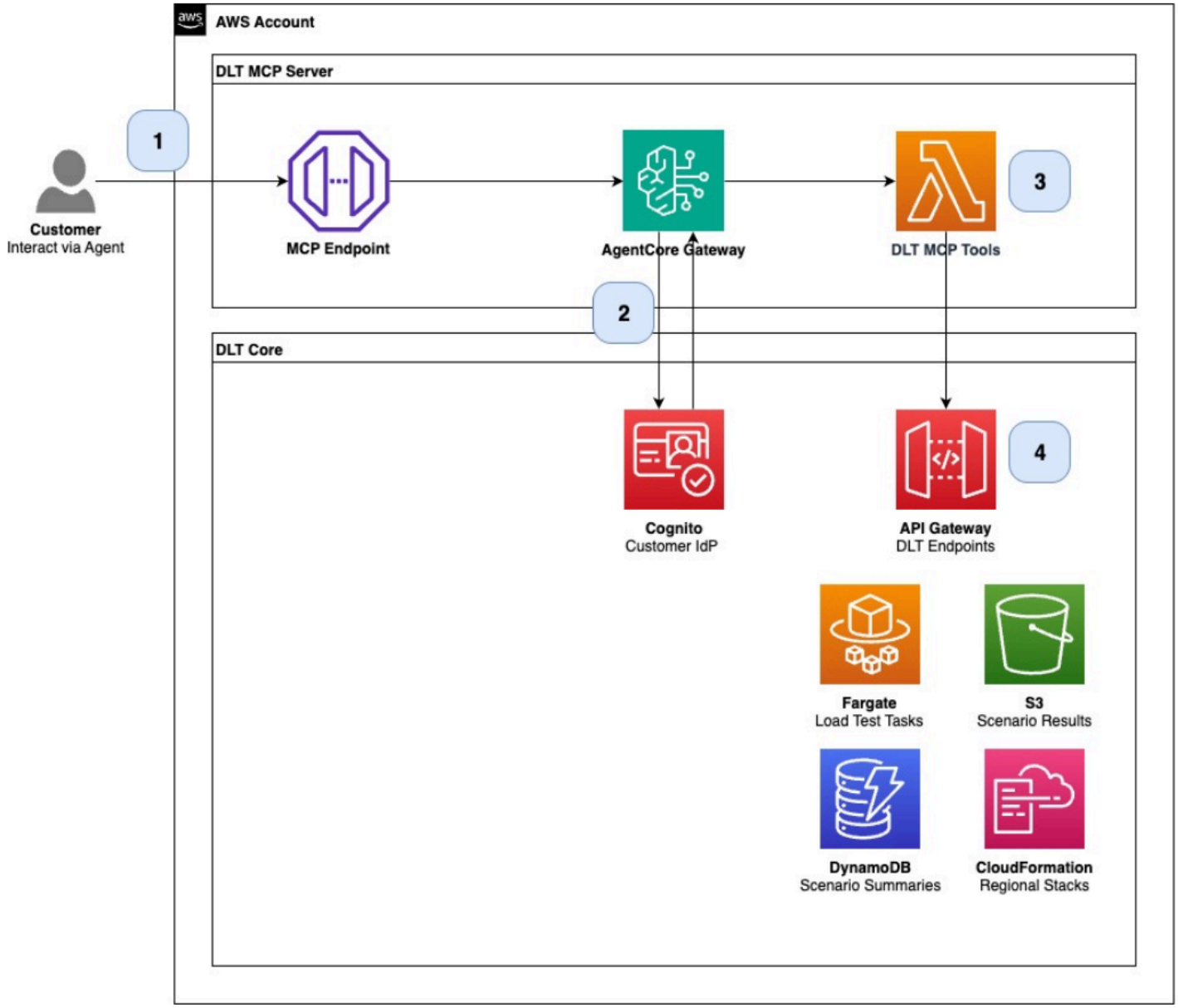
16 Lorsque les fichiers XML sont chargés sur Amazon S3, la fonction AWS Lambda de l'analyseur de résultats lit les résultats dans les fichiers XML en commençant par le préfixe, puis analyse et agrège tous les résultats en un seul résultat résumé.

17 La fonction AWS Lambda de l'analyseur de résultats écrit le résultat agrégé dans une table Amazon DynamoDB.

Flux de travail du serveur MCP (facultatif)

Si vous déployez l'intégration optionnelle du serveur MCP, les agents AI peuvent accéder à vos données de test de charge et les analyser via le flux de travail suivant :

Architecture du serveur MCP



1. Interaction avec le client : le client interagit avec le MCP de DLT via le point de terminaison MCP hébergé par AWS Gateway. AgentCore Les agents d'IA se connectent à ce point de terminaison pour demander l'accès aux données de test de charge.
2. Autorisation : AgentCore Gateway gère les autorisations relatives au client d'application Solution Cognito du pool d'utilisateurs. La passerelle valide le jeton Cognito de l'utilisateur pour s'assurer qu'il est autorisé à accéder au serveur DLT MCP. L'accès est accordé aux utilisateurs autorisés, l'accès à l'outil agent étant limité aux opérations en lecture seule.

3. Spécification de l'outil : la AgentCore passerelle se connecte à la fonction Lambda du serveur DLT MCP. Une spécification d'outil définit les outils disponibles que les agents d'IA peuvent utiliser pour interagir avec vos données de test de charge.
4. Accès aux API en lecture seule : la fonction Lambda est limitée à l'accès aux API en lecture seule via les points de terminaison DLT API Gateway existants. La fonction fournit quatre opérations principales :
   - Répertorier les scénarios - Récupérez une liste de scénarios de test dans le tableau des scénarios DynamoDB
   - Obtenez les résultats des tests de scénarios - Accédez aux résultats de test détaillés pour des scénarios spécifiques à partir de DynamoDB et S3
   - Get Fargate Load Test Runners : recherchez des informations sur l'exécution de tâches Fargate dans le cluster ECS
   - Obtenez des piles régionales disponibles - Récupérez des informations sur l'infrastructure régionale déployée auprès de CloudFormation

L'intégration du serveur MCP tire parti de l'infrastructure DLT existante (API Gateway, Cognito, DynamoDB, S3) pour fournir un accès sécurisé en lecture seule aux données de test pour des analyses et des informations basées sur l'IA.

Considérations relatives à la conception

Cette section décrit les décisions de conception et les options de configuration importantes pour la solution de test de charge distribué sur AWS, y compris les applications prises en charge, les types de tests, les options de planification et les considérations relatives au déploiement.

Applications prises en charge

Cette solution permet de tester des applications basées sur le cloud et des applications sur site, à condition que vous disposiez d'une connectivité réseau entre votre compte AWS et votre application. La solution prend en APIs charge l'utilisation des protocoles HTTP ou HTTPS.

Types de tests

Les tests de charge distribués sur AWS prennent en charge plusieurs types de tests : tests de point de terminaison HTTP simples JMeter, K6 et Locust.

Tests de point de terminaison HTTP simples

La console Web fournit une interface de configuration de point de terminaison HTTP qui vous permet de tester n'importe quel point de terminaison HTTP ou HTTPS sans écrire de scripts personnalisés. Vous définissez l'URL du point de terminaison, sélectionnez la méthode HTTP (GET, POST, PUT, DELETE, etc.) dans un menu déroulant et ajoutez éventuellement des en-têtes de requête et des charges utiles personnalisés. Cette configuration vous permet de tester APIs avec des jetons d'autorisation personnalisés, des types de contenu ou tout autre en-tête HTTP et corps de requête requis par votre application.

JMeter tests

Lorsque vous créez un scénario de test à l'aide de la console Web, vous pouvez télécharger un script de JMeter test. La solution télécharge le script dans le compartiment S3 des scénarios. Lorsque les tâches Amazon ECS sont exécutées, elles téléchargent le JMeter script depuis S3 et exécutent le test.

> **Important**
>
> Bien que votre JMeter script puisse définir la simultanéité (utilisateurs virtuels), les taux de transaction (TPS), les temps de montée en puissance et d'autres paramètres de chargement, la solution remplacera ces configurations par les valeurs que vous spécifiez sur l'écran Traffic Shape lors de la création du test. La configuration Traffic Shape contrôle le nombre de tâches, la simultanéité (utilisateurs virtuels par tâche), la durée de montée en puissance et la durée de suspension pour l'exécution du test.

Si vous avez des fichiers JMeter d'entrée, vous pouvez les compresser avec le JMeter script. Vous pouvez choisir le fichier zip lorsque vous créez un scénario de test.

Si vous souhaitez inclure des plugins, tous les fichiers .jar inclus dans un sous-répertoire /plugins du fichier zip fourni seront copiés dans le répertoire des JMeter extensions et seront disponibles pour les tests de charge.

> **Note**
>
> Si vous incluez des fichiers JMeter d'entrée dans votre fichier de JMeter script, vous devez inclure le chemin relatif des fichiers d'entrée dans votre fichier de JMeter script. En outre, les fichiers d'entrée doivent se trouver dans le chemin relatif. Par exemple, si vos fichiers JMeter

d'entrée et votre fichier de script se trouvent plutôt dans le home/user directory and you refer to the input files in the JMeter script file, the path of input files must be ./INPUT_FILES. If you use /home/user/INPUT répertoire/_FILES, le test échouera car il ne sera pas en mesure de trouver les fichiers d'entrée.

Si vous incluez JMeter des plug-ins, les fichiers .jar doivent être regroupés dans un sous-répertoire nommé /plugins à la racine du fichier zip. Par rapport à la racine du fichier zip, le chemin d'accès aux fichiers jar doit être. /Plugins/bundled_plugin.jar.

Pour plus d'informations sur l'utilisation JMeter des scripts, reportez-vous au manuel de JMeter l'utilisateur.

Tests K6

La solution prend en charge les tests basés sur le framework K6. K6 est publié sous licence AGPL-3.0. La solution affiche un message d'accusé de réception de licence lors de la création d'un nouveau test K6. Vous pouvez télécharger le fichier de test K6 ainsi que tous les fichiers d'entrée nécessaires dans un fichier d'archive.

Important

Bien que votre script K6 puisse définir la simultanéité (utilisateurs virtuels), les étapes, les seuils et d'autres paramètres de charge, la solution remplacera ces configurations par les valeurs que vous spécifiez sur l'écran Traffic Shape lors de la création du test. La configuration Traffic Shape contrôle le nombre de tâches, la simultanéité (utilisateurs virtuels par tâche), la durée de montée en puissance et la durée de suspension pour l'exécution du test.

Tests acridiens

La solution prend en charge les tests basés sur le framework Locust. Vous pouvez télécharger le fichier de test Locust ainsi que tous les fichiers d'entrée nécessaires dans un fichier d'archive.

Important

Bien que votre script Locust puisse définir la simultanéité (nombre d'utilisateurs), le taux d'apparition et d'autres paramètres de charge, la solution remplacera ces configurations

par les valeurs que vous spécifiez sur l'écran Traffic Shape lors de la création du test. La configuration Traffic Shape contrôle le nombre de tâches, la simultanéité (utilisateurs virtuels par tâche), la durée de montée en puissance et la durée de suspension pour l'exécution du test.

Planification des tests

La solution propose trois options de synchronisation d'exécution pour exécuter des tests de charge :

- Exécuter maintenant - Exécute le test de charge immédiatement après la création
- Exécuter une fois : exécute le test à une date et à une heure spécifiques dans le futur
- Exécuter selon un calendrier : créez des tests récurrents à l'aide d'expressions cron pour définir le calendrier

Lorsque vous sélectionnez Exécuter une fois, vous spécifiez la durée d'exécution au format 24 heures et la date d'exécution à laquelle le test de charge doit commencer.

Lorsque vous sélectionnez Exécuter selon un calendrier, vous pouvez soit saisir manuellement une expression cron, soit sélectionner l'un des modèles cron courants (par exemple toutes les heures, tous les jours à une heure précise, en semaine ou tous les mois). L'expression cron utilise un format de planification précis avec des champs pour les minutes, les heures, le jour du mois, le mois, le jour de la semaine et l'année. Vous devez également spécifier une date d'expiration, qui définit le moment où le test planifié doit cesser de fonctionner. Pour plus d'informations sur le fonctionnement de la planification, reportez-vous à la section Workflow de planification des tests de ce guide.

Note

- Durée des tests : tenez compte de la durée totale des tests lors de la planification. Par exemple, un test avec un temps de démarrage de 10 minutes et un temps d'attente de 40 minutes prendra environ 80 minutes.
- Intervalle minimum : assurez-vous que l'intervalle entre les tests planifiés est supérieur à la durée estimée du test. Par exemple, si le test dure environ 80 minutes, programmez-le pour qu'il ne soit pas exécuté plus fréquemment que toutes les 3 heures.
- Limite horaire : le système ne permet pas de planifier les tests avec une différence d'une heure seulement, même si la durée estimée du test est inférieure à une heure.

Tests simultanés

Cette solution crée un CloudWatch tableau de bord Amazon pour chaque test qui affiche le résultat combiné de toutes les tâches exécutées dans le cluster Amazon ECS en temps réel. Le CloudWatch tableau de bord indique le temps de réponse moyen, le nombre d'utilisateurs simultanés, le nombre de demandes réussies et le nombre de demandes ayant échoué. La solution agrège chaque métrique par seconde et met à jour le tableau de bord toutes les minutes.

Gestion des utilisateurs

Lors de la configuration initiale, vous fournissez un nom d'utilisateur et une adresse e-mail qu'Amazon Cognito utilise pour vous donner accès à la console Web de la solution. La console n'assure pas l'administration des utilisateurs. Pour ajouter des utilisateurs supplémentaires, vous devez utiliser la console Amazon Cognito. Pour plus d'informations, reportez-vous à la section Gestion des utilisateurs dans les groupes d'utilisateurs du manuel Amazon Cognito Developer Guide.

Pour la migration d'utilisateurs existants vers des groupes d'utilisateurs Amazon Cognito, consultez le blog AWS Approaches for migration des utilisateurs vers des groupes d'utilisateurs Amazon Cognito.

Déploiement régional

Cette solution utilise Amazon Cognito, disponible uniquement dans certaines régions AWS. Par conséquent, vous devez déployer cette solution dans une région où Amazon Cognito est disponible. Pour connaître la disponibilité des services la plus récente par région, consultez la liste des services régionaux AWS.

Planifiez votre déploiement

Cette section décrit les coûts, la sécurité, les régions prises en charge, les quotas et les autres considérations que vous devez prendre en compte avant de déployer la solution.

Coût

Vous êtes responsable du coût des services AWS utilisés lors de l'exécution de cette solution. Le coût total dépend du nombre de tests de charge exécutés, de la durée de ces tests et de la quantité de données générées. À partir de cette révision, le coût estimé de l'exécution de cette solution avec les paramètres par défaut dans la région de l'est des États-Unis (Virginie du Nord) est d'environ 30,90$ par mois.

Le tableau suivant fournit un exemple de ventilation des coûts pour le déploiement de cette solution avec les paramètres par défaut dans la région USA Est (Virginie du Nord) pendant un mois.

Service AWS	Dimensions	Coût [USD]
AWS Fargate	10 tâches à la demande (utilisant deux V CPUs et 4 Go de mémoire) exécutées pendant 30 heures	29,62$
Amazon DynamoDB	1 000 unités de capacité d'écriture à la demande 1 000 unités de capacité de lecture à la demande	0,0015$
AWS Lambda	1 000 demandes Durée totale de 10 minutes	1,25$
AWS Step Functions	1 000 transitions entre États	0,025 USD
Au total :		30,90$ par mois

Les ressources de solution sont étiquetées avec les balises Key= SolutionId et Value=SO0062. Vous pouvez activer la clé de balise SolutionId en suivant la documentation sur l'activation des balises. Une fois le tag activé, vous pouvez créer une règle de catégorie de coûts en suivant la documentation pour créer des catégories de coûts. Vous pouvez consulter le coût de la solution en surveillant la console des catégories de coûts et en sélectionnant le nom de la catégorie de coûts.

Nous vous recommandons de créer un budget via AWS Cost Explorer pour vous aider à gérer les coûts. Les prix sont susceptibles d'être modifiés. Pour plus de détails, consultez la page Web de tarification de chaque service AWS utilisé dans cette solution.

> **Note**
>
> La configuration de tâche par défaut utilise 2 V CPUs et 4 Go de mémoire par tâche. Si vos tests de charge ne nécessitent pas ces ressources, vous pouvez les réduire pour réduire les coûts. Inversement, vous pouvez augmenter les ressources pour favoriser une plus grande simultanéité par tâche. Pour plus d'informations, reportez-vous à la section Augmenter les ressources en conteneurs de ce guide.

> **Note**
>
> Cette solution offre la possibilité d'inclure des données en temps réel lors de l'exécution d'un test. Cette fonctionnalité nécessite une fonction AWS Lambda supplémentaire et une rubrique AWS IoT Core qui entraînent des coûts supplémentaires.

Coûts supplémentaires liés au serveur MCP (facultatif)

Le tableau suivant fournit une ventilation des coûts pour l'intégration du serveur MCP avec les tarifs dans la région de l'est des États-Unis (Virginie du Nord) pour un mois.

Composant de service	Dimensions	Coût [USD]
AgentCore Passerelle - Indexation des outils	10 outils × 0,02$ par 100 outils	0,002$
AgentCore Passerelle - API de recherche	10 000 interactions × 0,025$ pour 1 000	0,25$

Composant de service	Dimensions	Coût [USD]
AgentCore Passerelle - Invocations d'API	50 000 invocations × 0,005$ par 1 000	0,25$
Fonction AWS Lambda	Variable en fonction de l'utilisation (charges de travail typiques)	5,00$ - 20,00$
Coût supplémentaire total estimé :		5,50$ - 20,50$ par mois

Les prix sont susceptibles d'être modifiés. Pour plus de détails sur la tarification de AgentCore Gateway, consultez la section Tarification d'Amazon Bedrock (section AgentCore Gateway). Pour connaître les tarifs Lambda, reportez-vous à la section Tarification AWS Lambda.

Sécurité

Lorsque vous créez des systèmes sur l'infrastructure AWS, les responsabilités en matière de sécurité sont partagées entre vous et AWS. Ce modèle de responsabilité partagée réduit votre charge opérationnelle car AWS exploite, gère et contrôle les composants, notamment le système d'exploitation hôte, la couche de virtualisation et la sécurité physique des installations dans lesquelles les services fonctionnent. Pour plus d'informations sur la sécurité AWS, rendez-vous sur AWS Cloud Security.

Rôles IAM

Les rôles AWS Identity and Access Management (IAM) permettent aux clients d'attribuer des politiques d'accès et des autorisations détaillées aux services et aux utilisateurs sur le cloud AWS. Cette solution crée des rôles IAM qui accordent aux fonctions AWS Lambda de la solution l'accès pour créer des ressources régionales.

Amazon CloudFront

Cette solution déploie une interface utilisateur Web hébergée dans un compartiment Amazon S3, distribué par Amazon CloudFront. Afin de réduire le temps de latence et d'améliorer la sécurité, cette solution inclut une CloudFront distribution dotée d'une identité d'accès d'origine, à savoir un

CloudFront utilisateur fournissant un accès public au contenu du bucket du site Web de la solution. Par défaut, la CloudFront distribution utilise le protocole TLS 1.2 pour appliquer le plus haut niveau de protocole de sécurité. Pour plus d'informations, reportez-vous à la section Restreindre l'accès à une origine Amazon S3 dans le manuel Amazon CloudFront Developer Guide.

CloudFront active des mesures de sécurité supplémentaires pour ajouter des en-têtes de sécurité HTTP à chaque réponse du spectateur. Pour plus d'informations, reportez-vous à la section Ajout ou suppression d'en-têtes HTTP dans les CloudFront réponses.

Cette solution utilise le CloudFront certificat par défaut, dont le protocole de sécurité minimum pris en charge est TLS v1.0. Pour imposer l'utilisation de TLS v1.2 ou TLS v1.3, vous devez utiliser un certificat SSL personnalisé au lieu du certificat par défaut. CloudFront Pour plus d'informations, reportez-vous à Comment configurer ma CloudFront distribution pour utiliser un SSL/TLS certificat.

Amazon API Gateway

Cette solution déploie des points de terminaison Amazon API Gateway optimisés RESTful APIs pour les périphériques afin de fournir la fonctionnalité de test de charge en utilisant le point de terminaison API Gateway par défaut plutôt qu'un domaine personnalisé. Pour une optimisation en périphérie à APIs l'aide du point de terminaison par défaut, API Gateway utilise la politique de sécurité TLS-1-0. Pour plus d'informations, reportez-vous à la section Working with REST APIs du manuel Amazon API Gateway Developer Guide.

Cette solution utilise le certificat API Gateway par défaut, dont le protocole de sécurité minimum pris en charge est TLS v1.0. Pour imposer l'utilisation de TLS v1.2 ou TLS v1.3, vous devez utiliser un domaine personnalisé avec un certificat SSL personnalisé au lieu du certificat API Gateway par défaut. Pour plus d'informations, reportez-vous à la section Configuration de noms de domaine personnalisés pour REST APIs.

Groupe de sécurité AWS Fargate

Par défaut, cette solution ouvre la règle sortante du groupe de sécurité AWS Fargate au public. Si vous souhaitez empêcher AWS Fargate d'envoyer du trafic partout, remplacez la règle sortante par un routage interdomaine sans classe (CIDR) spécifique.

Ce groupe de sécurité inclut également une règle entrante qui autorise le trafic local sur le port 50 000 à destination de toute source appartenant au même groupe de sécurité. Ceci est utilisé pour permettre aux conteneurs de communiquer entre eux.

Test de stress du réseau

Vous êtes responsable de l'utilisation de cette solution dans le cadre de la politique relative aux tests de stress du réseau. Cette politique couvre les situations telles que lorsque vous prévoyez d'exécuter des tests réseau à volume élevé directement depuis vos EC2 instances Amazon vers d'autres sites, tels que d'autres EC2 instances Amazon, des propriétés/services AWS ou des points de terminaison externes. Ces tests sont parfois appelés tests de stress, tests de charge ou tests Gameday. La plupart des tests clients ne sont pas concernés par cette politique ; toutefois, référez-vous à cette politique si vous pensez que vous allez générer un trafic qui soutiendra, au total, pendant plus d'une minute, plus de 1 Gbit/s (1 milliard de bits par seconde) ou plus de 1 Gbit/s (1 milliard de paquets par seconde).

Restreindre l'accès à l'interface utilisateur publique

Pour restreindre l'accès à l'interface utilisateur destinée au public au-delà des mécanismes d'authentification et d'autorisation fournis par IAM et Amazon Cognito, utilisez la solution d'automatisation de sécurité AWS WAF (pare-feu d'applications Web).

Cette solution déploie automatiquement un ensemble de règles AWS WAF qui filtrent les attaques Web courantes. Les utilisateurs peuvent choisir parmi les fonctionnalités de protection préconfigurées qui définissent les règles incluses dans une liste de contrôle d'accès Web (ACL Web) AWS WAF.

Sécurité du serveur MCP (facultatif)

Si vous déployez l'intégration optionnelle du serveur MCP, la solution utilise AWS AgentCore Gateway pour fournir un accès sécurisé aux données de test de charge pour les agents d'intelligence artificielle. AgentCore Gateway valide les jetons d'authentification Amazon Cognito pour chaque demande, garantissant ainsi que seuls les utilisateurs autorisés peuvent accéder au serveur MCP. La fonction Lambda du serveur MCP implémente des modèles d'accès en lecture seule, empêchant les agents d'intelligence artificielle de modifier les configurations ou les résultats des tests. Toutes les interactions avec le serveur MCP utilisent les mêmes limites d'autorisation et les mêmes contrôles d'accès que ceux de la console Web.

Régions AWS prises en charge

Cette solution utilise le service Amazon Cognito, qui n'est actuellement pas disponible dans toutes les régions AWS. Pour connaître la disponibilité la plus récente des services AWS par région, consultez la liste des services régionaux AWS.

Les tests de charge distribués sur AWS sont disponibles dans les régions AWS suivantes :

Nom de la région	
USA Est (Ohio)	Asie-Pacifique (Tokyo)
USA Est (Virginie du Nord)	Canada (Centre)
USA Ouest (Californie du Nord)	Europe (Francfort)
USA Ouest (Oregon)	Europe (Irlande)
Asie-Pacifique (Mumbai)	Europe (Londres)
Asie-Pacifique (Séoul)	Europe (Paris)
Asie-Pacifique (Singapour)	Europe (Stockholm)
Asie-Pacifique (Sydney)	Amérique du Sud (São Paulo)

Régions AWS prises en charge par le serveur MCP (facultatif)

Si vous prévoyez de déployer l'intégration optionnelle du serveur MCP, vous devez déployer la solution dans une région AWS où AWS AgentCore Gateway est disponible. La fonctionnalité du serveur MCP n'est disponible que dans les régions AWS suivantes :

Nom de la région	Code région
USA Est (Virginie du Nord)	us-east-1
USA Ouest (Oregon)	us-west-2
Asie-Pacifique (Singapour)	ap-southeast-1
Asie-Pacifique (Sydney)	ap-southeast-2
Asia Pacific (Tokyo)	ap-northeast-1
Europe (Francfort)	eu-central-1

Nom de la région	Code région
Europe (Irlande)	eu-west-1
Europe (Londres)	eu-west-2
Europe (Paris)	eu-west-3

Pour connaître la disponibilité la plus récente d'AWS AgentCore Gateway par région, reportez-vous aux points de terminaison et quotas AWS AgentCore Gateway dans le guide du développeur d'AWS AgentCore Gateway.

Quotas

Les quotas de service, également appelés limites, représentent le nombre maximal de ressources ou d'opérations de service pour votre compte AWS.

Quotas pour les services AWS dans cette solution

Assurez-vous de disposer d'un quota suffisant pour chacun des services mis en œuvre dans cette solution. Pour de plus amples informations, veuillez consulter Quotas de service AWS.

Utilisez les liens suivants pour accéder à la page de ce service. Pour consulter les quotas de service pour tous les services AWS dans la documentation sans changer de page, consultez plutôt les informations de la page Points de terminaison et quotas du service dans le PDF.

CloudFormation Quotas AWS

Votre compte AWS comporte CloudFormation des quotas AWS dont vous devez tenir compte lorsque vous lancez la pile dans cette solution. En comprenant ces quotas, vous pouvez éviter les erreurs de limitation qui vous empêcheraient de déployer correctement cette solution. Pour plus d'informations, consultez CloudFormation les quotas AWS dans le guide de CloudFormation l'utilisateur AWS.

Quotas de test de charge

Le nombre maximum de tâches pouvant être exécutées dans Amazon ECS à l'aide du type de lancement AWS Fargate dépend de la taille du vCPU des tâches. La taille de tâche par défaut dans les tests de charge distribués sur AWS est de 2 vCPU. Pour connaître les quotas par défaut actuels,

reportez-vous à la section Quotas de service Amazon ECS. Les quotas du compte courant peuvent différer des quotas listés. Pour vérifier les quotas spécifiques à un compte, vérifiez le quota de service pour le nombre de ressources de vCPU à la demande de Fargate dans l'AWS Management Console. Pour savoir comment demander une augmentation, reportez-vous aux quotas de service AWS dans le guide de référence général AWS.

L'image du conteneur Amazon Linux 2023 (avec Taurus installé) ne limite pas les connexions simultanées par tâche, mais cela ne signifie pas qu'elle peut prendre en charge un nombre illimité d'utilisateurs. Pour déterminer le nombre d'utilisateurs simultanés que les conteneurs peuvent générer pour un test, reportez-vous à la section Déterminer le nombre d'utilisateurs de ce guide.

> **Note**
>
> La limite recommandée pour les utilisateurs simultanés sur la base des paramètres par défaut est de 200 utilisateurs.

Tests simultanés

Cette solution crée un CloudWatch tableau de bord Amazon pour chaque test qui affiche le résultat combiné de toutes les tâches exécutées dans le cluster Amazon ECS en temps réel. Le CloudWatch tableau de bord indique le temps de réponse moyen, le nombre d'utilisateurs simultanés, le nombre de demandes réussies et le nombre de demandes ayant échoué. La solution agrège chaque métrique par seconde et met à jour le tableau de bord toutes les minutes.

Politique d'Amazon en matière de EC2 tests

Vous n'avez pas besoin de l'approbation d'AWS pour exécuter des tests de charge à l'aide de cette solution tant que le trafic de votre réseau reste inférieur à 1 Gbit/s. Si votre test génère plus de 1 Gbit/s, contactez AWS. Pour plus d'informations, consultez la politique de EC2 test d'Amazon.

Politique d'Amazon en matière de tests de CloudFront charge

Si vous prévoyez de tester la charge d'un CloudFront point de terminaison, reportez-vous aux directives relatives aux tests de charge du manuel Amazon CloudFront Developer Guide. Nous avons également recommandé de répartir le trafic entre plusieurs tâches et régions. Prévoyez au moins 30 minutes de temps de montée en puissance pour le test de charge. Pour les tests de charge envoyant plus de 500 000 demandes par seconde ou exigeant plus de 300 Gbit/s de données,

nous vous recommandons d'obtenir au préalable une approbation préalable pour l'envoi du trafic. CloudFront peut limiter le trafic de test de charge non approuvé, ce qui a un impact sur la disponibilité du CloudFront service.

Surveillance de la solution après le déploiement

Après le déploiement de la solution, nous vous recommandons de surveiller en permanence les ressources de la solution à l'aide des CloudWatch alarmes et des métriques Amazon.

Configuration des CloudWatch alarmes

Vous pouvez configurer des CloudWatch alarmes pour surveiller les indicateurs clés et recevoir des notifications lorsque les seuils sont dépassés. Envisagez de configurer des alarmes pour les ressources suivantes :

Métriques CloudFront de distribution Amazon

Surveillez les performances et les erreurs de CloudFront distribution. Pour plus d'informations, consultez les statistiques de CloudFront distribution dans le manuel Amazon CloudFront Developer Guide.

Métriques pour Amazon API Gateway

Surveillez les taux de demandes d'API, la latence et les erreurs. Pour plus d'informations, consultez les dimensions et statistiques d'Amazon API Gateway dans le manuel du développeur Amazon API Gateway.

Métriques de la fonction AWS Lambda

Surveillez les appels, la durée, les erreurs et les limites des fonctions Lambda pour les microservices de la solution.

Métriques Amazon ECS et AWS Fargate

Surveillez l'utilisation du processeur et de la mémoire des tâches pendant les tests de charge pour garantir des ressources adéquates.

Métriques Amazon DynamoDB

Surveillez la consommation de capacité de lecture et d'écriture, les demandes limitées et la latence.

Déployez la solution

Cette solution utilise des CloudFormation modèles et des piles AWS pour automatiser son déploiement. Les CloudFormation modèles spécifient les ressources AWS incluses dans cette solution et leurs propriétés. La CloudFormation pile fournit les ressources décrites dans les modèles.

Vue d'ensemble du processus de déploiement

Avant de déployer la solution, examinez le coût, l'architecture, la sécurité et les autres considérations évoquées plus haut dans ce guide.

Temps de déploiement : environ 15 minutes pour la pile principale, plus 5 minutes pour chaque région supplémentaire

CloudFormation Modèle AWS

Vous pouvez télécharger le CloudFormation modèle de cette solution avant de la déployer. Cette solution utilise AWS CloudFormation pour automatiser le déploiement des tests de charge distribués sur AWS. Il inclut le CloudFormation modèle AWS suivant, que vous pouvez télécharger avant le déploiement :

View template distribu

load-testing-on-aws.template - Utilisez ce modèle pour lancer la solution et tous les composants associés. La configuration par défaut déploie les services principaux et de support figurant dans les services AWS de cette section de solution, mais vous pouvez personnaliser le modèle en fonction de vos besoins spécifiques.

> **Note**
>
> Les CloudFormation ressources AWS sont créées à partir des constructions du kit AWS Cloud Development Kit (AWS CDK). Si vous avez déjà déployé cette solution, consultez Mettre à jour la solution pour obtenir des instructions de mise à jour.

Lancement de la pile

Suivez ces étapes pour déployer la solution Distributed Load Testing on AWS sur votre compte.

> **Note**
>
> Cette solution inclut des métriques de collecte de données pour AWS. Nous utilisons ces données pour mieux comprendre la façon dont les clients utilisent cette solution et les services et produits associés. AWS est propriétaire des données collectées dans le cadre de cette enquête. La collecte de données est soumise à l'avis de confidentialité d'AWS.

Ce CloudFormation modèle AWS automatisé déploie des tests de charge distribués sur AWS.

> **Note**
>
> Vous êtes responsable du coût des services AWS utilisés lors de l'exécution de cette solution. Pour plus de détails, consultez la section Coût de ce guide et consultez la page Web de tarification de chaque service AWS utilisé dans cette solution.

1. Connectez-vous à l'AWS Management Console et sélectionnez le bouton pour lancer le CloudFormation modèle.

   Vous pouvez également télécharger le modèle comme point de départ pour votre propre implémentation.
2. Le modèle est lancé dans la région USA Est (Virginie du Nord) par défaut. Pour lancer cette solution dans une autre région AWS, utilisez le sélecteur de région dans la barre de navigation de la console.

   > **Note**
   >
   > Cette solution utilise Amazon Cognito, qui n'est actuellement disponible que dans certaines régions AWS. Par conséquent, vous devez lancer cette solution dans une région AWS où

Amazon Cognito est disponible. Pour connaître la disponibilité des services la plus récente par région, consultez la liste des services régionaux AWS.

3. Sur la page Create stack, vérifiez que l'URL du modèle correct apparaît dans la zone de texte URL Amazon S3 et choisissez Next.
4. Sur la page Spécifier les détails de la pile, attribuez un nom à votre pile de solutions.
5. Sous Paramètres, passez en revue les paramètres du modèle et modifiez-les si nécessaire. Cette solution utilise les valeurs par défaut suivantes.

Paramètre	Par défaut	Description
Nom de l'administrateur	<Entrée obligatoire>	Nom d'utilisateur de l'administrateur de la solution initiale.
Adresse e-mail de l'administrateur	*<Requires input>*	Adresse e-mail de l'utilisateur administrateur. Après le lancement, un e-mail contenant les instructions de connexion à la console sera envoyé à cette adresse.
ID VPC existant	<Optional input>	Si vous souhaitez utiliser un VPC déjà créé, entrez l'ID d'un VPC existant dans la même région où la pile a été déployée. Par exemple, vpc-1a2b3c4d5e6f.
Premier sous-réseau existant	<Optional input>	L'ID du premier sous-réseau de votre VPC existant. Ce sous-réseau a besoin d'une route vers Internet pour extraire l'image du conteneur afin d'exécuter des tests. Par exemple, subnet-7h8i9j0k.

Paramètre	Par défaut	Description
Deuxième sous-réseau existant	<Optional input>	L'ID du deuxième sous-rése au au sein du VPC existant. Ce sous-réseau a besoin d'une route vers Internet pour extraire l'image du conteneur afin d'exécuter des tests. Par exemple, subnet-1x2y3z.
Fournir un bloc CIDR valide pour que la solution crée un VPC	192.168.0.0/16	Vous pouvez laisser ce paramètre vide si vous utilisez un VPC existant
Fournissez un bloc d'adresse CIDR valide pour le sous-réseau A afin que la solution crée un VPC	192.168.0.0/20	Bloc CIDR pour le sous-rése au A du VPC AWS Fargate
Fournissez un bloc d'adresse CIDR valide pour le sous-réseau B afin que la solution crée un VPC	192,168.16,0/20	Bloc CIDR pour le sous-rése au B du VPC AWS Fargate
Fournir un bloc CIDR pour autoriser le trafic sortant des tâches Fargate	0.0.0.0/0	Bloc CIDR qui restreint l'accès sortant aux conteneur s Amazon ECS.
Mise à jour automatique de l'image du conteneur	No	Utilisez automatiquement l'image la plus récente et la plus sécurisée jusqu'à la prochaine version mineure. La No sélection permet d'extraire l'image telle qu'elle a été publiée à l'origine, sans aucune mise à jour de sécurité.

Paramètre	Par défaut	Description
Déployer un serveur MCP en option	No	Déployez le serveur MCP distant en option, en utilisant AgentCore Gateway pour connecter les applications d'IA aux tests de charge distribués sur AWS.

6. Choisissez Next (Suivant).
7. Sur la page Configurer les options de pile, choisissez Suivant.
8. Sur la page Vérification, vérifiez et confirmez les paramètres. Cochez la case indiquant que le modèle créera des ressources AWS Identity and Access Management (IAM).
9. Sélectionnez Create stack (Créer une pile) pour déployer la pile.

Vous pouvez consulter l'état de la pile dans la CloudFormation console AWS dans la colonne Status. Vous devriez recevoir le statut CREATE_COMPLETE dans 15 minutes environ.

> **Note**
>
> Outre la fonction principale d'AWS Lambda, cette solution inclut la fonction Lambda de ressources personnalisées, qui s'exécute uniquement lors de la configuration initiale ou lorsque les ressources sont mises à jour ou supprimées.

Lors de l'exécution de cette solution, la fonction Lambda des ressources personnalisées est inactive. Toutefois, ne supprimez pas cette fonction car elle est nécessaire pour gérer les ressources associées.

Déploiement multirégional

Temps de déploiement : environ 5 minutes par région

Vous pouvez effectuer des tests dans plusieurs régions. Lorsque vous déployez la solution de test de charge distribué, elle crée un CloudFormation modèle régional et le stocke dans le compartiment S3 des scénarios.

> **Note**
>
> Le nom du bucket scenarios inclut le nom de votre stack et le mot clé « scenarios ». Vous pouvez le localiser en accédant à la console S3 et en recherchant des compartiments contenant des « scénarios » dans le nom.

Pour exécuter un déploiement multirégional, vous devez déployer le CloudFormation modèle régional, qui est stocké dans le compartiment des scénarios Amazon S3, dans les régions où vous souhaitez exécuter le test. Vous pouvez installer le modèle régional en procédant comme suit :

1. Dans la console Web de la solution, accédez à Tableau de bord dans le menu de gauche.
2. Utilisez l'icône du presse-papiers pour copier le lien du CloudFormation modèle dans Amazon S3.
3. Connectez-vous à la CloudFormation console AWS et sélectionnez la bonne région.
4. Sur la page Create stack, vérifiez que l'URL du modèle correct apparaît dans la zone de texte URL Amazon S3 et choisissez Next.
5. Sur la page Spécifier les détails de la pile, attribuez un nom à votre pile de solutions.
6. Sous Paramètres, passez en revue les paramètres du modèle et modifiez-les si nécessaire. Cette solution utilise les valeurs par défaut suivantes.

Paramètre	Par défaut	Description
ID VPC existant	<Optional input>	Si vous souhaitez utiliser un VPC déjà créé, entrez l'ID d'un VPC existant dans la même région où la pile a été déployée. Par exemple, vpc-1a2b3c4d5e6f.
Premier sous-réseau existant	<Optional input>	L'ID du premier sous-rése au de votre VPC existant. Ce sous-réseau a besoin d'une route vers Internet pour extraire l'image du conteneur afin d'exécuter des tests. Par exemple, subnet-7h8i9j0k.

Paramètre	Par défaut	Description
Deuxième sous-réseau existant	<Optional input>	L'ID du deuxième sous-rése au au sein du VPC existant. Ce sous-réseau a besoin d'une route vers Internet pour extraire l'image du conteneur afin d'exécuter des tests. Par exemple, subnet-1x2y3z.
Fournir un bloc CIDR valide pour que la solution crée un VPC	192.168.0.0/16	Si vous ne fournissez aucune valeur pour un VPC existant, le bloc CIDR du VPC Amazon créé par la solution contient l'adresse IP d'AWS Fargate.
Fournir un bloc CIDR pour autoriser le trafic sortant des tâches Fargate	0.0.0.0/0	Bloc CIDR qui restreint l'accès sortant aux conteneur s Amazon ECS.

7. Choisissez Next (Suivant).
8. Sur la page Configurer les options de pile, choisissez Suivant.
9. Sur la page Vérification, vérifiez et confirmez les paramètres. Assurez-vous de cocher la case indiquant que le modèle créera des ressources AWS Identity and Access Management (IAM).
10. Sélectionnez Create stack (Créer une pile) pour déployer la pile.

    Vous pouvez consulter l'état de la pile dans la CloudFormation console AWS dans la colonne Status. Vous devriez recevoir le statut CREATE_COMPLETE dans environ cinq minutes.

Lorsque les régions ont été déployées avec succès, elles apparaissent dans la console Web. Lorsque vous créez un test, toutes les régions disponibles sont répertoriées sur le tableau de bord et dans Création de scénarios. Vous pouvez ajouter une région à un test à l'étape Traffic Shape de Scenario Creation.

La solution crée un élément DynamoDB pour chaque région déployée dans le tableau des scénarios, qui contient les informations nécessaires sur les ressources de test de cette région. Vous pouvez trier les résultats des tests par région dans la console Web. Pour afficher les résultats agrégés de toutes

les régions dans le cadre d'un test multirégional, utilisez CloudWatch les métriques Amazon. Vous trouverez le code source du graphique dans les résultats du test une fois celui-ci terminé.

> **Note**
>
> Vous pouvez lancer la pile régionale sans la console Web. Obtenez un lien vers le modèle régional dans le compartiment des scénarios Amazon S3 et fournissez-le comme source lors du lancement de la pile régionale dans la région requise. Vous pouvez également télécharger le modèle et le télécharger en tant que source pour la région de votre choix.

Mettre à jour la solution

La mise à jour de la solution applique les dernières fonctionnalités, correctifs de sécurité et corrections de bogues à votre déploiement. Si vous avez déjà déployé la solution, suivez cette procédure pour mettre à jour la CloudFormation pile avec la dernière version.

> **Important**
>
> Avant de procéder à la mise à jour, assurez-vous qu'aucun test de charge n'est en cours d'exécution. Le processus de mise à jour peut temporairement perturber la disponibilité de la solution.

1. Connectez-vous à la CloudFormation console, sélectionnez votre CloudFormation pile existante, puis sélectionnez Mettre à jour la pile.
2. Sélectionnez Effectuer une mise à jour directe.
3. Sélectionnez Remplacer le modèle existant.
4. Sous Spécifier le modèle :
   a. Sélectionnez l'URL Amazon S3.
   b. Copiez le lien du dernier modèle.
   c. Collez le lien dans le champ URL d'Amazon S3.
   d. Vérifiez que l'URL du modèle s'affiche correctement dans la zone de texte URL Amazon S3.
   e. Choisissez Suivant.
   f. Choisissez Suivant à nouveau.
5. Sous Paramètres, passez en revue les paramètres du modèle et modifiez-les si nécessaire. Reportez-vous à la section Lancer la pile pour plus de détails sur les paramètres.
6. Choisissez Next (Suivant).
7. Sur la page Configurer les options de pile, choisissez Suivant.
8. Sur la page Vérification, vérifiez et confirmez les paramètres.
9. Cochez la case indiquant que le modèle est susceptible de créer des ressources IAM.
10. Choisissez Afficher l'ensemble de modifications et vérifiez les modifications.
11. Choisissez Mettre à jour la pile pour déployer la pile.

Vous pouvez consulter l'état de la pile dans la CloudFormation console AWS dans la colonne Status. Vous devriez recevoir un `UPDATE_COMPLETE` statut dans 15 minutes environ.

> **Note**
>
> Si vous rencontrez des problèmes d'authentification avec Amazon Cognito lorsque vous vous connectez depuis votre navigateur après la mise à niveau de Stack, actualisez votre navigateur (Ctrl+Shift+R activé Windows/Linux ou Cmd+Shift+R sur Mac) pour effacer les données mises en cache et réessayez.

Résolution des problèmes liés aux mises à jour à partir de versions antérieures à la v3.3.0

> **Note**
>
> Cette section s'applique uniquement aux mises à jour des versions antérieures à la v3.3.0. Si vous effectuez une mise à jour à partir de la version 3.3.0 ou ultérieure, suivez la procédure de mise à jour standard ci-dessus.

1. Téléchargez le fichier distributed-load-testing-on-aws.template.
2. Ouvrez le modèle et accédez à Conditions : et recherchez DLTCommon ResourcesAppRegistryCondition
3. Le résultat devrait être similaire à ce qui suit :

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

4. Remplacez la deuxième valeur vraie par fausse :

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

```
- "false"
```

5. Utilisez le modèle personnalisé pour mettre à jour votre stack.
6. Cette mise à jour supprime les ressources liées au registre des applications de la pile, ce qui permet à la mise à jour de se terminer correctement.
7. Effectuez une autre mise à jour de la pile à l'aide de l'URL du modèle le plus récent pour ajouter à nouveau les ressources de l'application du registre des applications à votre pile.

Mise à jour des stacks régionaux

Si vous avez déployé la solution dans plusieurs régions, vous devez mettre à jour chaque stack régional séparément. Suivez la procédure de mise à jour standard pour chaque CloudFormation stack régional dans les régions où vous avez déployé une infrastructure de test.

Gestionnaire d'applications AWS Systems Manager

Après la mise à jour de la solution, AWS Systems Manager Application Manager fournit une vue d'ensemble de la solution et de ses ressources au niveau de l'application. Vous pouvez utiliser le Gestionnaire d'applications pour :

- Surveillez les ressources, les coûts des ressources déployées sur les stacks et les comptes AWS, ainsi que les journaux depuis un emplacement central.
- Affichez les données opérationnelles relatives aux ressources de la solution dans le contexte d'une application, telles que l'état du déploiement, les CloudWatch alarmes, les configurations des ressources et les problèmes opérationnels.

Résolution des problèmes

La résolution des problèmes connus fournit des instructions pour atténuer les erreurs connues. Si ces instructions ne répondent pas à votre problème, contactez le support AWS fournit des instructions pour ouvrir un dossier de support AWS pour cette solution.

Résolution des problèmes connus

Problème : vous utilisez un VPC existant et vos tests échouent avec le statut Echoué, ce qui entraîne le message d'erreur suivant :

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

- Résolution :

Assurez-vous que les sous-réseaux existent dans le VPC spécifié et qu'ils disposent d'une route vers Internet via une passerelle Internet ou une passerelle NAT. AWS Fargate a besoin d'un accès pour extraire l'image du conteneur depuis le référentiel public afin d'exécuter les tests avec succès.

Problème : les tests prennent trop de temps ou sont bloqués indéfiniment

- Résolution :

Annulez le test et vérifiez AWS Fargate pour vous assurer que toutes les tâches ont été arrêtées. Si elles ne se sont pas arrêtées, arrêtez manuellement toutes les tâches Fargate. Vérifiez les limites de tâches Fargate à la demande sur votre compte pour vous assurer que vous pouvez lancer le nombre de tâches souhaité. Vous pouvez également consulter les CloudWatch journaux de la fonction Lambda Task-Runner pour mieux comprendre les défaillances lors du lancement de tâches Fargate. Consultez les journaux CloudWatch ECS pour plus de détails sur ce qui se passe dans les conteneurs Fargate en cours d'exécution.

Problème : les tests démarrent mais ne se terminent pas ou l'état des tâches ECS est inconnu

- Résolution :

Si vous avez sélectionné l'option permettant de fournir un VPC existant dans le compte sur lequel la solution a été déployée, assurez-vous que le VPC utilisé par les tâches ECS possède suffisamment

d'adresses IP libres pour démarrer le nombre de tâches indiqué dans l'entrée de test. La définition des tâches ECS utilise l'image ECR qui nécessite une passerelle Internet ou une route vers Internet afin que le service ECS puisse fournir les tâches en téléchargeant l'image ECR de la solution depuis distributed-load-testing-onaws-solutions/ -. aws-load-tester Si vous ne pouvez pas fournir de route vers Internet étant donné que tous les sous-réseaux du VPC sont privés, vous pouvez héberger l'image ECR dans votre compte à l'aide du cache d'extraction ECR. Mettez à jour la définition de la tâche avec le nouvel URI de l'image ECR et créez une nouvelle révision. Une fois la définition de tâche mise à jour, la configuration de la solution dans la table DynamoDB doit être mise à jour pour utiliser la nouvelle révision. Le nom de la table DynamoDB se trouve dans l'onglet Stack Outputs sous CloudFormation la clé. ScenariosTable Mettez à jour l'attribut TaskDefinition pour l'élément avec la clé TestID et la valeur region- [SOLUTION-DEPLOYED-REGION].

Problème : les tests doivent utiliser un point de terminaison privé ou non disponible via la passerelle Internet

- Résolution :

Lorsque vous testez des points de terminaison d'API privés qui ne sont pas accessibles via la passerelle Internet, envisagez les approches suivantes :

1. Configuration réseau : assurez-vous que les tables de routage de sous-réseau utilisées par les tâches ECS sont mises à jour avec une route vers la plage d'adresses IP du point de terminaison privé testé. Cela permet au trafic de test d'atteindre le point de terminaison privé au sein de votre VPC.
2. Résolution DNS : pour les domaines personnalisés, configurez les paramètres DNS de votre VPC pour résoudre le nom de domaine du point de terminaison privé. Reportez-vous à la documentation DNS VPC pour obtenir des instructions détaillées.
3. Points de terminaison VPC : si vous testez les services AWS, envisagez d'utiliser des points de terminaison VPC (PrivateLinkAWS) pour établir une connectivité privée. Par exemple, pour tester une API Gateway privée, vous pouvez créer un point de terminaison VPC pour API Gateway. Consultez la documentation de Private API Gateway.
4. Peering VPC : si le point de terminaison privé se trouve dans un autre VPC, établissez le peering VPC entre le VPC où la solution est déployée et le VPC contenant le point de terminaison privé. Configurez les tables de routage appropriées dans les deux cas VPCs. Consultez la documentation VPC Peering.

5. Transit Gateway : pour les scénarios de mise en réseau plus complexes impliquant plusieurs VPCs utilisateurs, pensez à utiliser AWS Transit Gateway pour acheminer le trafic entre le VPC de la solution et le VPC contenant le point de terminaison privé. Consultez la documentation de Transit Gateway.
6. Groupes de sécurité : assurez-vous que les groupes de sécurité associés à vos tâches ECS autorisent le trafic sortant vers le point de terminaison privé et que les groupes de sécurité du point de terminaison privé autorisent le trafic entrant provenant des tâches ECS.

Pour tester des EC2 instances ou des équilibreurs de charge d'application internes, assurez-vous que les plages d'adresses CIDR VPC ne se chevauchent pas et que les routes nécessaires sont configurées dans les tables de routage.

Problème : les tests sont terminés mais les résultats ne sont pas disponibles dans l'interface utilisateur

- Résolution :

Si le test est terminé mais que les résultats ne sont pas disponibles dans l'interface utilisateur, les fichiers de résultats devraient toujours être disponibles dans le compartiment S3 à partir des tâches ECS qui ont effectué les tests. Il s'agit d'une limite connue de la solution. Dans l'architecture actuelle, la solution utilise une fonction Lambda d'analyse des résultats pour résumer les résultats de plusieurs tâches ECS, qui sont ensuite stockés sous forme d'élément dans la table DynamoDB. La taille maximale des éléments de la table DynamoDB est limitée à 400 Ko. Cette limite est atteinte en fonction de la complexité du script de test, de la simultanéité et du nombre de tâches utilisées. L'erreur ne signifie pas que le test échoue ; elle indique que le processus de synthèse des résultats et de leur stockage dans la table DynamoDB pour les opérations CRUD a échoué. Les résultats sont toujours disponibles dans le compartiment S3 pour le scénario de test.

Contacter AWS Support

Si vous bénéficiez d'AWS Developer Support, d'AWS Business Support ou d'AWS Enterprise Support, vous pouvez utiliser le Centre de support pour obtenir l'assistance d'experts concernant cette solution. Les sections suivantes fournissent des instructions.

Créer un dossier

1. Connectez-vous au Centre de Support.

2. Choisissez Create case (Créer une demande).

Comment pouvons-nous vous aider ?

1. Choisissez Technique
2. Pour Service, sélectionnez Solutions.
3. Dans Catégorie, sélectionnez Tests de charge distribués sur AWS.
4. Pour Severity, sélectionnez l'option qui correspond le mieux à votre cas d'utilisation.
5. Lorsque vous entrez le service, la catégorie et la gravité, l'interface contient des liens vers des questions de dépannage courantes. Si vous ne parvenez pas à résoudre vos questions à l'aide de ces liens, sélectionnez Étape suivante : Informations supplémentaires.

Informations supplémentaires

1. Dans le champ Objet, saisissez un texte résumant votre question ou problème.
2. Pour la description, décrivez le problème en détail.
3. Choisissez Joindre des fichiers.
4. Joignez les informations dont AWS Support a besoin pour traiter la demande.

Aidez-nous à résoudre votre cas plus rapidement

1. Entrez les informations demandées.
2. Choisissez Next step: Solve now or contact us (Étape suivante : résolvez maintenant ou contactez-nous).

Résolvez maintenant ou contactez-nous

1. Passez en revue les solutions Solve now.
2. Si vous ne parvenez pas à résoudre votre problème avec ces solutions, choisissez Contactez-nous, entrez les informations demandées, puis choisissez Soumettre.

Désinstallez la solution

Vous pouvez désinstaller la solution de test de charge distribué sur AWS depuis la console de gestion AWS ou en utilisant l'interface de ligne de commande AWS. Vous devez supprimer manuellement la console, le scénario et les compartiments de journalisation Amazon Simple Storage Service (Amazon S3) créés par cette solution. Les implémentations de solutions AWS ne les suppriment pas automatiquement au cas où vous auriez des données à conserver.

> **Note**
>
> Si vous avez déployé des piles régionales, vous devez supprimer les piles de ces régions avant de supprimer la pile principale.

Utilisation de la AWS Management Console

1. Connectez-vous à la CloudFormation console AWS.
2. Sur la page Stacks, sélectionnez la pile d'installation de cette solution.
3. Sélectionnez Delete (Supprimer).

Utilisation de l'interface de ligne de commande AWS

Déterminez si l'interface de ligne de commande AWS (AWS CLI) est disponible dans votre environnement. Pour les instructions d'installation, reportez-vous à la section Qu'est-ce que l'interface de ligne de commande AWS dans le guide de l'utilisateur de l'AWS CLI. Après avoir confirmé que l'AWS CLI est disponible, exécutez la commande suivante.

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

Supprimer les compartiments Amazon S3

Cette solution est configurée pour conserver les compartiments Amazon S3 créés par la solution (à déployer dans une région optionnelle) si vous décidez de supprimer la CloudFormation pile AWS afin d'éviter toute perte de données accidentelle. Après avoir désinstallé la solution, vous pouvez

supprimer manuellement ce compartiment S3 si vous n'avez pas besoin de conserver les données. Suivez ces étapes pour supprimer le compartiment Amazon S3.

1. Connectez-vous à la console Amazon S3.
2. Choisissez Buckets dans le volet de navigation de gauche.
3. Dans le champ Rechercher des compartiments par nom, entrez le nom de la pile de cette solution.
4. Sélectionnez l'un des compartiments S3 de la solution et choisissez Empty.
5. Entrez Supprimer définitivement dans le champ de vérification et choisissez Vide.
6. Sélectionnez le compartiment S3 que vous venez de vider et choisissez Supprimer.
7. Entrez le nom du compartiment S3 dans le champ de vérification et choisissez Supprimer le compartiment.

Répétez les étapes 4 à 7 jusqu'à ce que vous supprimiez tous les compartiments S3.

Pour supprimer le compartiment S3 à l'aide de l'AWS CLI, exécutez la commande suivante :

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

Utilisez la solution

Cette section fournit un guide complet sur l'utilisation de la solution de test de charge distribué sur AWS, de la création de votre premier scénario de test à l'analyse des résultats détaillés. Le flux de travail inclut la création d'un scénario de test, l'exécution d'un test et l'exploration des résultats des tests.

Création d'un scénario de test

La création d'un scénario de test implique quatre étapes principales : configurer les paramètres généraux, définir le scénario, définir les modèles de trafic et revoir votre configuration.

Étape 1 : Paramètres généraux

Configurez les paramètres de base de votre test de charge, notamment le nom du test, la description et les options de configuration générales.

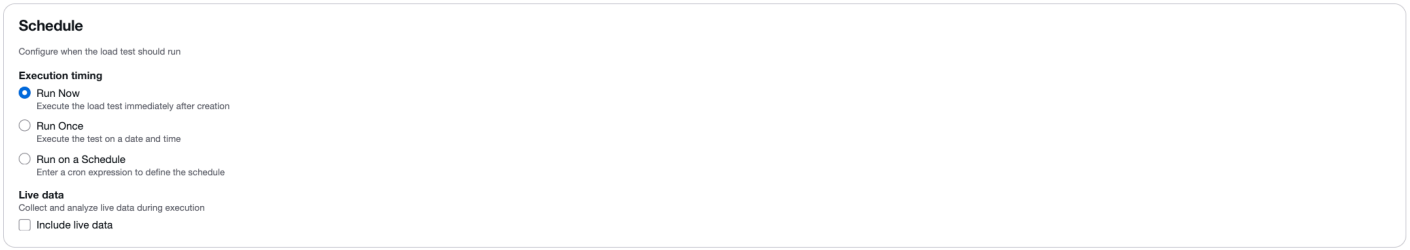
Identification du test

- Nom du test (obligatoire) : nom descriptif de votre scénario de test
- Description du test (obligatoire) - Détails supplémentaires sur l'objectif et la configuration du test
- Balises (facultatif) - Ajoutez jusqu'à 5 balises pour classer et organiser vos scénarios de test

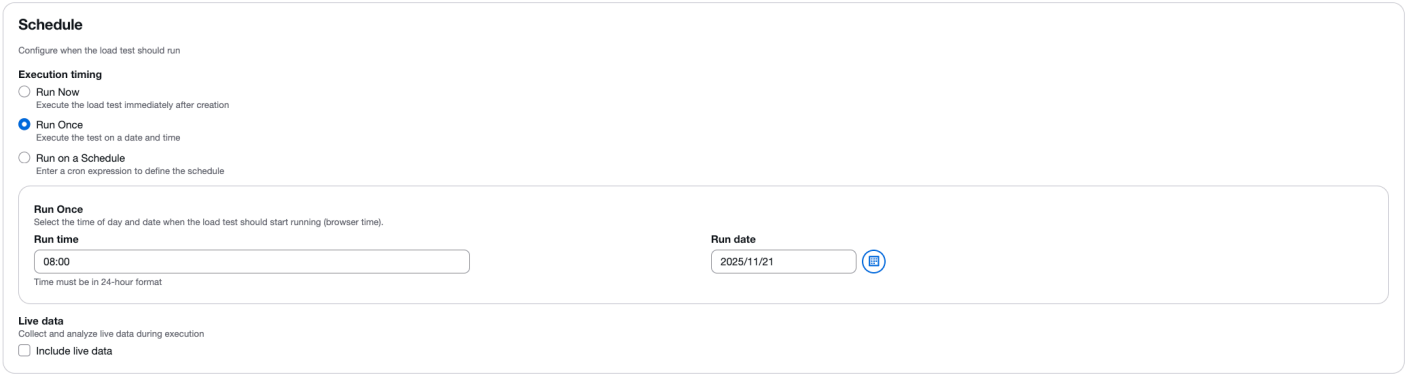
Options de planification

Configurez le moment où le test doit être exécuté :

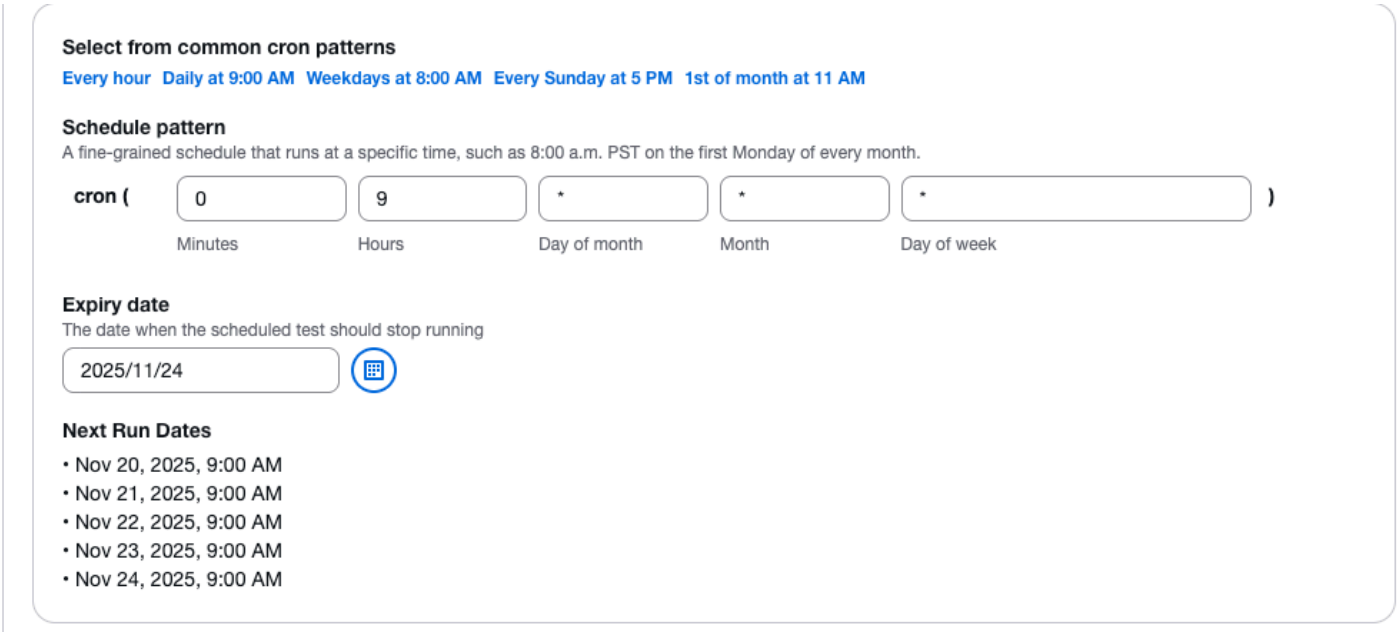
- Exécuter maintenant : exécute le test immédiatement après sa création.



- Exécuter une fois : planifiez le test pour qu'il soit exécuté à une date et à une heure spécifiques.



- Exécuter selon un calendrier : utilisez la planification basée sur le cron pour exécuter des tests automatiquement à intervalles réguliers. Vous pouvez sélectionner des modèles courants (toutes les heures, tous les jours, toutes les semaines) ou définir une expression cron personnalisée.



Flux de travail de planification

Lorsque vous planifiez un test, le flux de travail suivant se produit :

- Les paramètres de planification sont envoyés à l'API de la solution via Amazon API Gateway.
- L'API transmet les paramètres à une fonction Lambda qui crée une règle d' CloudWatch événements dont l'exécution est planifiée à la date spécifiée.
- Pour les tests ponctuels (Run Once), la règle CloudWatch Events s'exécute à la date spécifiée et la fonction `api-services` Lambda exécute le test.

- Pour les tests récurrents (Exécuter selon un calendrier), la règle CloudWatch Events s'active à la date spécifiée, et la fonction `api-services` Lambda crée une nouvelle règle qui s'exécute immédiatement et de manière récurrente en fonction de la fréquence spécifiée.

Données en direct

Cochez la case Inclure les données en temps réel pour afficher les métriques en temps réel pendant l'exécution de votre test. Lorsque cette option est activée, vous pouvez surveiller :

- Temps de réponse moyen
- Nombre d'utilisateurs virtuels.
- Les demandes réussies comptent.
- Les demandes ayant échoué sont comptabilisées.

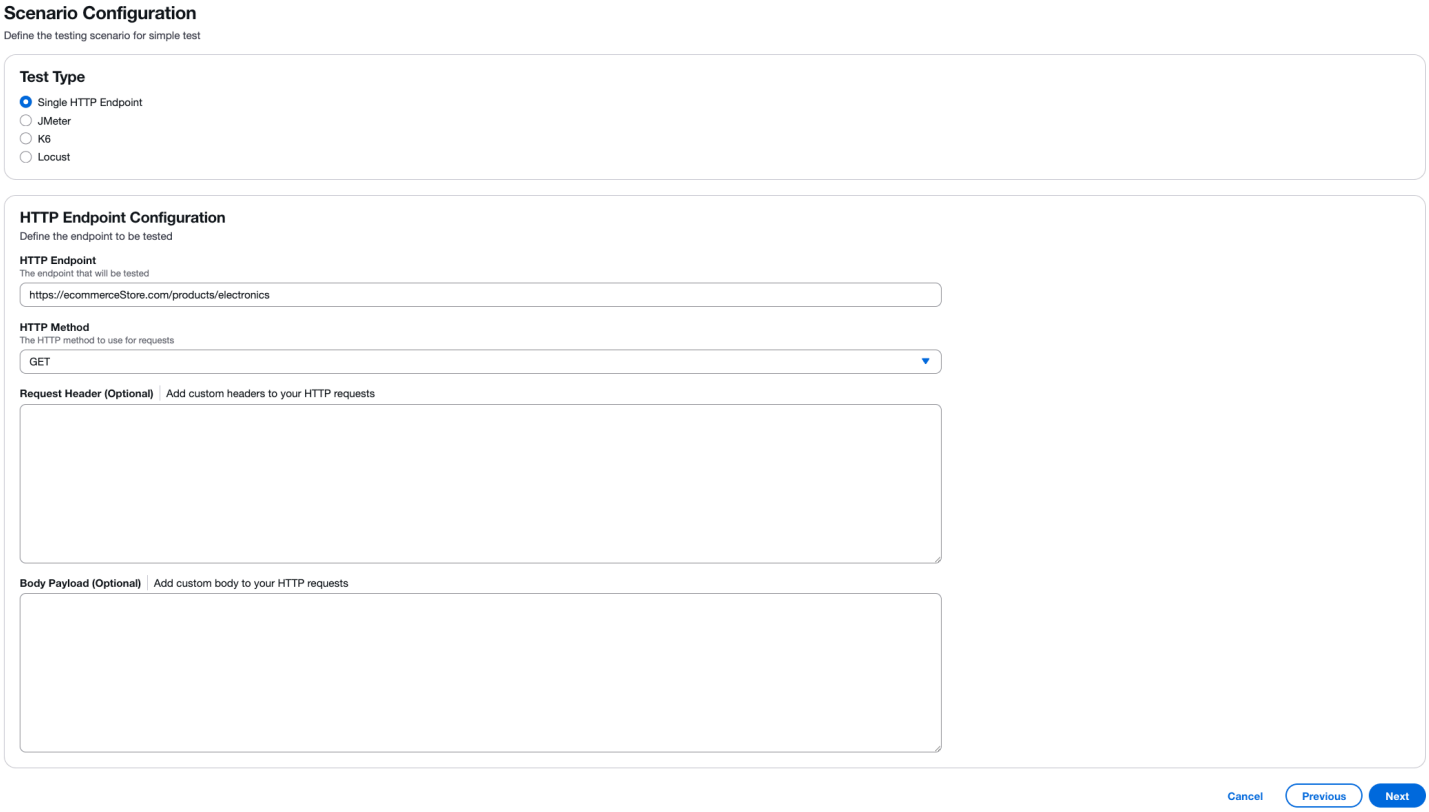
La fonction de données en temps réel fournit des graphiques en temps réel avec des données agrégées à intervalles d'une seconde. Pour plus d'informations, reportez-vous à la section Surveillance à l'aide de données en temps réel.

Étape 2 : Configuration du scénario

Définissez le scénario de test spécifique et sélectionnez votre framework de test préféré.

Sélection du type de test

Choisissez le type de test de charge que vous souhaitez effectuer :



- Point de terminaison HTTP unique : testez un point de terminaison d'API ou une page Web unique avec une configuration simple.
- JMeter- Téléchargez des scripts de JMeter test (fichiers .jmx ou archives .zip).
- K6 - Téléchargez des scripts de test K6 (fichiers .js ou archives .zip).
- Locust - Téléchargez des scripts de test Locust (fichiers .py ou archives .zip).

Configuration du point de terminaison HTTP image : :images/test-types.png [Sélectionnez le type de test à exécuter] Lorsque « Point de terminaison HTTP unique » est sélectionné, configurez les paramètres suivants :

Point de terminaison HTTP (obligatoire)

Entrez l'URL complète du point de terminaison que vous souhaitez tester. Par exemple, `https://api.example.com/users` . Assurez-vous que le point de terminaison est accessible depuis l'infrastructure AWS.

Méthode HTTP (obligatoire)

Sélectionnez la méthode HTTP pour vos requêtes. La valeur par défaut est `GET`. Les autres options incluent `POST` `PUT`,`DELETE`,`PATCH`,`HEAD`, et`OPTIONS`.

En-tête de demande (facultatif)

Ajoutez des en-têtes HTTP personnalisés à vos demandes. Les exemples les plus courants incluent :

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

Choisissez Ajouter un en-tête pour inclure plusieurs en-têtes.

Charge utile du corps (en option)

Ajoutez le contenu du corps de la requête pour les requêtes POST ou PUT. Supporte les formats JSON, XML ou texte brut. Par exemple : `{"userId": 123, "action": "test"}`.

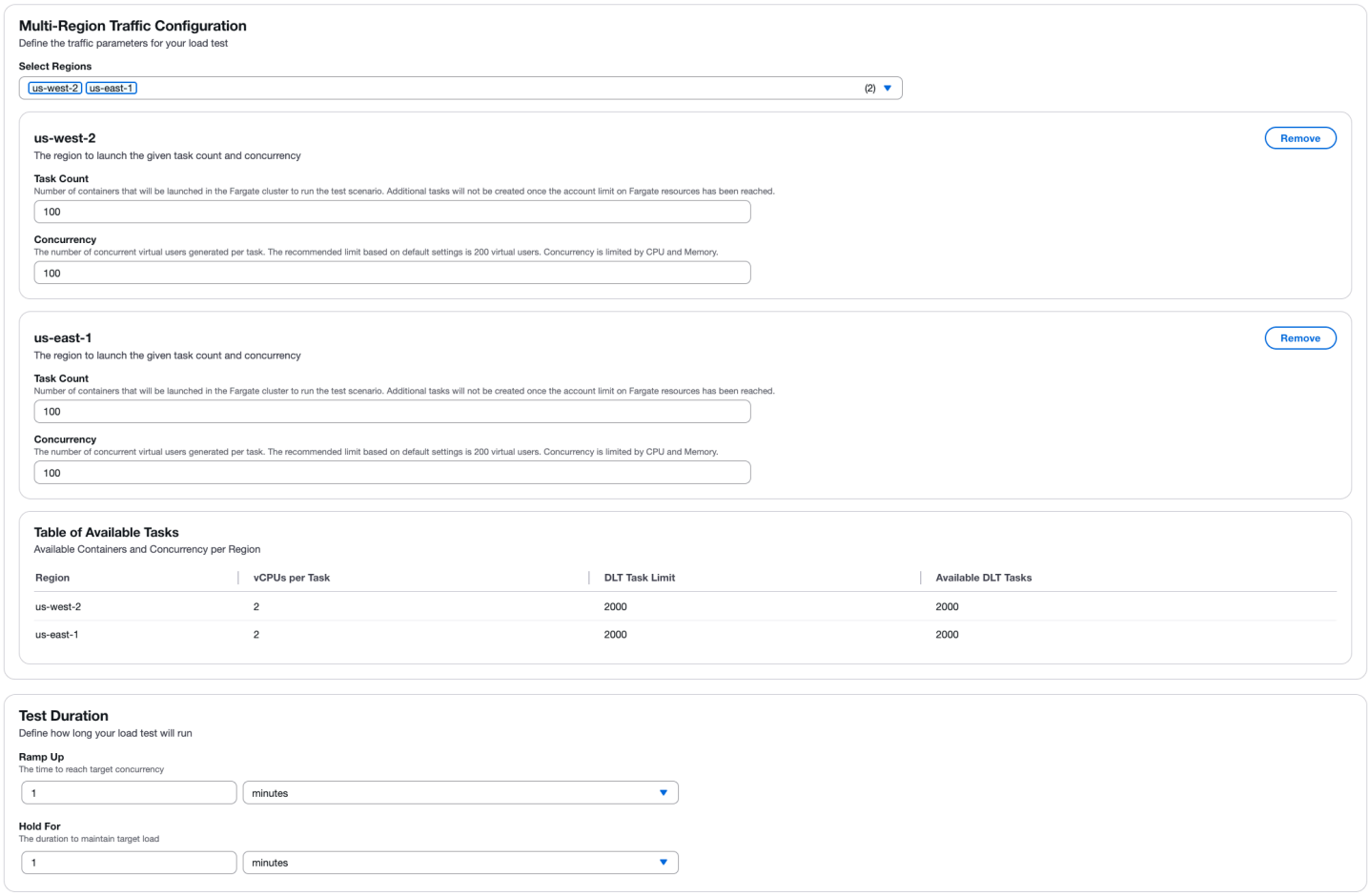
Scripts du framework de test

Lorsque vous utilisez JMeter K6 ou Locust, téléchargez votre fichier de script de test ou une archive .zip contenant votre script de test et les fichiers de support. En JMeter effet, vous pouvez inclure des plugins personnalisés dans un `/plugins` dossier de votre archive .zip.

> **Important**
>
> Bien que votre script de test (JMeterK6 ou Locust) puisse définir la simultanéité (utilisateurs virtuels), les taux de transaction (TPS), les temps de montée en puissance et d'autres paramètres de chargement, la solution remplacera ces configurations par les valeurs que vous spécifiez sur l'écran Traffic Shape lors de la création du test. La configuration Traffic Shape contrôle le nombre de tâches, la simultanéité (utilisateurs virtuels par tâche), la durée de montée en puissance et la durée de suspension pour l'exécution du test.

Étape 3 : Forme du trafic

Configurez la manière dont le trafic sera distribué pendant votre test, y compris le support multirégional.



Configuration du trafic multirégional

Sélectionnez une ou plusieurs régions AWS pour répartir géographiquement votre test de charge. Pour chaque région sélectionnée, configurez :

Nombre de tâches

> Le nombre de conteneurs (tâches) qui seront lancés dans le cluster Fargate pour le scénario de test. Aucune tâche supplémentaire ne sera créée une fois que le compte aura atteint la limite « La ressource Fargate a été atteinte ».

Simultanéité

> Le nombre d'utilisateurs virtuels générés simultanément par tâche. La limite recommandée est basée sur les paramètres par défaut de 2 V CPUs par tâche. La simultanéité est limitée par les ressources du processeur et de la mémoire.

Déterminer le nombre d'utilisateurs

Le nombre d'utilisateurs qu'un conteneur peut prendre en charge pour un test peut être déterminé en augmentant progressivement le nombre d'utilisateurs et en surveillant les performances sur Amazon CloudWatch. Une fois que vous avez constaté que les performances du processeur et de la mémoire approchent de leurs limites, vous avez atteint le nombre maximum d'utilisateurs qu'un conteneur peut prendre en charge pour ce test dans sa configuration par défaut (2 vCPU et 4 Go de mémoire).

Processus d'étalonnage

Vous pouvez commencer à déterminer les limites d'utilisateurs simultanés pour votre test en utilisant l'exemple suivant :

1. Créez un test avec un maximum de 200 utilisateurs.
2. Pendant le test, surveillez le processeur et la mémoire à l'aide de la CloudWatch console :
   a. Dans le volet de navigation de gauche, sous Container Insights, sélectionnez Performance Monitoring.
   b. Sur la page Surveillance des performances, dans le menu déroulant de gauche, sélectionnez ECS Clusters.
   c. Dans le menu déroulant de droite, sélectionnez votre cluster Amazon Elastic Container Service (Amazon ECS).
3. Pendant la surveillance, surveillez le processeur et la mémoire. Si le processeur ne dépasse pas 75 % ou si la mémoire ne dépasse pas 85 % (ignorez les pics ponctuels), vous pouvez exécuter un autre test avec un plus grand nombre d'utilisateurs.

Répétez les étapes 1 à 3 si le test n'a pas dépassé les limites de ressources. Vous pouvez éventuellement augmenter les ressources du conteneur pour permettre un plus grand nombre d'utilisateurs simultanés. Cela entraîne toutefois un coût plus élevé. Pour plus de détails, consultez le guide du développeur.

> **Note**
>
> Pour obtenir des résultats précis, n'exécutez qu'un seul test à la fois pour déterminer les limites d'utilisateurs simultanés. Tous les tests utilisent le même cluster, et CloudWatch Container Insights agrège les données de performance en fonction du cluster. Les deux tests sont donc signalés simultanément à CloudWatch Container Insights, ce qui entraîne des mesures d'utilisation des ressources inexactes pour un seul test.

Pour plus d'informations sur le calibrage des utilisateurs par moteur, reportez-vous à la section Étalonnage d'un test Taurus dans la documentation. BlazeMeter

> **Note**
>
> La solution affiche les informations de capacité disponibles pour chaque région, ce qui vous aide à planifier votre configuration de test dans les limites disponibles.

Tableau des tâches disponibles

Le tableau des tâches disponibles indique la disponibilité des ressources pour chaque région sélectionnée :

- Région : nom de la région AWS.
- v CPUs par tâche : nombre de machines virtuelles CPUs allouées à chaque tâche (par défaut : 2).
- Limite de tâches DLT : nombre maximum de tâches pouvant être créées en fonction des limites Fargate de votre compte (par défaut : 2000).
- Tâches DLT disponibles : nombre actuel de tâches disponibles dans la région (par défaut : 2000).

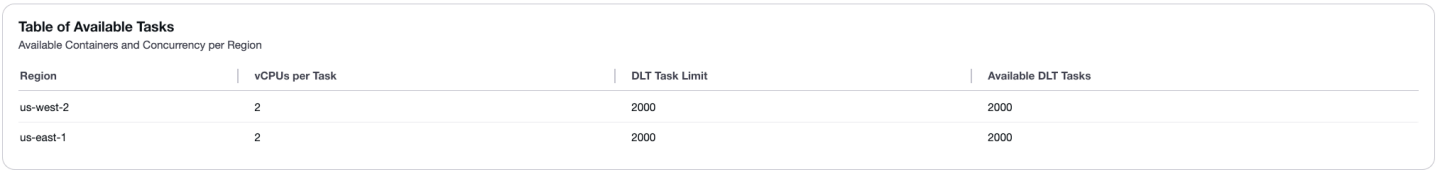


Table of Available Tasks

Available Containers and Concurrency per Region

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

Pour augmenter le nombre de tâches disponibles ou v CPUs par tâche, consultez le guide du développeur.

Durée du test

Définissez la durée de votre test de charge :

Passez à la vitesse supérieure

Le temps nécessaire pour atteindre la simultanéité cible. La charge augmente progressivement de 0 au niveau de simultanéité configuré au cours de cette période.

Maintenez pour

Durée nécessaire pour maintenir la charge cible. Le test se poursuit en simultané pendant cette période.

Étape 4 : vérifier et créer

Passez en revue toutes vos configurations avant de créer le scénario de test. Vérifier:

- Réglages généraux (nom, description, calendrier).
- Configuration du scénario (type de test, point de terminaison ou script).
- Forme du trafic (tâches, utilisateurs, durée, régions).

Après examen, choisissez Créer pour enregistrer votre scénario de test.

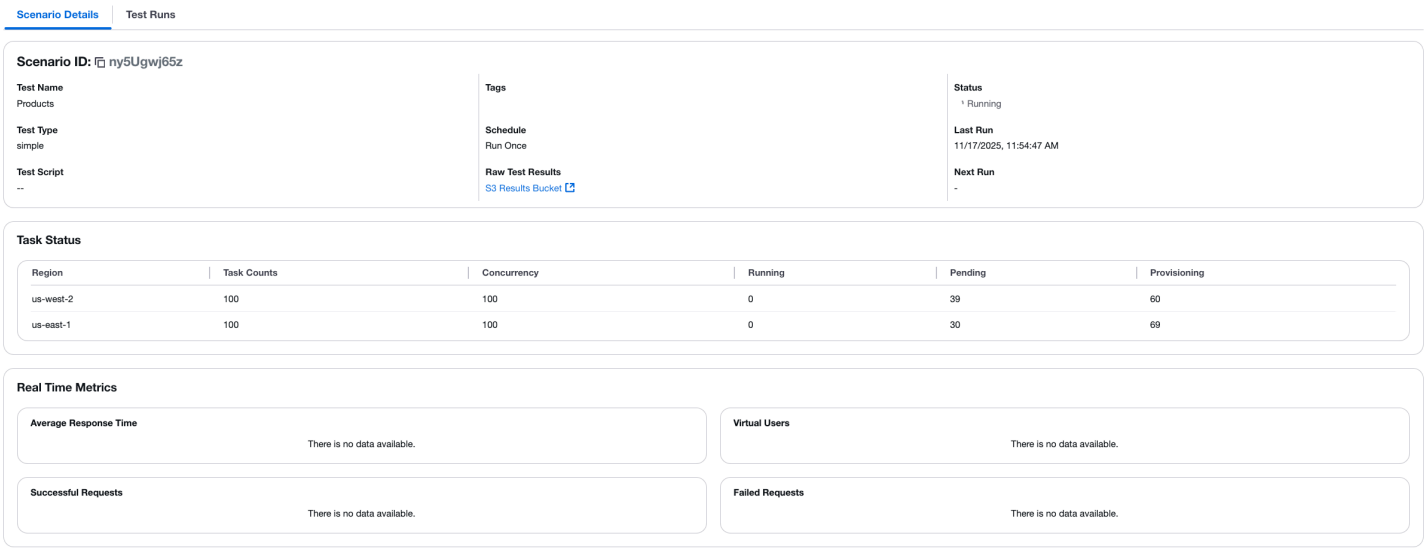
Gestion des scénarios de test

Après avoir créé un scénario de test, vous pouvez :

- Modifier : modifiez la configuration de test. Cas d’utilisation courants :
  - Affiner la forme du trafic pour atteindre le taux de transaction souhaité.
- Copier : dupliquez un scénario de test existant pour créer des variantes. Cas d’utilisation courants :
  - Mettre à jour les points de terminaison ou ajouter des headers/body paramètres.
  - Ajouter ou modifier des scripts de test.
- Supprimer : supprimez les scénarios de test dont vous n'avez plus besoin.

Exécuter un scénario de test

Après avoir créé un scénario de test, vous pouvez l'exécuter immédiatement ou le planifier pour qu'il s'exécute à un moment précis dans le futur. Lorsque vous accédez à un test en cours, la console affiche l'onglet Détails du scénario avec l'état des tâches et des mesures en temps réel.



Affichage des détails du scénario

L'onglet Détails du scénario affiche des informations clés sur votre test. Le tableau d'état des tâches fournit des informations en temps réel pour chaque région.

Tableau d'état des tâches

Le tableau d'état des tâches affiche des informations en temps réel pour chaque région :

- Région : région AWS dans laquelle les tâches sont exécutées
- Nombre de tâches : nombre total de tâches configurées pour la région
- Concurrence : nombre d'utilisateurs virtuels par tâche
- En cours d'exécution : nombre de tâches exécutant actuellement le test
- En attente : nombre de tâches en attente de démarrage
- Provisionnement : nombre de tâches en cours de provisionnement

Flux de travail d'exécution des tests

Lorsqu'un test démarre, le flux de travail suivant se produit :

1. Provisionnement des tâches : la solution fournit des conteneurs (tâches) dans les régions AWS spécifiées. Les tâches apparaissent dans la colonne « Provisioning ».

2. Démarrage des tâches : la solution continue de provisionner les tâches jusqu'à ce que le nombre de tâches cible soit atteint dans chaque région. Les tâches passent de « Provisionnement » à « En attente » puis « En cours ».
3. Génération de trafic : une fois que la solution a provisionné toutes les tâches d'une région, elle commence à envoyer du trafic vers votre point de terminaison cible.
4. Exécution du test : le test s'exécute pendant la durée configurée (montée en puissance et temps d'attente).
5. Analyse des résultats : à la fin du test, une tâche d'analyse en arrière-plan regroupe et traite les résultats de toutes les régions.

Statuts d'exécution des tests

Les essais peuvent avoir les statuts suivants :

- Planifié : le test est prévu pour être exécuté dans le futur.
- En cours d'exécution : le test est actuellement en cours.
- Annulé : un utilisateur a annulé un test en cours.
- Erreur - Le test a rencontré une erreur.
- Terminé : le test s'est terminé avec succès et les résultats sont prêts.

Surveillance à l'aide de données en temps réel

Si vous avez activé les données en temps réel lors de la création du scénario de test, vous pouvez consulter les métriques en temps réel pendant l'exécution du test. La section Mesures en temps réel affiche quatre graphiques qui sont mis à jour en permanence au fur et à mesure que le test progresse, les données étant agrégées à intervalles d'une seconde.

Descriptions des graphiques

Temps de réponse moyen

Affiche le temps de réponse moyen en secondes pour les demandes traitées par chaque région. L'axe Y indique le temps de réponse en secondes et l'axe X indique l'heure du jour. Chaque région est représentée par une couleur différente dans la légende.

Utilisateurs virtuels

Indique le nombre d'utilisateurs virtuels simultanés générant activement de la charge dans chaque région. Le graphique montre comment les utilisateurs virtuels augmentent pendant le test et maintiennent le niveau de simultanéité cible.

Demandes réussies

Affiche le nombre cumulé de demandes réussies au fil du temps pour chaque région. Le graphique montre le taux de traitement des demandes acceptées.

Demandes ayant échoué

Affiche le nombre cumulé de demandes ayant échoué au fil du temps pour chaque région. Un nombre faible ou nul indique une bonne exécution du test.

Visualisation multirégionale

Lorsque vous effectuez des tests dans plusieurs régions, chaque graphique affiche les données de toutes les régions simultanément. La légende au bas de chaque graphique indique la couleur qui représente chaque région (par exemple, us-west-2 et us-east-1).

Mise en œuvre technique

Le groupe de CloudWatch journaux pour les tâches Fargate contient un filtre d'abonnement qui capture les résultats des tests. Lorsque le modèle est détecté, une fonction Lambda structure les données et les publie dans une rubrique AWS IoT Core. La console Web s'abonne à cette rubrique et affiche les statistiques en temps réel.

> **Note**
>
> Les données en temps réel sont éphémères et ne sont disponibles que pendant le test. La console Web conserve un maximum de 5 000 points de données, après quoi les données les plus anciennes sont remplacées par les plus récentes. Si la page est actualisée, les graphiques seront vides et repartiront du prochain point de données disponible. Une fois le test terminé, la solution stocke les données de résultats dans DynamoDB et Amazon S3. Si aucune donnée n'est encore disponible, les graphiques indiquent « Aucune donnée n'est disponible ».

Annulation d'un test

Vous pouvez annuler un test en cours depuis la console Web. Lorsque vous annulez un test, le flux de travail suivant se produit :

1. La demande d'annulation est envoyée à l'`microservices`API
2. L'`microservices`API appelle la fonction `task-canceler` Lambda, qui arrête toutes les tâches actuellement lancées.
3. Si la fonction `task-runner` Lambda continue de s'exécuter après l'appel d'annulation initial, les tâches peuvent continuer à être lancées brièvement
4. Une fois la fonction `task-runner` Lambda terminée, AWS Step Functions passe à l'`Cancel Test`étape, qui exécute à nouveau la fonction `task-canceler` Lambda pour arrêter les tâches restantes

> **Note**
>
> Les tests annulés mettent du temps à terminer le processus d'arrêt car la solution met fin à tous les conteneurs. Le statut du test passera à « Annulé » une fois que toutes les ressources auront été nettoyées.

Explorez les résultats des tests

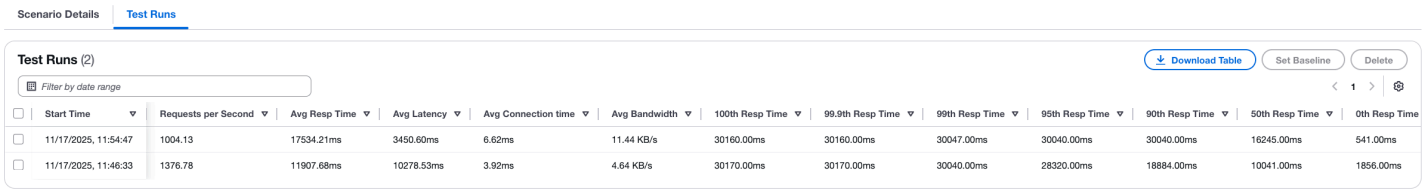
Une fois le travail d'analyse terminé, les résultats des tests sont disponibles pour analyse. La solution fournit des mesures et des outils complets pour vous aider à comprendre les performances de votre application sous charge.

Métriques récapitulatives des essais

Lorsqu'un test est terminé, la solution génère un résumé qui inclut les mesures suivantes :

- Temps de réponse moyen : temps de réponse moyen, en secondes, pour toutes les demandes générées par le test.
- Latence moyenne : latence moyenne, en secondes, pour toutes les demandes générées par le test.
- Temps de connexion moyen : temps moyen, en secondes, nécessaire pour se connecter à l'hôte pour toutes les demandes.
- Bande passante moyenne : bande passante moyenne pour toutes les demandes générées par le test.
- Nombre total : nombre total de demandes.
- Nombre de demandes réussies : nombre total de demandes réussies.
- Nombre d'erreurs : nombre total d'erreurs.
- Demandes par seconde : nombre moyen de demandes par seconde pour toutes les demandes générées par le test.
- Percentiles : percentiles de temps de réponse, y compris p50 (médiane), p90, p95 et p99, indiquant la distribution des temps de réponse entre toutes les demandes.

Tableau des essais



Le tableau des essais affiche l'historique de tous les essais d'un scénario. Vous pouvez :

- Consultez les statistiques récapitulatives pour chaque essai.
- Définissez un cycle de test de référence pour la comparaison des performances.
- Téléchargez le tableau sous forme de fichier CSV.
- Basculez entre les colonnes pour personnaliser votre affichage.
- Sélectionnez un essai pour afficher les résultats détaillés.

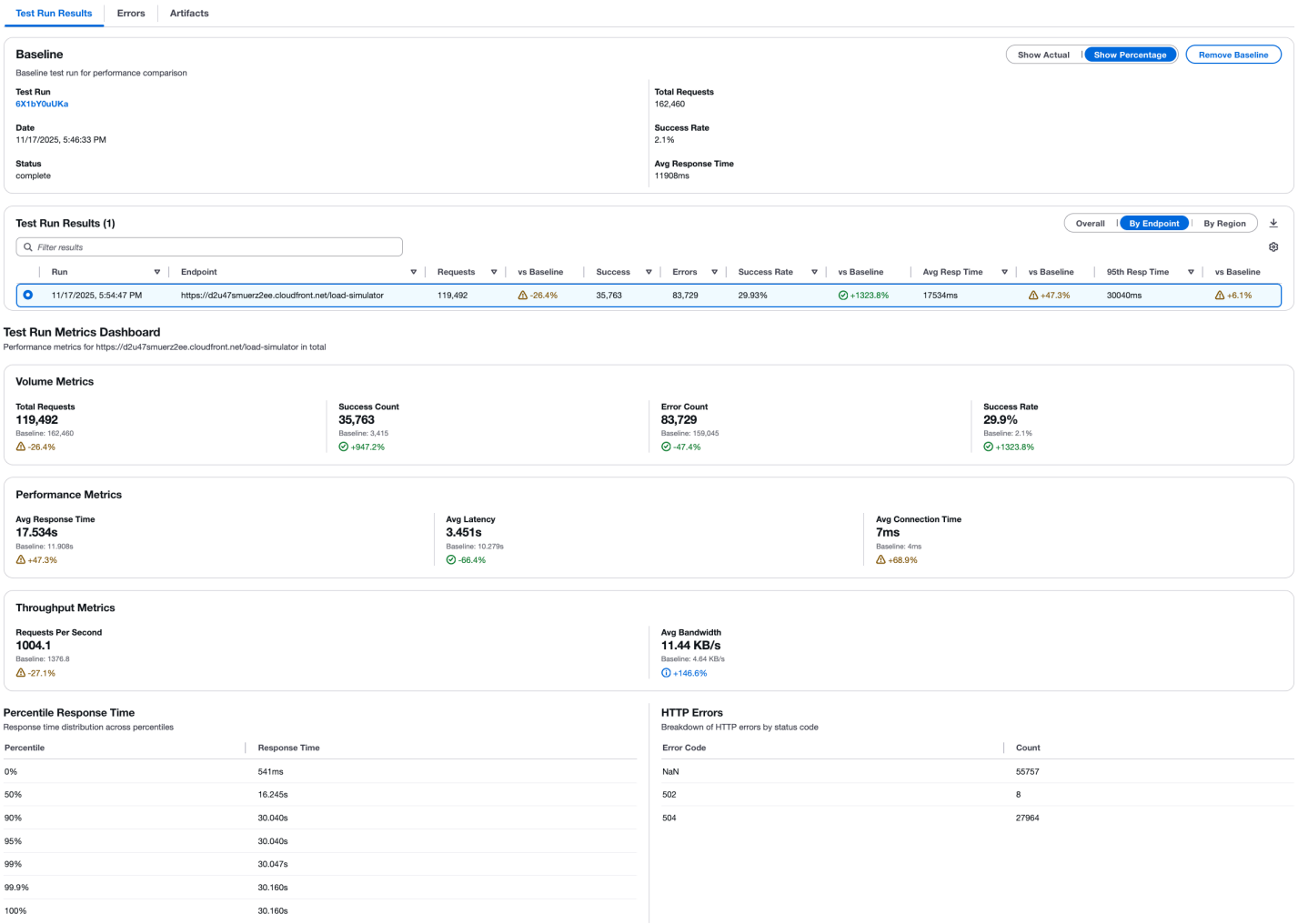
Comparaison de référence

Vous pouvez désigner un cycle de test comme référence pour comparer les futurs essais par rapport à celui-ci. Lorsqu'une ligne de base est définie :

- Le tableau des essais montre les différences en pourcentage (+/- %) par rapport à la base de référence pour chaque métrique.
- L'indicateur de référence vous aide à identifier rapidement les améliorations ou les régressions des performances.
- Vous pouvez modifier ou effacer la ligne de base à tout moment.

Résultats de test détaillés

La sélection d'un essai ouvre la vue détaillée des résultats avec trois onglets : Résultats du test, Erreurs et Artefacts.



Informations de base

Si un test de référence est défini, il s'affiche en haut de la page. Vous pouvez choisir Afficher la valeur réelle, Afficher le pourcentage ou Supprimer la ligne de base pour contrôler le mode d'affichage des comparaisons de lignes de base.

Tableau des résultats des essais

Le tableau des résultats fournit des mesures détaillées avec les fonctionnalités suivantes :

Vues dimensionnelles

Basculez entre trois vues à l'aide des boutons de dimension :

- Globalement : résultats agrégés pour tous les terminaux et toutes les régions
- Par point de terminaison - Résultats ventilés par point de terminaison individuel

- Par région - Résultats ventilés par région AWS

Boutons d'action

- Afficher les valeurs réelles - Afficher les valeurs métriques réelles
- Afficher le pourcentage - Afficher les différences en pourcentage par rapport à la ligne de base
- Supprimer la ligne de base - Efface la comparaison de la ligne de base

Exportation et personnalisation des données

- Téléchargez le tableau des résultats sous forme de fichier CSV
- Basculez entre les colonnes pour personnaliser votre affichage
- Filtrez et triez les données pour vous concentrer sur des indicateurs spécifiques
- Filtrez et triez les données pour vous concentrer sur des indicateurs spécifiques.

Onglet Erreurs

L'onglet erreurs fournit une analyse détaillée des erreurs :

- Afficher le nombre d'erreurs par type.
- Consultez les erreurs agrégées par test global ou par point de terminaison.
- Identifiez les modèles de demandes ayant échoué.
- Résolvez les problèmes liés à des terminaux ou à des régions spécifiques.

Onglet Artifacts

L'onglet artefacts vous permet d'accéder à tous les fichiers générés lors du test :

- Afficher les artefacts individuels (journaux, fichiers de résultats).
- Téléchargez des artefacts spécifiques pour une analyse hors ligne.
- Téléchargez tous les artefacts du test sous forme d'archive unique.

Structure des résultats S3

Dans la version 4.0, la structure des résultats S3 a été modifiée pour améliorer l'organisation :

- Nouvelle structure -`scenario-id/test-run-id/results-files`.
- Structure héritée - Les tests exécutés avant la version 4.0 affichent tous les fichiers de résultats au niveau de l'ID du scénario.

> **Note**
>
> Les résultats des tests sont affichés dans la console. Vous pouvez également accéder aux résultats bruts des tests directement dans le compartiment Amazon S3 situé sous le `Results` dossier. Pour plus d'informations sur les résultats des tests Taurus, consultez la section Génération de rapports de test dans le manuel d'utilisation de Taurus.

Intégration au serveur MCP

Si vous avez déployé le composant serveur MCP optionnel lors du déploiement de la solution, vous pouvez intégrer la solution de test de charge distribué aux outils de développement d'intelligence artificielle qui prennent en charge le protocole Model Context. Le serveur MCP fournit un accès programmatique pour récupérer, gérer et analyser les tests de charge par le biais d'assistants IA.

Les clients peuvent se connecter au serveur DLT MCP en utilisant le client de leur choix (Amazon Q, Claude, etc.), dont les instructions de configuration sont légèrement différentes. Cette section fournit des instructions de configuration pour MCP Inspector, Amazon Q CLI, Cline et Amazon Q Suite.

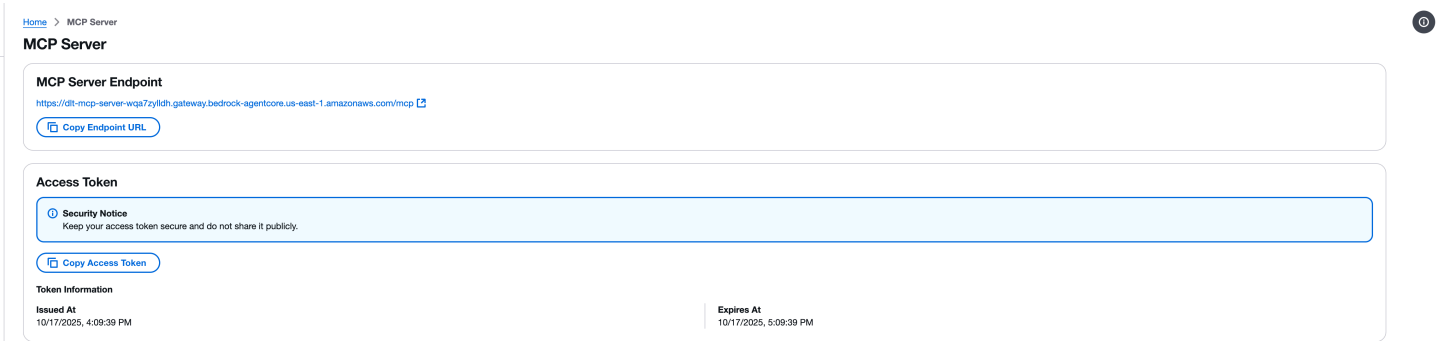
Étape 1 : Obtenir le point de terminaison et le jeton d'accès MCP

Avant de configurer un client MCP, vous devez récupérer le point de terminaison et le jeton d'accès de votre serveur MCP depuis la console Web DLT.

1. Accédez à la page du serveur MCP dans la console Web de test de charge distribué.
2. Localisez la section Point de terminaison du serveur MCP.
3. Copiez l'URL du point de terminaison en utilisant le bouton Copier l'URL du point de terminaison. L'URL du point de terminaison suit le format suivant : `https://[api-id].execute-api.[region].amazonaws.com/[stage]/gateway/backend-agent/sse/mcp`
4. Localisez la section Jeton d'accès.
5. Copiez le jeton d'accès à l'aide du bouton Copier le jeton d'accès.

Important

Protégez votre jeton d'accès et ne le partagez pas publiquement. Le jeton fournit un accès en lecture seule à votre solution de test de charge distribué via l'interface MCP.



Étape 2 : Test avec MCP Inspector

Le Model Context Protocol propose MCP Inspector, un outil permettant de se connecter directement aux serveurs MCP et d'invoquer des outils. Cela fournit une interface utilisateur pratique et des exemples de requêtes réseau pour tester votre connexion au serveur MCP avant de configurer les clients AI.

Note

MCP Inspector nécessite la version 0.17 ou ultérieure. Toutes les demandes peuvent également être effectuées directement avec JSON RPC, mais MCP Inspector fournit une interface plus conviviale.

Installez et lancez MCP Inspector

1. Installez npm si nécessaire.
2. Exécutez la commande suivante pour lancer MCP Inspector :

```
npx @modelcontextprotocol/inspector
```

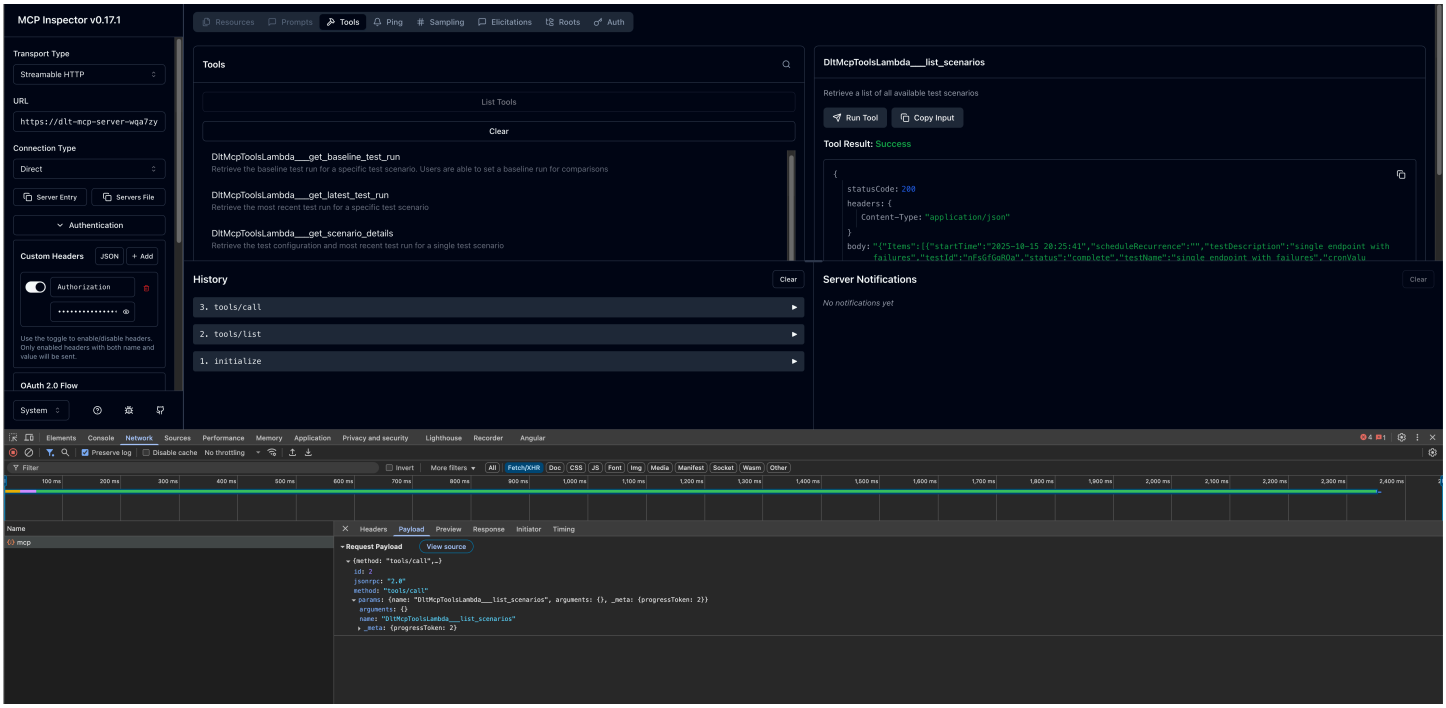
Configuration de la connexion

1. Dans l'interface MCP Inspector, entrez l'URL de votre point de terminaison du serveur MCP.
2. Ajoutez un en-tête d'autorisation avec votre jeton d'accès.
3. Cliquez sur Connect pour établir la connexion.

Outils d'appel

Une fois connecté, vous pouvez tester les outils MCP disponibles :

1. Parcourez la liste des outils disponibles dans le panneau de gauche.
2. Sélectionnez un outil (par exemple,`list_scenarios`).
3. Fournissez tous les paramètres requis.
4. Cliquez sur Invoke pour exécuter l'outil et afficher la réponse.



Étape 3 : Configuration des clients de développement AI

Après avoir vérifié votre connexion au serveur MCP avec MCP Inspector, vous pouvez configurer votre client de développement AI préféré.

interface de ligne de commande d’Amazon Q

La CLI Amazon Q fournit un accès en ligne de commande au développement assisté par l'IA avec intégration au serveur MCP.

Étapes de configuration

1. Modifiez le fichier `mcp.json` de configuration. Pour plus d'informations sur l'emplacement du fichier de configuration, reportez-vous à la section Configuration de serveurs MCP distants dans le manuel Amazon Q Developer User Guide.
2. Ajoutez la configuration de votre serveur DLT MCP :

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

```
      "headers": {
        "Authorization": "your_access_token_here"
      }
    }
  }
}
```

Vérifiez la configuration

1. Dans un terminal, tapez q pour lancer Amazon Q CLI.
2. Tapez `/mcp` pour voir tous les serveurs MCP disponibles.
3. Tapez `/tools` pour voir les outils disponibles fournis par les serveurs MCP configurés `dlt-mcp` et les autres serveurs MCP configurés.
4. Vérifiez que l'initialisation est `dlt-mcp` réussie.

Cline

Cline est un assistant de codage AI qui prend en charge l'intégration des serveurs MCP.

Étapes de configuration

1. Dans Cline, accédez à Gérer les serveurs MCP > Configurer > Configurer les serveurs MCP.
2. Mettez à jour le `cline_mcp_settings.json` fichier :

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

3. Enregistrez le fichier de configuration.
4. Redémarrez Cline pour appliquer les modifications.

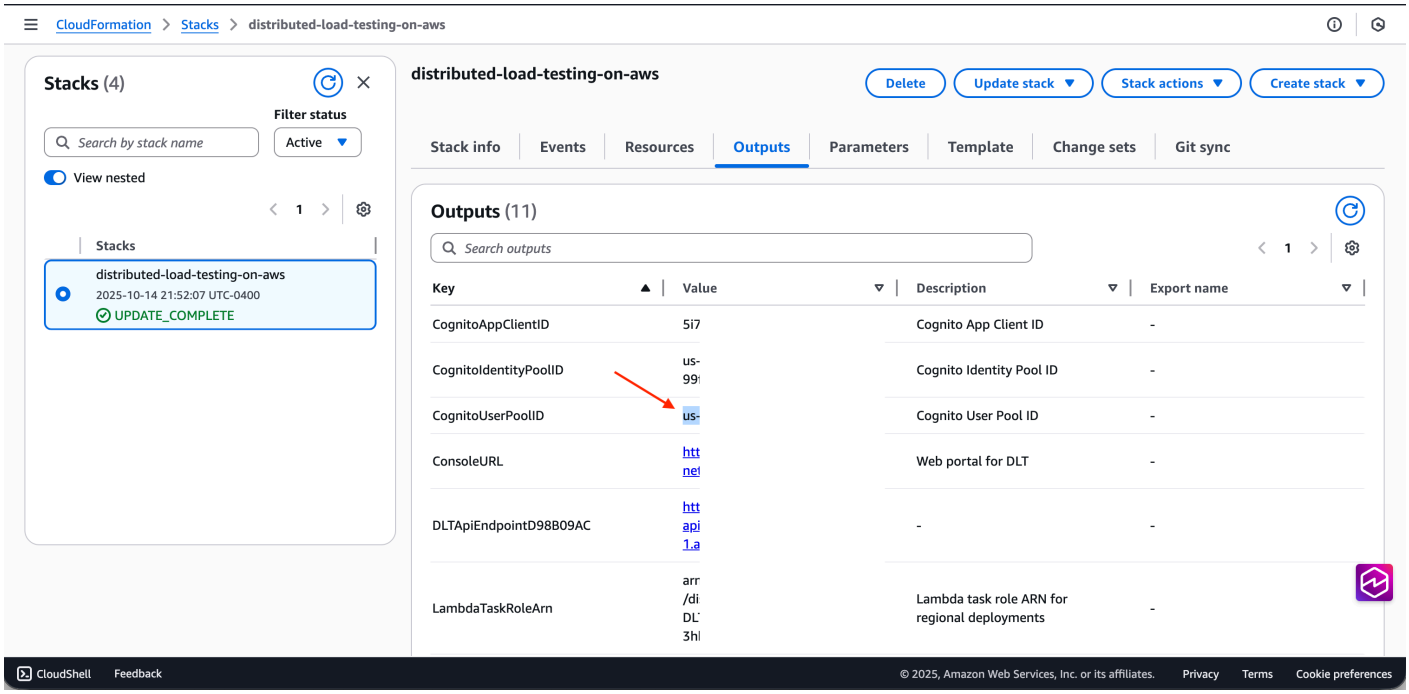
Suite Amazon Q

Amazon Q Suite fournit une plate-forme d'assistance AI complète prenant en charge les actions du serveur MCP.

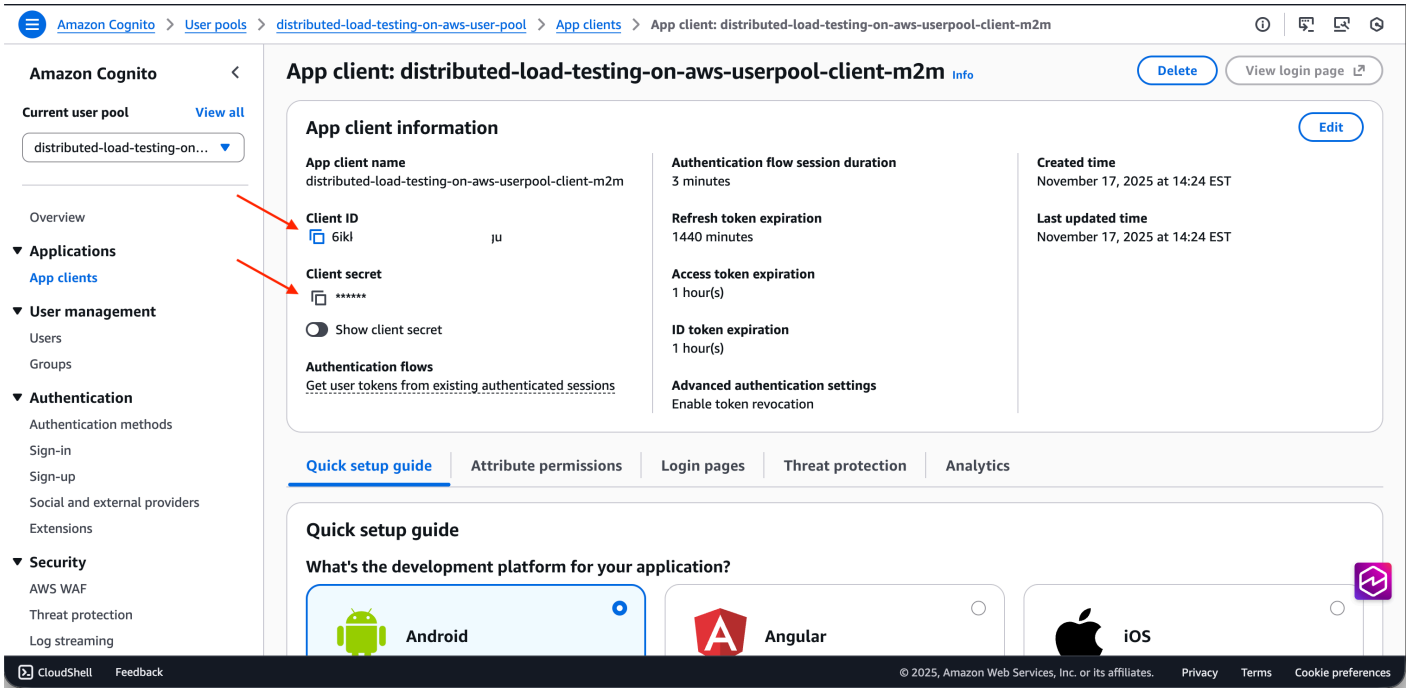
Prérequis

Avant de configurer le serveur MCP dans Amazon Q Suite, vous devez récupérer les informations d' OAuth identification du groupe d'utilisateurs Cognito de votre déploiement DLT :

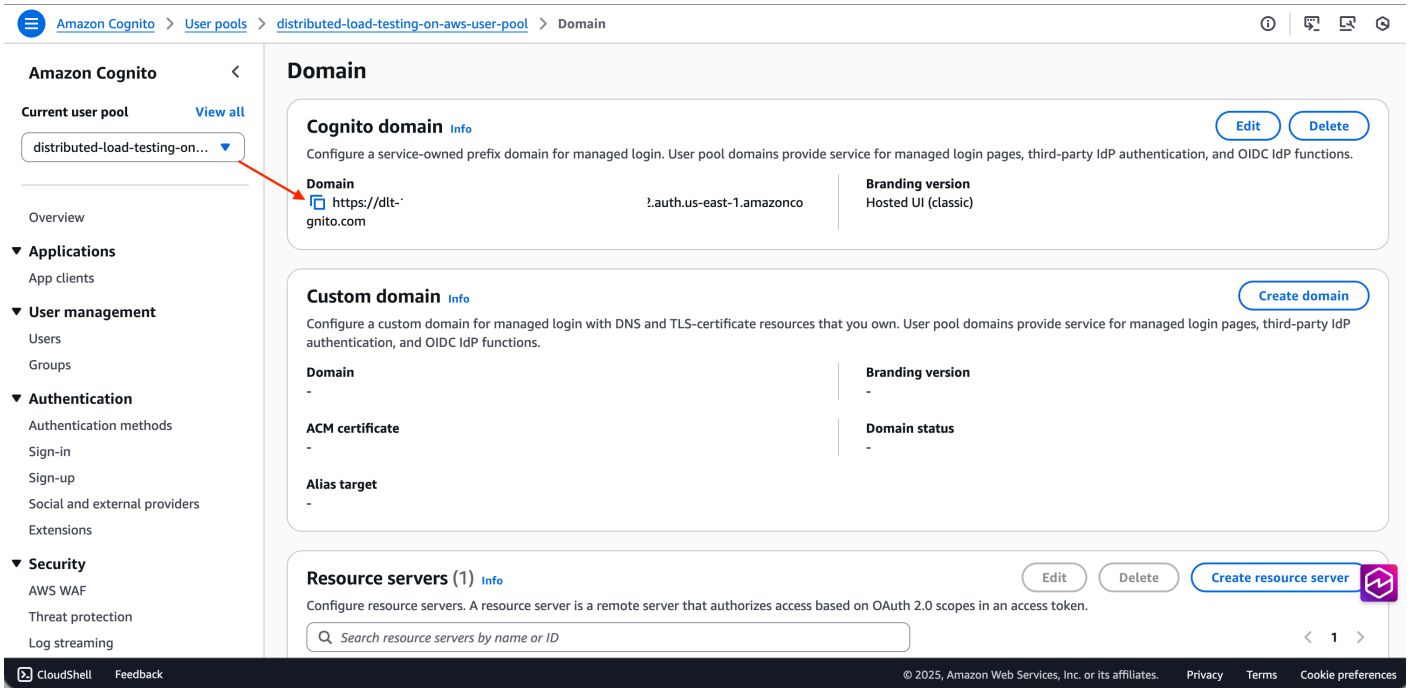
1. Accédez à la CloudFormation console AWS.
2. Sélectionnez la pile de tests de charge distribués.
3. Dans l'onglet Sorties, recherchez et copiez l'ID du groupe d'utilisateurs Cognito associé à votre déploiement DLT.



4. Accédez à la console Amazon Cognito.
5. Sélectionnez le groupe d'utilisateurs à l'aide de l'ID du groupe d'utilisateurs dans les CloudFormation sorties.
6. Dans le volet de navigation de gauche, sélectionnez Intégration des applications > Clients des applications.



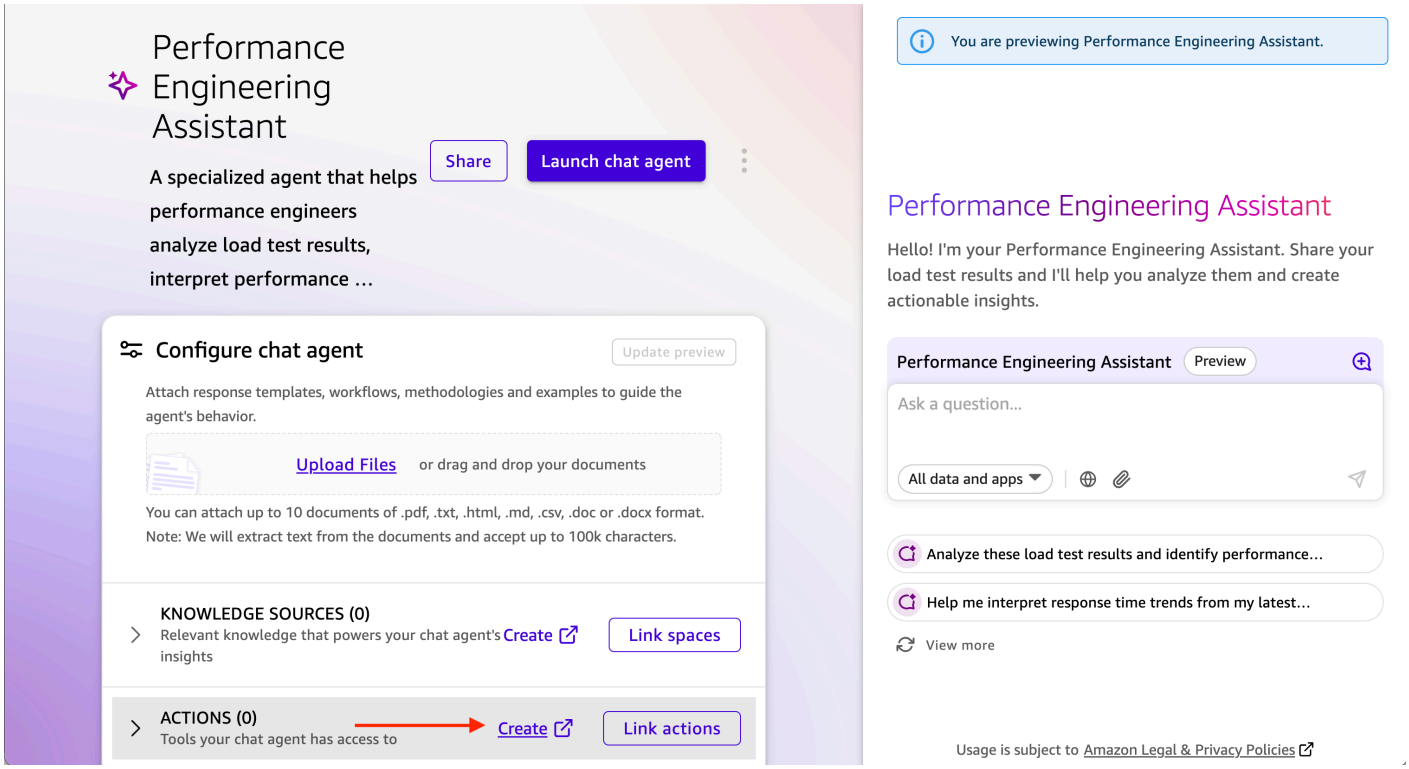
7. Localisez le client de l'application dont le nom se termine par `m2m` (machine-to-machine).
8. Copiez l'ID client et le secret du client.
9. Obtenez le domaine du groupe d'utilisateurs dans l'onglet Domaine.

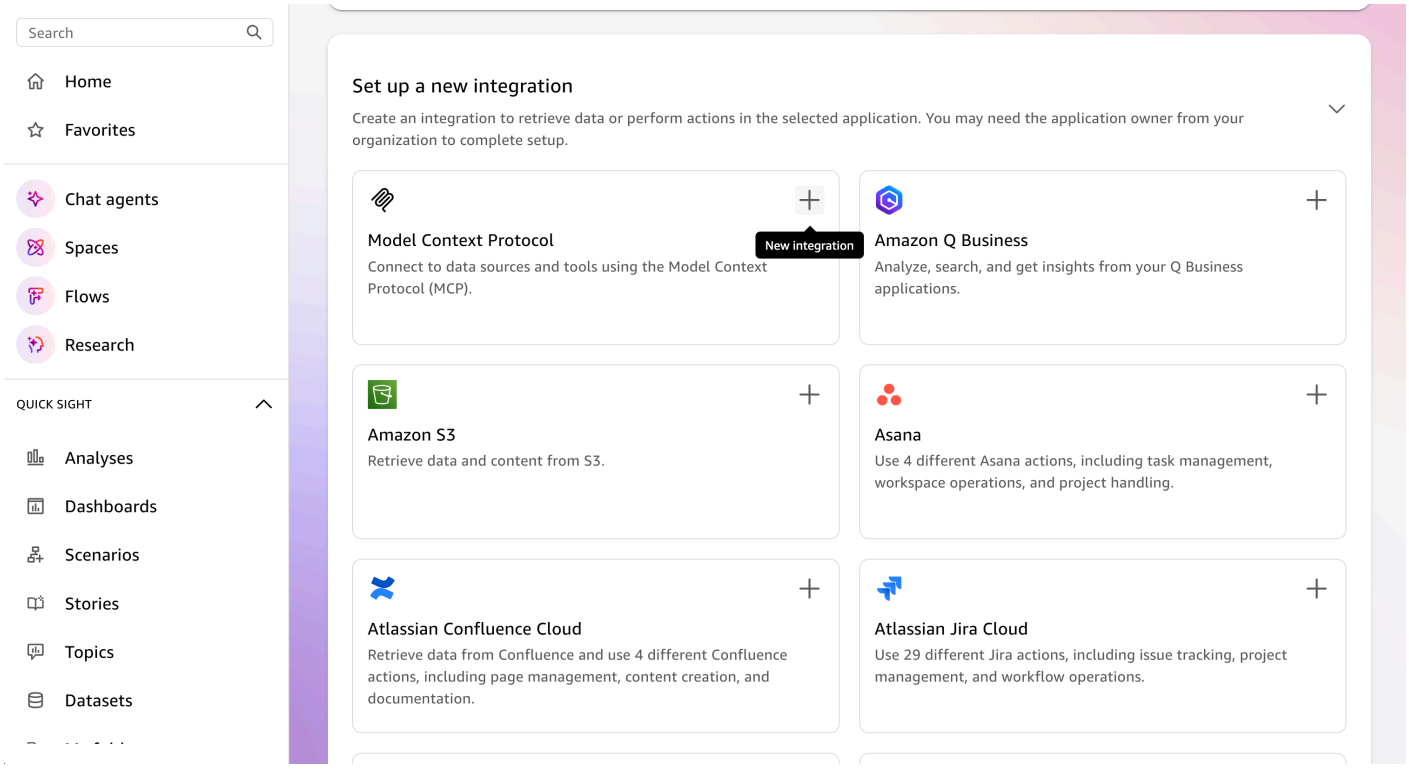


10. Créez l'URL du point de terminaison du jeton en l'ajoutant `/oauth2/token` à la fin du domaine.

Étapes de configuration

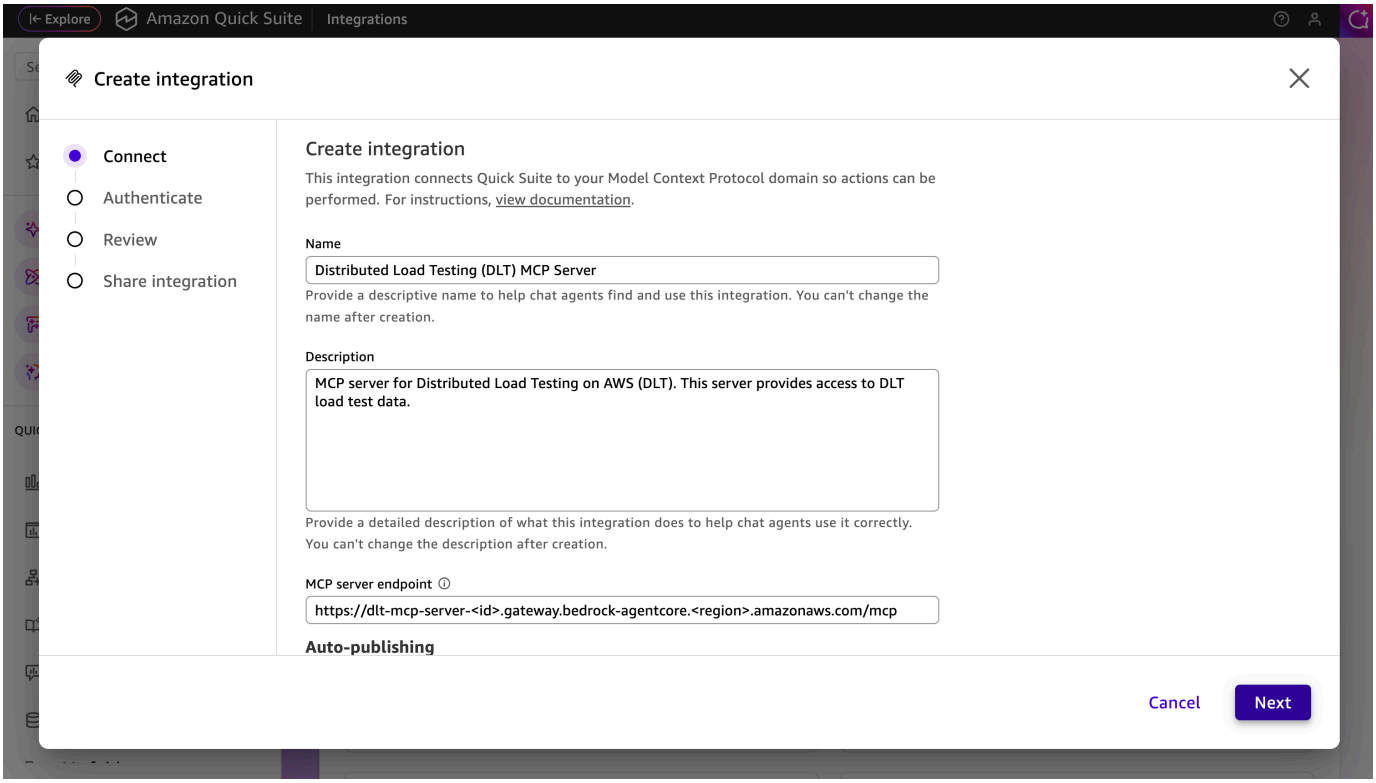
1. Dans Amazon Q Suite, créez un nouvel agent ou sélectionnez un agent existant.
2. Ajoutez une invite d'agent qui décrit comment interagir avec le serveur DLT MCP.
3. Ajoutez une nouvelle action et sélectionnez l'action du serveur MCP.



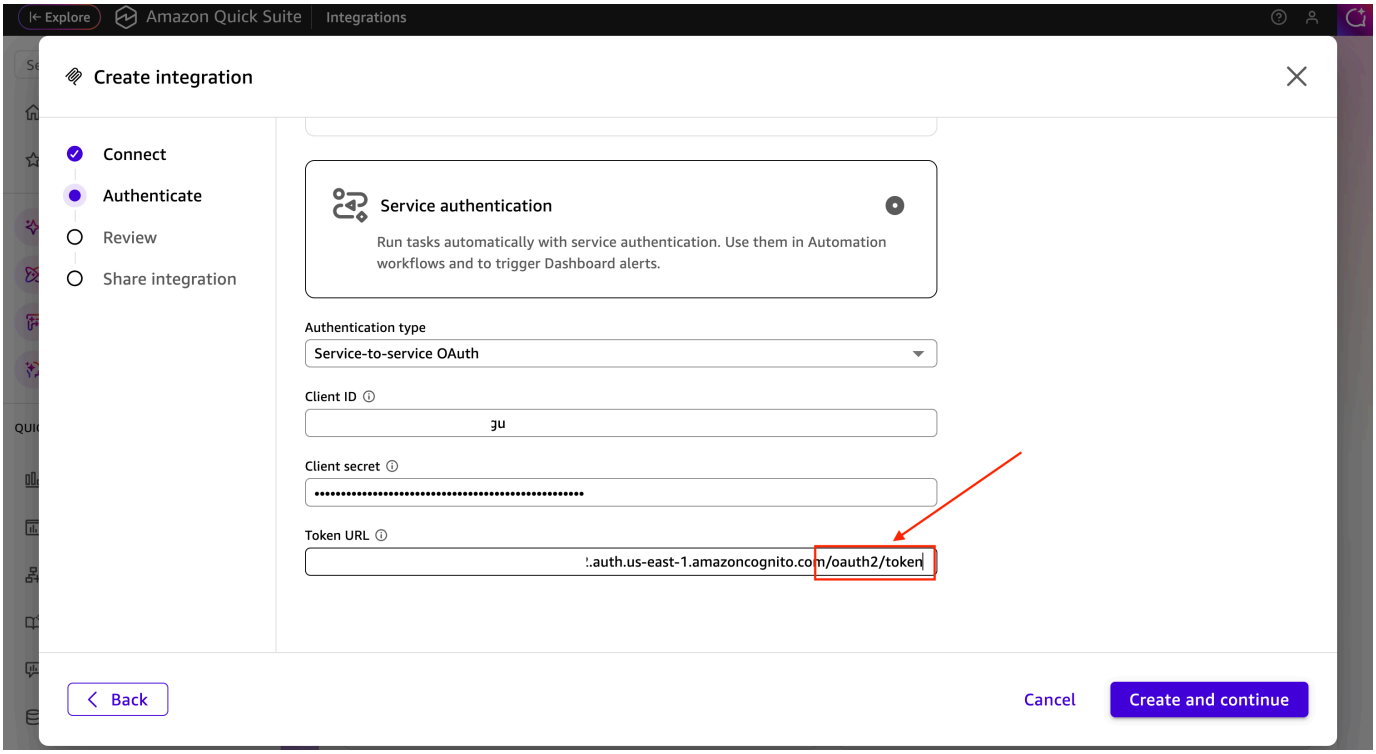


4. Configurez les détails du serveur MCP :

- URL du serveur MCP : votre point de terminaison DLT MCP



- Type d'authentification : Authentification basée sur le service
- Point de terminaison du jeton : URL du point de terminaison de votre jeton Cognito
- ID client : ID client du client de l'application m2m
- Secret du client : le secret du client de l'application m2m



5. Enregistrez la configuration des actions du serveur MCP.
6. Ajoutez la nouvelle action du serveur MCP à votre agent.

Lancez et testez l'agent

1. Lancez l'agent dans Amazon Q Suite.
2. Entamez une conversation avec l'agent à l'aide d'instructions en langage naturel.
3. L'agent utilisera les outils MCP pour récupérer et analyser vos données de test de charge.

Exemples d’invites

Les exemples suivants montrent comment interagir avec votre assistant AI pour analyser les données de test de charge via l'interface MCP. Personnalisez le test IDs, les plages de dates et les critères en fonction de vos besoins de test spécifiques.

Pour des informations détaillées sur les outils MCP disponibles et leurs paramètres, reportez-vous aux spécifications des outils MCP dans le manuel du développeur.

Demande simple de résultats de test

L'interaction en langage naturel avec le serveur MCP peut être aussi simple `Show me the load tests that have completed in the last 24 hours with their associated completion status` ou plus descriptive, par exemple

```
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.
```

Analyse de performance interactive avec divulgation progressive

```
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.
```

Validation du niveau de préparation

```
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]%.
```

Analyse des tendances en matière de performance

```
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.
```

Dépannage des tests ayant échoué

```
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].
```

Guide du développeur

Cette section fournit le code source de la solution ainsi que des personnalisations supplémentaires.

Code source

Consultez notre GitHub référentiel pour télécharger les modèles et les scripts de cette solution et pour partager vos personnalisations avec d'autres.

Maintenance

Cette solution utilise des images Docker avec des versions fixes correspondant à chaque version de solution. Par défaut, les mises à jour automatiques sont désactivées, ce qui vous permet de contrôler totalement le moment et la version des mises à jour appliquées à votre déploiement. L'équipe AWS Solutions utilise Amazon ECR Enhanced Scanning pour détecter les vulnérabilités et les expositions courantes (CVEs) dans l'image de base et les packages installés. Dans la mesure du possible, l'équipe publie des images corrigées avec le même tag de version afin de les résoudre CVEs sans compromettre la compatibilité avec la version publiée de la solution.

Lorsque des images sont corrigées sur la même version mineure, la balise stable est automatiquement mise à jour et une balise d'image supplémentaire est créée au format`<solution-version>_<date-of-fix>`. Si une version majeure ou mineure est publiée, vous devez effectuer une mise à jour complète pour obtenir la dernière version de l'image, car la balise stable est incrémentée pour correspondre à la version de la solution.

Si vous optez pour les mises à jour automatiques pendant le déploiement, les modifications apportées à l'image, y compris les correctifs CVE et les corrections de bogues mineurs, sont automatiquement appliquées à la dernière version mineure correspondante.

Versions

Par défaut, cette solution est déployée avec les mises à jour automatiques désactivées. Cela signifie que la version de l'image du conteneur est verrouillée sur la version spécifique correspondant à la version de votre solution déployée, ce qui vous permet de contrôler totalement les mises à jour de version.

Si vous choisissez d'activer les mises à jour automatiques en sélectionnant Oui lors du CloudFormation déploiement, la solution recevra automatiquement les correctifs de sécurité et

les corrections de bogues mineurs et non révolutionnaires jusqu'à la dernière version mineure correspondante. Par exemple, si vous déployez la version 4.0.0 avec les mises à jour automatiques activées, vous recevrez les mises à jour jusqu'à la version 4.0.x, mais pas la version 4.1.0 ou supérieure.

Pour contrôler manuellement la version de l'image du conteneur, vous pouvez modifier la définition de la tâche afin de spécifier une version d'image particulière à l'aide du format de version balisé. Cela vous permet d'épingler une version d'image spécifique quel que soit le paramètre de mise à jour automatique.

Personnalisation de l'image du conteneur

Cette solution utilise un référentiel d'images public Amazon Elastic Container Registry (Amazon ECR) géré par AWS pour stocker l'image utilisée pour exécuter les tests configurés. Si vous souhaitez personnaliser l'image du conteneur, vous pouvez la reconstruire et l'envoyer dans un référentiel d'images ECR de votre propre compte AWS.

Si vous souhaitez personnaliser cette solution, vous pouvez utiliser l'image du conteneur par défaut ou modifier ce conteneur en fonction de vos besoins. Si vous personnalisez la solution, utilisez l'exemple de code suivant pour déclarer les variables d'environnement avant de créer votre solution personnalisée.

```
#!/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
```

Si vous choisissez de personnaliser l'image du conteneur, vous pouvez l'héberger soit dans un référentiel d'images privé, soit dans un référentiel d'images public dans votre compte AWS. Les ressources d'image se trouvent dans le `deployment/ecr/distributed-load-testing-on-aws-load-tester` répertoire, situé dans la base de code.

Vous pouvez créer et envoyer l'image vers la destination hôte.

- Pour les référentiels et images privés Amazon ECR, reportez-vous aux référentiels privés et aux images privées Amazon ECR dans le guide de l'utilisateur Amazon ECR.
- Pour les référentiels publics et les images Amazon ECR, reportez-vous aux référentiels publics et aux images publiques Amazon ECR dans le manuel Amazon ECR Public User Guide.

Une fois que vous avez créé votre propre image, vous pouvez déclarer les variables d'environnement suivantes avant de créer votre solution personnalisée.

```
#!/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
```

L'exemple suivant montre le fichier conteneur.

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

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

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


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

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

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

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

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

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

Outre un fichier conteneur, le répertoire contient le script bash suivant qui télécharge la configuration de test depuis Amazon S3 avant d'exécuter le Taurus/Blazemeter programme.

```
#!/bin/bash

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

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

sigterm_handler() {
  if [ $pypid -ne 0 ]; then
    echo "container received SIGTERM."
    kill -15 $pypid
```

```
    wait $pypid
    exit 143 #128 + 15
  fi
}
trap 'sigterm_handler' SIGTERM

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

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

# download JMeter jmx file
if [ "$TEST_TYPE" != "simple" ]; then
  # setting the log file values to the test type
  LOG_FILE="${TEST_TYPE}.log"
  OUT_FILE="${TEST_TYPE}.out"
  ERR_FILE="${TEST_TYPE}.err"

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

```
  fi

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

    # If zip and locust, make sure to pick locustfile
    if [ "$TEST_TYPE" != "locust" ]; then
      TEST_SCRIPT=$(find . -name "*.${EXT}" | head -n 1)
    else
      TEST_SCRIPT=$(find . -name "locustfile.py" | head -n 1)
    fi
    # only looks for the first test script file.
    TEST_SCRIPT=`find . -name "*.${EXT}" | head -n 1`
    echo $TEST_SCRIPT
    if [ -z "$TEST_SCRIPT" ]; then
      echo "There is no test script (.${EXT}) in the zip file."
      exit 1
    fi

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

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

```
      cp -v $BUNDLED_PLUGIN_DIR/*.jar $JMETER_EXT_PATH
    fi
  fi
fi

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

echo "Running test"

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

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

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

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

    files=(`cat results.txt`)
    extensions=()
```

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

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

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

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

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

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


  sed -i.bak 's/<\/FinalStatus>/<TaskId>'"$TASK_ID"'<\/TaskId><\/FinalStatus>/' /tmp/
artifacts/results.xml
```

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

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

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

  if [ "$TEST_TYPE" == "simple" ]; then
    TEST_TYPE="jmeter"
  fi

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

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

Outre le Dockerfile et le script bash, deux scripts Python sont également inclus dans le répertoire. Chaque tâche exécute un script Python à partir du script bash. Les tâches de travail exécutent le `ecslistener.py` script, tandis que la tâche principale exécute le `ecscontroller.py` script. Le `ecslistener.py` script crée un socket sur le port 50000 et attend un message. Le

`ecscontroller.py` script se connecte au socket et envoie le message de test de démarrage aux tâches de travail, ce qui leur permet de démarrer simultanément.

API de test de charge distribuée

Cette solution de test de charge vous permet d'exposer les données des résultats de test de manière sécurisée. L'API fait office de « porte d'entrée » pour accéder aux données de test stockées dans Amazon DynamoDB. Vous pouvez également utiliser le APIs pour accéder à toutes les fonctionnalités étendues que vous intégrez à la solution.

Cette solution utilise un groupe d'utilisateurs Amazon Cognito intégré à Amazon API Gateway pour l'identification et l'autorisation. Lorsqu'un groupe d'utilisateurs est utilisé avec l'API, les clients ne sont autorisés à appeler les méthodes activées par le groupe d'utilisateurs qu'après avoir fourni un jeton d'identité valide.

Pour plus d'informations sur l'exécution de tests directement via l'API, consultez la section Signing Requests dans la documentation de référence de l'API REST Amazon API Gateway.

Les opérations suivantes sont disponibles dans l'API de la solution.

> **Note**
>
> Pour plus d'informations `testScenario` et d'autres paramètres, reportez-vous aux scénarios et à l'exemple de charge utile dans le GitHub référentiel.

Informations sur la pile

- OBTENIR /stack-info

Scénarios

- GET /scénarios
- POST /scénarios
- OPTIONS/SCÉNARIOS
- OBTENEZ /scenarios/ {testId}
- POST /scenarios/ {testId}

- SUPPRIMER /scenarios/ {testId}
- OPTIONS /scénarios/ {testId}

Tests

- GET /scenarios/ {testId} /testruns
- GET /scenarios/ {testId} /testruns/ {} testRunId
- SUPPRIMER /scenarios/ {testId} /testruns/ {} testRunId

Base de référence

- GET /scenarios/ {testId} /baseline
- PUT /scenarios/ {testId} /baseline
- SUPPRIMER /scenarios/ {testId} /baseline

Tâches

- GET /tâches
- OPTIONS /tâches

Régions

- GET /régions
- OPTIONS /régions

OBTENIR /stack-info

Description

L'`GET /stack-info`opération récupère des informations sur la pile déployée, notamment l'heure de création, la région et la version. Ce point de terminaison est utilisé par le front-end.

Réponse

200 - Succès

Nom	Description
`created_time`	Horodatage ISO 8601 lors de la création de la pile (par exemple,) `2025-09-09T19:40:22Z`
`region`	Région AWS dans laquelle la pile est déployée (par exemple,`us-east-1`)
`version`	Version de la solution déployée (par exemple,`v4.0.0`)

Réponses aux erreurs

- `403`- Interdit : autorisations insuffisantes pour accéder aux informations de la pile
- `404`- Introuvable : les informations relatives à la pile ne sont pas disponibles
- `500`- Erreur interne du serveur

GET /scénarios

Description

L'`GET /scenarios`opération permet de récupérer une liste de scénarios de test.

Réponse

Nom	Description
`data`	Une liste de scénarios comprenant l'ID, le nom, la description, le statut, la durée d'exécution, les balises, le nombre total d'essais et la dernière exécution de chaque test

POST /scénarios

Description

L'`POST /scenarios`opération permet de créer ou de planifier un scénario de test.

Corps de la demande

Nom	Description
`testName`	Le nom du test
`testDescription`	Description du test
`testTaskConfigs`	Un objet qui spécifie `concurrency` (le nombre d'exécutions parallèles), `taskCount` (le nombre de tâches nécessaires pour exécuter un test) et `region` pour le scénario
`testScenario`	La définition du test, y compris la simultanéité, la durée du test, l'hôte et la méthode du test
`testType`	Le type de test (par exemple`simple`,`jmeter`)
`fileType`	Le type de fichier de téléchargement (par exemple,`none`,`script`,`zip`)
`tags`	Un tableau de chaînes permettant de classer les tests. Champ facultatif d'une longueur maximale de 5 (par exemple,`["blue", "3.0", "critical"]`)
`scheduleDate`	Date d'exécution d'un test. Fourni uniquement si vous planifiez un test (par exemple,`2021-02-28`)
`scheduleTime`	C'est le moment d'effectuer un test. Fourni uniquement si vous planifiez un test (par exemple,`21:07`)

Nom	Description
`scheduleStep`	Étape du processus de planification. Fourni uniquement si vous planifiez un test récurrent . (Les étapes disponibles incluent `create` et`start`)
`cronvalue`	La valeur cron pour personnaliser la planifica tion récurrente. Le cas échéant, omettez ScheduleDate et ScheduleTime.
`cronExpiryDate`	Date requise pour que le cron expire et ne s'exécute pas indéfiniment.
`recurrence`	Récurrence d'un test programmé. Fourni uniquement si vous planifiez un test récurrent (par exemple `daily`,`weekly`,`biweekly`, ou`monthly`)

Réponse

Nom	Description
`testId`	L'identifiant unique du test
`testName`	Le nom du test
`status`	État du test

OPTIONS/SCÉNARIOS

Description

L'`OPTIONS /scenarios`opération fournit une réponse à la demande avec les en-têtes de réponse CORS corrects.

Réponse

Nom	Description
`testId`	L'identifiant unique du test
`testName`	Le nom du test
`status`	État du test

OBTENEZ /scenarios/ {testId}

Description

L'`GET /scenarios/{testId}`opération permet de récupérer les détails d'un scénario de test spécifique.

Paramètres de demande

`testId`

- L'identifiant unique du test

  Type : String

  Obligatoire : oui

`latest`

- Paramètre de requête pour renvoyer uniquement le dernier test effectué. La valeur par défaut est `true`

  Type : booléen

  Obligatoire : non

`history`

- Paramètre de requête pour inclure l'historique des tests dans la réponse. La valeur par défaut est `true`. Régler sur `false` pour exclure l'historique

  Type : booléen

Obligatoire : non

Réponse

Nom	Description
`testId`	L'identifiant unique du test
`testName`	Le nom du test
`testDescription`	Description du test
`testType`	Le type de test qui est exécuté (par exemple`simple`,`jmeter`)
`fileType`	Type de fichier chargé (par exemple, `none`,`script`,`zip`)
`tags`	Un tableau de chaînes pour classer les tests
`status`	État du test
`startTime`	L'heure et la date du début du dernier test
`endTime`	L'heure et la date de fin du dernier test
`testScenario`	La définition du test, y compris la simultanéité, la durée du test, l'hôte et la méthode du test
`taskCount`	Le nombre de tâches nécessaires pour exécuter le test
`taskIds`	Une liste de tâches IDs pour exécuter des tests
`results`	Les résultats finaux du test
`history`	Une liste des résultats finaux des tests passés (exclus lorsque`history=false`)

Nom	Description
`totalRuns`	Le nombre total de tests pour ce scénario
`lastRun`	L'horodatage du dernier test
`errorReason`	Un message d'erreur généré lorsqu'une erreur se produit
`nextRun`	La prochaine exécution planifiée (par exemple,`2017-04-22 17:18:00`)
`scheduleRecurrence`	La récurrence du test (par exemple,,`daily`, `weeklybiweekly`,`monthly`)

POST /scenarios/ {testId}

Description

L'`POST /scenarios/{testId}`opération permet d'annuler un scénario de test spécifique.

Paramètre de demande

`testId`

- L'identifiant unique du test

  Type : String

  Obligatoire : oui

Réponse

Nom	Description
`status`	État du test

SUPPRIMER /scenarios/ {testId}

Description

L'`DELETE /scenarios/{testId}`opération permet de supprimer toutes les données relatives à un scénario de test spécifique.

Paramètre de demande

`testId`

- L'identifiant unique du test

  Type : String

  Obligatoire : oui

Réponse

Nom	Description
`status`	État du test

OPTIONS /scénarios/ {testId}

Description

L'`OPTIONS /scenarios/{testId}`opération fournit une réponse à la demande avec les en-têtes de réponse CORS corrects.

Réponse

Nom	Description
`testId`	L'identifiant unique du test
`testName`	Le nom du test

Nom	Description
`testDescription`	Description du test
`testType`	Le type de test qui est exécuté (par exemple`simple`,`jmeter`)
`fileType`	Type de fichier chargé (par exemple, `none`ou`script`,`zip`)
`status`	État du test
`startTime`	L'heure et la date du début du dernier test
`endTime`	L'heure et la date de fin du dernier test
`testScenario`	La définition du test, y compris la simultanéité, la durée du test, l'hôte et la méthode du test
`taskCount`	Le nombre de tâches nécessaires pour exécuter le test
`taskIds`	Une liste de tâches IDs pour exécuter des tests
`results`	Les résultats finaux du test
`history`	Une liste des résultats finaux des tests passés
`errorReason`	Un message d'erreur généré lorsqu'une erreur se produit

GET /scenarios/ {testId} /testruns

Description

L'`GET /scenarios/{testId}/testruns`opération récupère le test exécuté IDs pour un scénario de test spécifique, éventuellement filtré par plage de temps. When`latest=true`, renvoie uniquement le test le plus récent.

Paramètres de demande

`testId`

- L'ID du scénario de test

  Type : String

  Obligatoire : oui

`latest`

- Renvoie uniquement l'ID de test le plus récent

  Type : booléen

  Par défaut: `false`

  Obligatoire : non

`start_timestamp`

- Horodatage ISO 8601 à partir duquel filtrer les essais (inclus). Par exemple, `2024-01-01T00:00:00Z`

  Type : chaîne (format date-heure)

  Obligatoire : non

`end_timestamp`

- Horodatage ISO 8601 pour filtrer les essais jusqu'au (inclus). Par exemple, `2024-12-31T23:59:59Z`

  Type : chaîne (format date-heure)

  Obligatoire : non

`limit`

- Nombre maximum de tests à renvoyer (ignoré lorsque`latest=true`)

  Type : entier (minimum : 1, maximum : 100)

  Par défaut: `20`

  Obligatoire : non

`next_token`

- Jeton de pagination de la réponse précédente pour accéder à la page suivante

  Type : chaîne

  Obligatoire : non

Réponse

200 - Succès

Nom	Description
`testRuns`	Tableau d'objets de test, chacun contenant `testRunId` (chaîne) et `startTime` (date-heure ISO 8601)
`pagination`	Objet contenant `limit` (entier) et `next_token` (chaîne ou valeur nulle). Le jeton est nul s'il n'y a plus de résultats

Réponses aux erreurs

- `400`- Format ou paramètres d'horodatage non valides
- `404`- Scénario de test introuvable
- `500`- Erreur interne du serveur

Exemple d’utilisation

- Dernier test effectué uniquement : `GET /scenarios/test123/testruns?latest=true`
- Plus récent dans l'intervalle de temps : `GET /scenarios/test123/testruns?latest=true&start_timestamp=2024-01-01T00:00:00Z`
- Demande de page suivante : `GET /scenarios/test123/testruns?limit=20&next_token=eyJ0ZXN0SWQiOiJzZVFVeTEyTEtMIiwic3RhcnRUaW1lIjoiMjAyNC0wMS`

GET /scenarios/ {testId} /testruns/ {} testRunId

Description

L'`GET /scenarios/{testId}/testruns/{testRunId}`opération récupère les résultats complets et les métriques d'un essai spécifique. Vous pouvez éventuellement omettre les résultats de l'historique `history=false` pour une réponse plus rapide.

Paramètres de demande

`testId`

- L'ID du scénario de test

  Type : String

  Obligatoire : oui

`testRunId`

- L'ID de cycle de test spécifique

  Type : String

  Obligatoire : oui

`history`

- Incluez un tableau d'historique en réponse. Réglez sur `false` pour omettre l'historique afin d'accélérer la réponse

  Type : booléen

  Par défaut: `true`

  Obligatoire : non

Réponse

200 - Succès

Nom	Description
`testId`	L'identifiant unique du test (par exemple,`seQUy12LKL`)
`testRunId`	L'ID d'exécution de test spécifique (par exemple,`2DEwHItEne`)
`testDescription`	Description du test de charge
`testType`	Le type de test (par exemple`simple`,`jmeter`)
`status`	État du test :`complete`, `running``failed`, ou `cancelled`
`startTime`	L'heure et la date de début du test (par exemple,`2025-09-09 21:01:00`)
`endTime`	L'heure et la date de fin du test (par exemple,`2025-09-09 21:18:29`)
`succPercent`	Pourcentage de réussite (par exemple,`100.00`)
`testTaskConfigs`	Tableau d'objets de configuration de tâches contenant `region``taskCount` , et `concurrency`
`completeTasks`	Objet mappant les régions au nombre de tâches achevées
`results`	Objet contenant des métriques détaillées, notamment `avg_lt` (latence moyenne), les percentiles (`p0_0`,`p50_0`,`p90_0`,`p95_0`,`p99_0`,`p99_9`,`p100_0`), `avg_rt` (temps de réponse moyen), `avg_ct` (temps de connexion moyen), `stdev_rt` (temps de réponse à l'écart type),`concurrency` ,`throughput` , `succ` (nombre de réussites), `fail` (nombre d'échecs),`bytes`,

Nom	Description
	`testDuration metricS3Location` , `rc` (tableau de codes de réponse) et tableau `labels`
`testScenario`	Objet contenant la configuration de test avec `execution` les `scenarios` propriété `sreporting` ,, et
`history`	Tableau des résultats de tests historiques (exclus lorsque`history=false`)

Réponses aux erreurs

- `400`- TestID non valide ou testRunId
- `404`- Le test est introuvable
- `500`- Erreur interne du serveur

SUPPRIMER /scenarios/ {testId} /testruns/ {} testRunId

Description

L'`DELETE /scenarios/{testId}/testruns/{testRunId}`opération supprime toutes les données et tous les artefacts liés à un essai spécifique. Les données de test sont supprimées de DynamoDB, tandis que les données de test réelles dans S3 restent inchangées.

Paramètres de demande

`testId`

- L'ID du scénario de test

  Type : String

  Obligatoire : oui

`testRunId`

- L'ID de test spécifique à supprimer

Type : String

Obligatoire : oui

Réponse

204 - Succès

Le test a été supprimé avec succès (aucun contenu renvoyé)

Réponses aux erreurs

- `400`- TestID non valide ou testRunId
- `403`- Interdit : autorisations insuffisantes pour supprimer le test
- `404`- Le test est introuvable
- `409`- Conflit : le test est en cours et ne peut pas être supprimé
- `500`- Erreur interne du serveur

GET /scenarios/ {testId} /baseline

Description

L'`GET /scenarios/{testId}/baseline`opération récupère le résultat du test de référence désigné pour un scénario. Renvoie soit l'ID d'exécution du test de référence, soit les résultats complets de la ligne de base en fonction du `data` paramètre.

Paramètres de demande

`testId`

- L'ID du scénario de test

  Type : String

  Obligatoire : oui

`data`

- Renvoie les données complètes du cycle de test de référence si`true`, sinon, uniquement testRunId

Type : booléen

Par défaut: `false`

Obligatoire : non

Réponse

200 - Succès

Quand `data=false` (par défaut) :

Nom	Description
`testId`	L'ID du scénario de test (par exemple,`seQUy12LKL`)
`baselineTestRunId`	L'ID d'exécution du test de référence (par exemple,`2DEwHItEne`)

Quand `data=true` :

Nom	Description
`testId`	L'ID du scénario de test (par exemple,`seQUy12LKL`)
`baselineTestRunId`	L'ID d'exécution du test de référence (par exemple,`2DEwHItEne`)
`baselineData`	Objet complet des résultats de test (même structure que`GET /scenarios/{testId }/testruns/{testRunId}`)

Réponses aux erreurs

- `400`- Paramètre TestID non valide

- `404`- Scénario de test introuvable ou aucun scénario de référence défini
- `500`- Erreur interne du serveur

PUT /scenarios/ {testID} /baseline

Description

L'`PUT /scenarios/{testId}/baseline`opération désigne un essai spécifique comme base de référence pour la comparaison des performances. Une seule référence peut être définie par scénario.

Paramètres de demande

`testId`

- L'ID du scénario de test

  Type : String

  Obligatoire : oui

Corps de la demande

Nom	Description
`testRunId`	L'ID d'exécution du test à définir comme référence (par exemple,`2DEwHItEne`)

Réponse

200 - Succès

Nom	Description
`message`	Message de confirmation (par exemple,`Baseline set successfully`)
`testId`	L'ID du scénario de test (par exemple,`seQUy12LKL`)

Nom	Description
`baselineTestRunId`	L'ID d'exécution du test de référence qui a été défini (par exemple,`2DEwHItEne`)

Réponses aux erreurs

- `400`- TestID non valide ou testRunId
- `404`- Scénario de test ou exécution de test introuvable
- `409`- Conflit : le test ne peut pas être défini comme référence (par exemple, échec du test)
- `500`- Erreur interne du serveur

SUPPRIMER /scenarios/ {testId} /baseline

Description

L'`DELETE /scenarios/{testId}/baseline`opération efface la valeur de référence d'un scénario en la définissant sur une chaîne vide.

Paramètres de demande

`testId`

- L'ID du scénario de test

  Type : String

  Obligatoire : oui

Réponse

204 - Succès

La ligne de base a été correctement effacée (aucun contenu renvoyé)

Réponses aux erreurs

- `400`- Identifiant de test non valide

- `500`- Erreur interne du serveur

GET /tâches

Description

L'`GET /tasks`opération vous permet de récupérer une liste des tâches Amazon Elastic Container Service (Amazon ECS) en cours d'exécution.

Réponse

Nom	Description
`tasks`	Une liste de tâches IDs pour exécuter des tests

OPTIONS /tâches

Description

L'opération `OPTIONS /tasks` des tâches fournit une réponse à la demande avec les en-têtes de réponse CORS corrects.

Réponse

Nom	Description
`taskIds`	Une liste de tâches IDs pour exécuter des tests

GET /régions

Description

L'`GET /regions`opération vous permet de récupérer les informations sur les ressources régionales nécessaires pour exécuter un test dans cette région.

Réponse

Nom	Description
`testId`	L'ID de région
`ecsCloudWatchLogGroup`	Le nom du groupe de CloudWatch journaux Amazon pour les tâches Amazon Fargate dans la région
`region`	La région dans laquelle se trouvent les ressources du tableau
`subnetA`	L'ID de l'un des sous-réseaux de la région
`subnetB`	L'ID de l'un des sous-réseaux de la région
`taskCluster`	Le nom du cluster AWS Fargate de la région
`taskDefinition`	L'ARN de la définition de tâche dans la région
`taskImage`	Le nom de l'image de la tâche dans la région
`taskSecurityGroup`	L'ID du groupe de sécurité dans la région

OPTIONS /régions

Description

L'`OPTIONS /regions`opération fournit une réponse à la demande avec les en-têtes de réponse CORS corrects.

Réponse

Nom	Description
`testId`	L'ID de région

Nom	Description
`ecsCloudWatchLogGroup`	Le nom du groupe de CloudWatch journaux Amazon pour les tâches Amazon Fargate dans la région
`region`	La région dans laquelle se trouvent les ressources du tableau
`subnetA`	L'ID de l'un des sous-réseaux de la région
`subnetB`	L'ID de l'un des sous-réseaux de la région
`taskCluster`	Le nom du cluster AWS Fargate de la région
`taskDefinition`	L'ARN de la définition de tâche dans la région
`taskImage`	Le nom de l'image de la tâche dans la région
`taskSecurityGroup`	L'ID du groupe de sécurité dans la région

Augmenter les ressources en conteneurs

Pour augmenter le nombre d'utilisateurs virtuels simultanés (simultanéité) que vos tests de charge peuvent simuler, vous devez augmenter les ressources de processeur et de mémoire allouées à chaque tâche Amazon ECS. Cela implique de créer une nouvelle révision de définition de tâche avec des limites de ressources plus élevées, puis de mettre à jour la configuration DynamoDB de la solution afin d'utiliser la nouvelle définition de tâche pour les futurs tests.

Création d'une nouvelle révision de définition de tâche

Procédez comme suit pour créer une nouvelle définition de tâche avec des ressources de processeur et de mémoire accrues :

1. Connectez-vous à la console Amazon Elastic Container Service.
2. Dans le menu de navigation de gauche, sélectionnez Définitions de tâches.
3. Cochez la case à côté de la définition de tâche correspondant à cette solution. Par exemple, `[replaceable]` <stackName>`- EcsTaskDefinition -*<system-generated-random-Hash>*.

4. Choisissez Créer une révision.
5. Sur la page Créer une nouvelle révision, effectuez les actions suivantes :
   a. Sous Taille de la tâche, modifiez la mémoire des tâches et le processeur des tâches selon les valeurs souhaitées. Les valeurs les plus élevées autorisent un plus grand nombre d'utilisateurs virtuels simultanés par tâche.
   b. Sous Définitions des conteneurs, passez en revue les limites de mémoire matérielle/logicielle. Si cette limite est inférieure à la mémoire souhaitée, choisissez le conteneur.
   c. Dans la boîte de dialogue Modifier le conteneur, accédez à Limites de mémoire et mettez à jour la limite stricte pour qu'elle corresponde ou soit inférieure à l'allocation de mémoire de votre tâche.
   d. Choisissez Mettre à jour.
6. Sur la page Créer une nouvelle révision, choisissez Créer.
7. Une fois la définition de tâche créée avec succès, enregistrez l'ARN complet de la définition de tâche, y compris le numéro de version. Par exemple : `[replaceable]` <stackName>`- EcsTaskDefinition - *`<system-generated-random-Hash>`* : [remplaçable]`<system-generated-versionNumber>`.

Mettre à jour la table DynamoDB

Après avoir créé la nouvelle révision de définition de tâche, vous devez mettre à jour la table DynamoDB de la solution afin que les futurs tests utilisent la nouvelle définition de tâche. Répétez ces étapes pour chaque région AWS dans laquelle vous souhaitez utiliser la définition de tâche mise à jour :

1. Accédez à la console DynamoDB.
2. Dans le volet de navigation de gauche, sélectionnez Explorer les éléments sous Tables.
3. Sélectionnez la table `scenarios-table` DynamoDB associée à cette solution. Par exemple, `[replaceable]` <stackName>`- DLTTest RunnerStorage DLTScenarios Table-*`<system-generated-random-Hash>`*.
4. Sélectionnez l'élément correspondant à la région dans laquelle vous avez créé la nouvelle révision de définition de tâche. Par exemple, `region-[replaceable]` <region-name>`.
5. Dans l'éditeur d'éléments, recherchez l'attribut TaskDefinition et mettez à jour sa valeur avec l'ARN de définition de tâche complet que vous avez enregistré dans la section précédente (y compris le numéro de version).

6. Sélectionnez Enregistrer les modifications.

> **Note**
>
> La définition de tâche mise à jour ne sera utilisée que pour les nouveaux tests. Tous les tests en cours d'exécution ou planifiés continueront à utiliser la définition de tâche précédente.

Spécification des outils MCP

La solution de test de charge distribué expose un ensemble d'outils MCP qui permettent aux agents d'intelligence artificielle d'interagir avec les scénarios et les résultats des tests. Ces outils fournissent des fonctionnalités abstraites de haut niveau qui correspondent à la façon dont les agents d'IA traitent les informations, ce qui leur permet de se concentrer sur l'analyse et les informations plutôt que sur des contrats d'API détaillés.

> **Note**
>
> Tous les outils MCP fournissent un accès en lecture seule aux données de la solution. Aucune modification des scénarios ou des configurations de test n'est prise en charge via l'interface MCP.

liste_scénarios

Description

L'`list_scenarios`outil extrait une liste de tous les scénarios de test disponibles avec des métadonnées de base.

Endpoint

```
GET /scenarios
```

Parameters

Aucune

Réponse

Nom	Description
`testId`	Identifiant unique pour le scénario de test
`testName`	Nom du scénario de test
`status`	État actuel du scénario de test
`startTime`	Date de création ou de dernière exécution du test
`testDescription`	Description du scénario de test

get_scenario_details

Description

L'`get_scenario_details`outil récupère la configuration de test et la dernière exécution de test pour un seul scénario de test.

Endpoint

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

Paramètre de demande

`test_id`

- L'identifiant unique du scénario de test

  Type : String

  Obligatoire : oui

Réponse

Nom	Description
`testTaskConfigs`	Configuration des tâches pour chaque région
`testScenario`	Définition et paramètres du test
`status`	État actuel du test
`startTime`	Horodatage de début du test
`endTime`	Horodatage de fin du test (s'il est terminé)

list_test_runs

Description

L'`list_test_runs`outil récupère une liste de tests pour un scénario de test spécifique, triée du plus récent au plus ancien. Renvoie un maximum de 30 résultats.

Endpoint

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

or

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

Paramètres de demande

`test_id`

- L'identifiant unique du scénario de test

  Type : String

  Obligatoire : oui

`limit`

- Nombre maximum de tests à renvoyer

Type : entier

Valeur par défaut : 20

Maximum : 30

Obligatoire : non

`start_date`

- Horodatage ISO 8601 pour filtrer les séries à partir d'une date spécifique

  Type : chaîne (format date-heure)

  Obligatoire : non

`end_date`

- Horodatage ISO 8601 pour filtrer les exécutions jusqu'à une date précise

  Type : chaîne (format date-heure)

  Obligatoire : non

Réponse

Nom	Description
`testRuns`	Tableau de résumés des essais avec indicateurs de performance et percentiles pour chaque cycle

get_test_run

Description

L'`get_test_run`outil récupère les résultats détaillés d'un seul test avec des ventilations par région et par point de terminaison.

Endpoint

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

Paramètres de demande

`test_id`

- L'identifiant unique du scénario de test

  Type : String

  Obligatoire : oui

`test_run_id`

- L'identifiant unique pour le cycle de test spécifique

  Type : String

  Obligatoire : oui

Réponse

Nom	Description
`results`	Données complètes des tests, y compris la répartition des résultats régionaux, les mesures spécifiques aux terminaux, les percentiles de performance (p50, p90, p95, p99), le nombre de réussites et d'échecs, les temps de réponse et la latence, ainsi que la configuration de test utilisée pour l'exécution

get_latest_test_run

Description

L'`get_latest_test_run`outil récupère le dernier test pour un scénario de test spécifique.

Endpoint

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

> **Note**
>
> Les résultats sont triés par heure à l'aide d'un index secondaire global (GSI), ce qui garantit que le test le plus récent est renvoyé.

Paramètre de demande

`test_id`

- L'identifiant unique du scénario de test

  Type : String

  Obligatoire : oui

Réponse

Nom	Description
`results`	Données de test les plus récentes au même format que `get_test_run`

get_baseline_test_run

Description

L'`get_baseline_test_run`outil récupère le test de référence pour un scénario de test spécifique. La base de référence est utilisée à des fins de comparaison des performances.

Endpoint

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

Paramètre de demande

`test_id`

- L'identifiant unique du scénario de test

  Type : String

  Obligatoire : oui

Réponse

Nom	Description
`baselineData`	Données de test de référence à des fins de comparaison, y compris toutes les mesures et configurations issues de l'exécution de référence désignée

get_test_run_artefacts

Description

L'`get_test_run_artifacts`outil récupère les informations du compartiment Amazon S3 pour accéder aux artefacts de test, notamment les journaux, les fichiers d'erreurs et les résultats.

Endpoint

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

Paramètres de demande

`test_id`

- L'identifiant unique du scénario de test

  Type : String

  Obligatoire : oui

`test_run_id`

- L'identifiant unique pour le cycle de test spécifique

  Type : String

  Obligatoire : oui

Réponse

Nom	Description
`bucketName`	Nom du compartiment S3 dans lequel les artefacts sont stockés
`testRunPath`	Préfixe de chemin pour le stockage actuel des artefacts (version 4.0+)
`testScenarioPath`	Préfixe de chemin pour le stockage des artefacts existants (version antérieure à 4.0)

> **Note**
>
> Tous les outils MCP exploitent les points de terminaison d'API existants. Aucune modification du APIs sous-jacent n'est requise pour prendre en charge la fonctionnalité MCP.

Référence

Cette section inclut des informations sur la collecte de données, des pointeurs vers des ressources connexes et une liste des créateurs qui ont contribué à cette solution.

Collecte des données

Cette solution envoie des métriques opérationnelles à AWS (les « données ») concernant l'utilisation de cette solution. Nous utilisons ces données pour mieux comprendre comment les clients utilisent cette solution et les services et produits associés. La collecte de ces données par AWS est soumise à l'avis de confidentialité d'AWS.

Collaborateurs

- Tom Nightingale
- Fernando Dingler
- Beomseok Lee
- George Lenz
- Erin McGill
- Dimitri López
- Kamyar Ziabari
- Bassem Wanis
- Garvit Singh
- Nikhil Reddy
- Simon Kroll
- Ahern Knox
- Ian Downard
- Owen Brady
- Jim Thario
- Thyag Ramachandran
- Yang Qin
- James Wang

Glossaire

Ce glossaire définit les acronymes et les abréviations utilisés dans le guide de mise en œuvre des tests de charge distribués sur AWS.

Protocoles et formats techniques

AGPL

Licence publique générale Affero. Une licence logicielle open source utilisée par K6.

« Hello, World! »

Interface de programmation d'applications. Ensemble de protocoles et d'outils permettant de créer des applications logicielles et de permettre la communication entre différents systèmes.

INTERFACE DE LIGNE DE COMMANDE (CLI)

Interface de ligne de commande. Interface textuelle permettant d'interagir avec les logiciels et les systèmes d'exploitation.

CORS

Partage de ressources entre origines. Fonctionnalité de sécurité qui autorise ou restreint les applications Web exécutées sur une origine à accéder à des ressources provenant d'une autre origine.

CSV

Valeurs séparées par des virgules. Format de fichier utilisé pour stocker des données tabulaires en texte brut, couramment utilisé pour l'exportation de données.

gRPC

Appel de procédure à distance gRPC. Un framework open source performant pour les appels de procédure à distance.

HTTP

Protocole de transfert hypertexte. Protocole de base utilisé pour transmettre des données sur le World Wide Web.

HTTPS

HTTP sécurisé. Extension du protocole HTTP qui utilise le chiffrement pour sécuriser les communications sur un réseau.

JSON

JavaScript Notation d'objets. Format d'échange de données léger, facile à lire et à écrire pour les humains et facile à analyser et à générer pour les machines.

JWT

Jeton Web JSON. Un moyen compact et sécurisé pour les URL de représenter les demandes à transférer entre deux parties à des fins d'authentification et d'autorisation.

OAuth

Autorisation ouverte. Norme ouverte de délégation d'accès couramment utilisée pour l'authentification et l'autorisation basées sur des jetons.

REST

Transfert d'État représentatif. Style architectural permettant de concevoir des applications en réseau utilisant une communication sans état et des méthodes HTTP standard.

SSE

Événements envoyés par le serveur. Technologie push du serveur permettant à un client de recevoir des mises à jour automatiques d'un serveur via une connexion HTTP.

UI

Interface utilisateur. Les éléments visuels et les commandes par le biais desquels les utilisateurs interagissent avec les applications logicielles.

URL

Localisateur de ressources uniforme. Adresse utilisée pour accéder aux ressources sur Internet.

xml

Langage de balisage extensible. Langage de balisage qui définit les règles de codage des documents dans un format à la fois lisible par l'homme et lisible par machine.

Termes de test et de base de données

FTP

Protocole de transfert de fichiers. Protocole réseau standard utilisé pour transférer des fichiers entre un client et un serveur.

GSI

Index secondaire mondial. Fonctionnalité DynamoDB qui permet d'interroger des données à l'aide d'une clé alternative.

JDBC

Connectivité de base de données Java. Une API Java pour connecter et exécuter des requêtes avec des bases de données.

JMS

Service de messagerie Java. Une API Java pour envoyer des messages entre deux clients ou plus.

TPS

Transactions par seconde. Mesure du nombre de transactions qu'un système peut traiter en une seconde.

AWS et termes du système

ARN

Amazon Resource Name. Identifiant unique pour les ressources AWS utilisé pour spécifier les ressources des services AWS.

ISO

Organisation internationale de normalisation. Une organisation non gouvernementale indépendante qui développe des normes internationales. Référencé dans ce guide pour le format d'horodatage ISO 8601.

SLA

Contrat de niveau de service. Engagement entre un fournisseur de services et un client qui définit le niveau de service attendu.

UUID

Identifiant unique universel. Numéro à 128 bits utilisé pour identifier de manière unique les informations contenues dans les systèmes informatiques.

vCPU

Unité centrale de traitement virtuelle. Processeur virtuel attribué à une machine virtuelle ou à un conteneur, représentant une partie de la puissance de traitement du processeur physique.

Termes relatifs aux tests de charge

simultanéité

Nombre d'utilisateurs virtuels simultanés par tâche. Ce paramètre contrôle le nombre d'utilisateurs simulés générés par chaque tâche Fargate lors d'un test de charge.

pile régionale

Une CloudFormation pile déployée dans une région AWS pour fournir une infrastructure de test pour les tests de charge multirégionaux.

nombre de tâches

Le nombre de conteneurs Fargate (tâches) lancés pour exécuter un scénario de test. La charge totale générée est égale au nombre de tâches multiplié par la simultanéité.

scénario de test

Test de charge configuré comprenant le type de test, les points de terminaison cibles, le nombre de tâches, la simultanéité, la durée et d'autres paramètres.

Révisions

Consultez le fichier ChangeLog.md dans notre GitHub référentiel pour suivre les améliorations et les correctifs spécifiques à chaque version.

Avis

Il incombe aux clients de procéder à une évaluation indépendante des informations contenues dans le présent document. Ce document : (a) est fourni à titre informatif uniquement, (b) représente les offres de produits et les pratiques actuelles d'AWS, qui sont susceptibles d'être modifiées sans préavis, et (c) ne crée aucun engagement ni aucune garantie de la part d'AWS et de ses filiales, fournisseurs ou concédants de licence. Les produits ou services AWS sont fournis « tels quels » sans garanties, déclarations ou conditions d'aucune sorte, qu'elles soient explicites ou implicites. Les responsabilités et obligations d'AWS à l'égard de ses clients sont régies par les accords AWS, et le présent document ne fait partie d'aucun accord conclu entre AWS et ses clients et ne les modifie pas.

Les tests de charge distribués sur AWS sont fournis sous licence selon les termes de la licence Apache version 2.0 disponible auprès de l'Apache Software Foundation.

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.