

Guide de mise en œuvre

Tests de charge distribués sur AWS



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

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Les marques commerciales et la présentation commerciale d'Amazon ne peuvent pas être utilisées en relation avec un produit ou un service extérieur à Amazon, d'une manière susceptible d'entraîner une confusion chez les clients, ou d'une manière qui dénigre ou discrédite Amazon. Toutes les autres marques commerciales qui ne sont pas la propriété d'Amazon appartiennent à leurs propriétaires respectifs, qui peuvent ou non être affiliés ou connectés à Amazon, ou sponsorisés par Amazon.

Table of Contents

Présentation de la solution	1
Fonctionnalités	2
Avantages	3
Cas d'utilisation	4
Concepts et définitions	5
Présentation de l'architecture	7
Diagramme d'architecture	7
Considérations relatives à la conception d'AWS Well-Architected	8
Excellence opérationnelle	8
Sécurité	9
Fiabilité	10
Efficacité des performances	10
Optimisation des coûts	10
Durabilité	11
Détails de l'architecture	12
Partie avant	12
API de test de charge	12
console Web	12
Backend	13
Pipeline d'images de conteneurs	13
Infrastructure de test	13
Moteur d'essai de charge	14
Services AWS inclus dans cette solution	14
Comment fonctionnent les tests de charge distribués sur AWS	16
Considérations relatives à la conception	18
Applications prises en charge	18
JMeter prise en charge des scripts	18
Prise en charge des scripts K6	19
Support des scripts Locust	19
Planification des tests	19
Tests simultanés	20
Gestion des utilisateurs	20
Déploiement régional	21
Planifiez votre déploiement	22

Coût	22
Sécurité	23
Rôles IAM	24
Amazon CloudFront	24
Groupe de sécurité AWS Fargate	24
Test de stress du réseau	24
Restreindre l'accès à l'interface utilisateur publique	25
Régions AWS prises en charge	25
Quotas	26
Quotas pour les services AWS dans cette solution	26
CloudFormation Quotas AWS	26
Quotas de test de charge	26
Tests simultanés	20
Politique d'Amazon en matière de EC2 tests	27
Politique d'Amazon en matière de tests de CloudFront charge	27
Surveillance de la solution après le déploiement	28
Activation ou configuration des CloudWatch alarmes	28
Déployez la solution	29
Vue d'ensemble du processus de déploiement	29
CloudFormation Modèle AWS	29
Lancement de la pile	30
Déploiement multirégional	33
Mettre à jour la solution	37
Lors de la mise à jour de versions DLT antérieures à v3.2.6 vers v3.3.x et v3.3.x vers les versions les plus récentes, la mise à jour de la pile échoue	38
Résolution des problèmes	40
Résolution des problèmes connus	40
Contacter AWS Support	42
Créer un dossier	42
Comment pouvons-nous vous aider ?	43
Informations supplémentaires	43
Aidez-nous à résoudre votre cas plus rapidement	43
Résolvez maintenant ou contactez-nous	43
Désinstallez la solution	44
Utilisation de la AWS Management Console	44
Utilisation de l'interface de ligne de commande AWS	44

Supprimer les compartiments Amazon S3	44
Utilisez la solution	46
Résultats du test	46
Flux de travail de planification des tests	47
Déterminer le nombre d'utilisateurs	47
Données en direct	48
Flux de travail d'annulation des tests	49
Manuel du développeur	50
Code source	50
Maintenance	50
Versions	50
Personnalisation de l'image du conteneur	51
API de test de charge distribuée	58
GET /scenarios	60
POST /scénarios	60
OPTIONS/SCÉNARIOS	62
OBTENEZ /scenarios/ {testId}	62
POST /scenarios/ {testId}	64
SUPPRIMER /scenarios/ {testId}	64
OPTIONS /scénarios/ {testId}	65
GET /tâches	66
OPTIONS /tâches	66
GET /régions	67
OPTIONS /régions	68
Augmenter les ressources en conteneurs	68
Création d'une nouvelle révision de définition de tâche	69
Mettre à jour la table DynamoDB	69
Référence	70
Collecte de données anonymisée	70
Collaborateurs	71
Révisions	72
Avis	73
.....	lxxiv

Automatisez les tests de vos applications logicielles à grande échelle

Date de publication : novembre 2019

Les tests de charge distribués sur AWS vous permettent d'automatiser les tests de vos applications logicielles à grande échelle et au chargement afin d'identifier les goulots d'étranglement avant de publier votre application. Cette solution crée et simule des milliers d'utilisateurs connectés qui génèrent des enregistrements transactionnels à un rythme constant, 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 capables d'exécuter toutes vos simulations et offre les fonctionnalités suivantes :

- Déployez Amazon ECS sur des conteneurs Fargate AWS qui peuvent s'exécuter indépendamment pour tester les capacités de charge du logiciel testé.
- Simulez des dizaines de milliers d'utilisateurs connectés, dans plusieurs régions AWS, en générant des enregistrements transactionnels à un rythme soutenu.
- Personnalisez les tests de votre application en créant des [JMeter scripts](#) personnalisés.
- Planifiez les tests de charge pour qu'ils commencent automatiquement à une date future ou à des dates récurrentes.
- Exécutez les tests de chargement de votre application simultanément ou exécutez plusieurs tests simultanément.

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
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 :

Out-of-the-Box Tests de performances configurables

Inclut des tests de performance préconfigurés disponibles pour une utilisation immédiate.

Tests d'applications personnalisables

Permet une personnalisation flexible et précise des tests afin d'identifier les problèmes potentiels et d'adapter les tests à des exigences et à des scénarios spécifiques à l'aide de JMeter scripts.

Simule une charge utilisateur élevée

Capable de simuler des dizaines de milliers d'utilisateurs connectés pour tester le stress de votre application.

Génération continue de transactions

Génère des enregistrements transactionnels en continu pour évaluer les performances sous charge constante.

Surveillance en temps réel

Fournit un suivi en temps réel de la progression et des résultats des tests et vous permet de planifier les tests pour qu'ils démarrent automatiquement à des dates spécifiées ou à des intervalles récurrents.

Simulation de demandes régionales

Simulez les demandes des utilisateurs de n'importe quelle région pour évaluer les performances globales.

Flexibilité des terminaux

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

Résultats de test détaillés

Consultez les résultats complets des tests, notamment le temps de réponse moyen, le nombre d'utilisateurs simultanés, les demandes réussies et les demandes ayant échoué.

Console Web intuitive

Offre une console easy-to-use Web pour la gestion et le suivi des tests.

Supporte plusieurs protocoles

Compatible avec divers protocoles tels que HTTP WebSocket, HTTPS, JDBC, JMS, FTP et gRPC.

Avantages

La solution offre les avantages suivants :

Prend en charge les tests de performance complets

Facilite les tests de charge, de stress et d'endurance pour une évaluation approfondie des applications.

Détection précoce des problèmes de performance

Identifie les problèmes de performances et les goulots d'étranglement avant la mise en production.

Simulation d'utilisation dans le monde réel

Reflète avec précision les modèles d'utilisation réels pour mettre en évidence les goulots d'étranglement et les domaines d'optimisation.

Informations détaillées sur les Performances

Fournit des informations sur les performances et la résilience des logiciels sous une charge importante.

Évaluation automatisée des performances

Permet des évaluations régulières des performances sans intervention manuelle.

Tests rentables

Propose un pay-as-you-go modèle qui élimine le besoin d'une infrastructure de test dédiée et de frais d'abonnement.

Cas d'utilisation

Simuler la charge de production

Testez les applications Web et mobiles dans des conditions similaires à celles de la production avant de lancer une nouvelle version.

Valider les performances des applications

Assurez-vous que votre application peut gérer le trafic utilisateur attendu sans dégradation.

Testez les limites des applications à l'aide des ressources par défaut et évaluez l'évolutivité de l'infrastructure.

Gérez les pics de charge

Vérifiez que votre infrastructure peut gérer les pics de charge ou les pics de trafic inattendus, garantissant ainsi la stabilité en cas de forte demande.

Optimisez les performances

Comprenez le profil de performance de votre application et identifiez les obstacles tels que l'exécution inefficace du code, les requêtes de base de données et la latence du réseau.

Démarrage rapide des tests

Commencez à tester rapidement avec des tests de out-of-the-box performance.

Tests personnalisables

Adaptez les tests à des scénarios et à des exigences spécifiques, en ajustant le nombre d'utilisateurs simultanés et les tâches lancées.

Tests planifiés

Planifiez des tests pour les tests de régression et la surveillance continue des performances, afin de garantir des performances cohérentes des applications.

Évaluation de la performance géographique

Évaluez les performances des applications dans différentes régions géographiques pour garantir une efficacité globale.

Intégration du pipeline CI/CD

Intégrez les tests de performance dans votre CI/CD pipeline pour des tests transparents et automatisés pendant les cycles de développement.

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

Le nombre d'utilisateurs virtuels générés simultanément par tâche. La limite recommandée sur la base des paramètres par défaut est de 200 utilisateurs virtuels. La simultanéité est limitée par le processeur et la mémoire. Pour les tests basés sur Apache JMeter, plus le nombre d'utilisateurs virtuels est élevé, plus la mémoire utilisée par la JVM sur la tâche ECS est importante. La définition de tâche ECS par défaut crée des tâches avec 4 Go de mémoire. Il est recommandé de commencer par des valeurs d'utilisateurs virtuels 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

Le temps nécessaire pour atteindre la simultanéité cible.

hold-for

Il est temps de maintenir la simultanéité de la cible.

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, qui 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 exécuter des scénarios de test.
4. [Une topologie de réseau Amazon Virtual Private Cloud \(Amazon VPC\) est déployée. Elle contient les conteneurs Amazon Elastic Container Service \(Amazon ECS\) de la solution exécutés sur AWS Fargate.](#)
5. Les conteneurs incluent l'image de [conteneur conforme à l'Open Container Initiative AmazonLinux](#)(OCI) (lorsque le framework de test de charge Blazemeter est installé), qui est utilisée pour générer de la charge afin de tester les performances de votre application. Taurus/Blazemeter est un framework d'automatisation des tests open source. L'image du conteneur est hébergée par AWS dans un référentiel public [Amazon Elastic Container Registry](#) (Amazon ECR). Pour plus d'informations sur le référentiel d'images ECR, reportez-vous à la section [Personnalisation des images du conteneur](#).

6. Une console Web alimentée par [AWS Amplify](#) est déployée dans un compartiment Amazon S3 configuré pour un 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, cette solution crée également un rôle d'administrateur de solution 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 et à l'API du testeur de charge distribué.
10. Après avoir déployé cette solution, vous pouvez utiliser la console Web pour créer un scénario de test qui définit une série de tâches.
11. Les microservices utilisent ce scénario de test pour exécuter des tâches Amazon ECS sur AWS Fargate dans les régions spécifiées.
12. [Outre le stockage des résultats dans Amazon S3 et DynamoDB, une fois le test terminé, la sortie est enregistrée dans Amazon. CloudWatch](#)
13. Si vous sélectionnez l'option Live Data, la solution envoie les CloudWatch journaux Amazon relatifs aux tâches AWS Fargate à une fonction Lambda pendant le test, pour chaque région dans laquelle le test a été effectué.
14. La fonction Lambda publie ensuite les données dans la rubrique correspondante dans [AWS IoT Core](#) dans la région où la pile principale a été déployée. La console Web s'abonne à la rubrique et vous pouvez consulter les données pendant l'exécution du test dans la console Web.

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](#).

- Ressources définies comme une infrastructure utilisant du code CloudFormation.
- La solution transmet des métriques à Amazon CloudWatch à différentes étapes afin de garantir l'observabilité de l'infrastructure : fonctions Lambda, tâches Amazon ECS, compartiments Amazon S3 et 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](#).

- Amazon Cognito authentifie et autorise les utilisateurs de l'application d'interface utilisateur Web.
- Toutes les communications interservices utilisent les rôles [AWS Identity and Access Management \(IAM\)](#) applicables.
- Tous les rôles utilisés par la solution suivent le principe du moindre privilège d'accès. Ils contiennent uniquement les autorisations minimales requises pour accomplir la tâche.
- Tout le stockage de données, y compris les compartiments S3, crypte les données au repos.
- Un groupe d'utilisateurs Amazon Cognito gère l'accès des utilisateurs à la console et aux points de terminaison API Gateway du testeur de charge distribué.
- La journalisation, le suivi et le versionnement sont activés le cas échéant.
- L'accès au réseau est privé par défaut, les points de terminaison [Amazon Virtual Private Cloud \(Amazon VPC\)](#) étant activés lorsqu'ils sont disponibles.

Note

Les tests de charge distribués sur AWS créent plusieurs groupes de CloudWatch journaux en fonction du service utilisé. La période de conservation des journaux varie en fonction du stockage et du coût du volume d'événements de journal générés. La solution crée des journaux d'aperçu des conteneurs pour les tâches ECS ; la durée de conservation de ces journaux est configurée pour un jour. Les journaux créés pour les services AWS Step Functions, les journaux personnalisés ECS Load Testing et les journaux API Gateway sont configurés avec une période de conservation d'un an. La durée de conservation des journaux d'exécution AWS Lambda est configurée sur 2 ans. Les journaux d'exécution d'API Gateway sont conservés de manière à ne jamais expirer. Vous pouvez modifier les périodes de conservation des journaux en fonction de vos besoins dans la CloudWatch console.

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.

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 de les augmenter ou de les réduire 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 deux composants de haut niveau : un [front-end](#) et un [backend](#).

Partie avant

Le front-end comprend une API de test de charge et une console Web que vous utilisez pour interagir avec le backend de la solution.

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 les données de test en toute sécurité via la console Web et RESTful l'API incluses. 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 hébergée sur 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.

La console Web est conçue pour montrer comment vous pouvez interagir avec cette solution de test de charge. Dans un environnement de production, nous vous recommandons de personnaliser la console Web pour répondre à vos besoins spécifiques ou de créer votre propre console.

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.

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 [AmazonLinux](#) comme image de base avec le framework de test de charge Blazemeter installé. Cette image est hébergée dans un référentiel public Amazon Elastic Container Registry (Amazon ECR). L'image est utilisée 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 modèle principal, la solution crée un modèle secondaire pour lancer les ressources nécessaires pour exécuter des tests dans plusieurs régions. Le modèle est stocké dans Amazon S3

et un lien vers le modèle est fourni dans la console Web. Les modèles secondaires créent un VPC, un cluster AWS Fargate et une fonction Lambda pour le traitement des données en direct.

Pour plus d'informations sur le lancement d'une région secondaire, 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 connectés, dans plusieurs régions, générant un certain nombre de transactions par seconde.

Vous définissez les paramètres des tâches qui seront exécutées dans le cadre du 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.

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.

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.

Service AWS	Description
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 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.
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.

Service AWS	Description
AWS Systems Manager	Soutenir. Assure la surveillance des ressources au niveau de l'application 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.

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.

Considérations relatives à la conception

Applications prises en charge

Cette solution prend en charge les applications basées sur le cloud et les applications sur site, à condition que vous disposiez d'une connexion réseau entre votre compte AWS et votre application. La solution prend en charge APIs l'utilisation du protocole HTTP ou du protocole HTTPS. Vous pouvez également contrôler les en-têtes de requête HTTP. Vous pouvez donc ajouter des en-têtes d'autorisation ou personnalisés pour transmettre des jetons ou des clés d'API.

JMeter prise en charge des scripts

Lorsque vous créez un scénario de test à l'aide de l'interface utilisateur (UI) de cette solution, vous pouvez utiliser un script de JMeter test. Après avoir sélectionné le fichier de JMeter script, celui-ci est chargé dans le compartiment `<stack-name>-scenariosbucket` Amazon Simple Storage Service (Amazon S3). Lorsque les tâches Amazon Elastic Container Service (Amazon ECS) sont exécutées, JMeter le script est téléchargé depuis le `<stack-name>compartiment Amazon S3 -scenariosbucket` et le test s'exécute.

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](#).

Prise en charge des scripts K6

La solution prend en charge les tests basés sur le framework K6. K6 est publié avec la licence [AGPL-3.0](#). La solution affiche un message d'accusé de réception de licence lors de la création d'un nouveau test pour K6. Le fichier de test K6 ainsi que tous les fichiers d'entrée nécessaires peuvent être inclus dans un fichier d'archive et téléchargés pour le scénario de test à l'aide de l'option de téléchargement.

Support des scripts Locust

La solution prend en charge les tests basés sur le framework Locust. Le fichier de test Locust ainsi que tous les fichiers d'entrée nécessaires peuvent être inclus dans un fichier d'archive et téléchargés pour le scénario de test à l'aide de l'option de téléchargement.

Planification des tests

Vous pouvez planifier des tests pour qu'ils soient exécutés à une date future ou utiliser l'option Exécuter maintenant. Vous pouvez planifier un test en une seule fois dans le futur ou configurer un test récurrent dans lequel vous spécifiez une date de première exécution et une récurrence planifiée. Les options de récurrence sont les suivantes : quotidienne, hebdomadaire, bihebdomadaire

et mensuelle. Pour plus d'informations sur le fonctionnement de la planification, reportez-vous à la section [Workflow de planification des tests](#) de ce guide.

À partir de la version 3.3.0, les tests de charge distribués sur AWS permettent aux utilisateurs de planifier des tests de charge à l'aide d'expressions cron. Sélectionnez Exécuter selon le calendrier, puis sur l'onglet CRON pour saisir manuellement une valeur cron ou utiliser les champs déroulants. Le cronExpiryDate doit correspondre à la date de test planifiée. Passez en revue les dates de la prochaine course (UTC) pour confirmer votre calendrier.

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 inclut un tableau de CloudWatch bord Amazon pour chaque test et affiche le résultat combiné de toutes les tâches exécutées pour ce test dans le cluster Amazon ECS en temps réel. Le CloudWatch tableau de bord affiche le temps de réponse moyen, le nombre d'utilisateurs simultanés, le nombre de demandes réussies et le nombre de demandes ayant échoué. Chaque métrique est agrégée par seconde et le tableau de bord est mis à jour 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 le [coût](#), la [sécurité](#), [les régions](#) et les autres considérations à 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 d'exécution de cette solution dépend du nombre de tests de charge exécutés, de la durée de ces tests de charge et de la quantité de données utilisées dans le cadre des tests. À partir de cette révision, le coût d'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	0,0015\$
	1 000 unités de capacité de lecture à la demande	
AWS Lambda	1 000 demandes	1,25\$
	Durée totale de 10 minutes	
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 faciliter la gestion des 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](#).

Important

À partir de la version 1.3.0, le processeur est augmenté à 2 vCPU et la mémoire à 4 Go. Ces modifications augmentent le coût estimé par rapport aux versions précédentes de cette solution. Si vos tests de charge ne nécessitent pas ces augmentations de vos ressources AWS, vous pouvez les réduire. 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.

Les prix sont susceptibles d'être modifiés. Pour plus de détails, consultez la page Web de tarification de chaque service AWS que vous utiliserez dans cette solution.

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. Pour réduire le temps de latence et améliorer la sécurité, cette solution inclut une CloudFront distribution dotée d'une identité d'accès d'origine, c'est-à-dire 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](#).

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.

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)

Nom de la région	
Asie-Pacifique (Singapour)	Europe (Stockholm)
Asie-Pacifique (Sydney)	Amérique du Sud (São Paulo)

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' AmazonLinux image du conteneur image (avec Blazemeter 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 inclut un tableau de CloudWatch bord Amazon pour chaque test et affiche le résultat combiné de toutes les tâches exécutées pour ce test dans le cluster Amazon ECS en temps réel. Le CloudWatch tableau de bord affiche le temps de réponse moyen, le nombre d'utilisateurs simultanés, le nombre de demandes réussies et le nombre de demandes ayant échoué. Chaque métrique est agrégée par seconde et le tableau de bord est mis à jour 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 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

Activation ou configuration des CloudWatch alarmes

Il est recommandé de surveiller en permanence les ressources de la solution après le déploiement. Vous pouvez envisager de configurer [CloudWatch des métriques](#) pour les ressources créées par la solution.

Métriques CloudFront de distribution Amazon

Passez en revue les [métriques de CloudFront distribution](#) et configurez des alarmes pour surveiller et recevoir des notifications.

Métriques Amazon API Gateway

Passez en revue les [dimensions et les métriques d'Amazon API Gateway](#) et configurez des alarmes pour surveiller et recevoir des notifications.

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

Suivez les step-by-step instructions de cette section pour configurer et déployer la solution dans votre compte.

Avant de lancer la solution, examinez le [coût](#), [l'architecture](#), la [sécurité du réseau](#) et les autres considérations abordées précédemment dans ce guide.

Temps de déploiement : environ 15 minutes

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 :

[load-testing-on-aws.template](#) - Utilisez ce modèle pour lancer la solution et tous les composants distribués.

[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

Important

Si vous mettez à jour la pile d'une version antérieure à la version 3.2.6 vers la dernière version, lisez [cette section](#) avant de mettre à jour la pile.

Avant de lancer le déploiement automatique, passez en revue l'architecture et les autres considérations abordées dans ce guide. Suivez les step-by-step instructions de cette section pour configurer et déployer des tests de charge distribués sur AWS dans votre compte.

Temps de déploiement : environ 15 minutes

Important

Cette solution inclut une option permettant d'envoyer des métriques opérationnelles anonymisées à 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](#).

Pour désactiver cette fonctionnalité, téléchargez le modèle, modifiez la section de CloudFormation mappage AWS, puis utilisez la CloudFormation console AWS pour télécharger votre modèle mis à jour et déployer la solution. Pour plus d'informations, consultez la section [Collecte de données anonymisée](#) de ce guide.

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 la console de gestion AWS et sélectionnez le bouton ci-dessous pour lancer le CloudFormation modèle distributed-load-testing-on -aws AWS.

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	<i><Requires input></i>	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

Paramètre	Par défaut	Description
		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.
Deuxième sous-réseau existant	<Optional input>	L'ID du deuxième sous-réseau 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
Fournir un bloc CIDR valide pour le sous-réseau A afin que la solution crée un VPC	192.168.0.0/20	Bloc CIDR pour le sous-réseau A du VPC AWS Fargate
Fournissez un bloc CIDR valide pour le sous-réseau B afin que la solution crée un VPC	192,168.16,0/20	Bloc CIDR pour le sous-réseau 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 conteneurs Amazon ECS.

Paramètre	Par défaut	Description
Mise à jour automatique de l'image du conteneur	Yes	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.

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

Vous pouvez effectuer des tests dans plusieurs régions. Lorsque vous déployez la solution de test de charge distribué, elle crée trois compartiments Amazon S3. La solution crée une pile régionale secondaire et la stocke dans le compartiment des scénarios Amazon S3.

Note

La convention de dénomination des compartiments est `<stack-name> - dltestrunnerstoragedltsenariosbucket <_[0-9][0-9]..-<[0-9][0-9].._` avec le mot clé `scenarios` dans le nom du compartiment, que vous pouvez localiser en accédant à la console S3, puis à Buckets.

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 Gérer les régions dans le menu supérieur.
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-réseau de votre VPC existant. Ce sous-réseau a besoin

Paramètre	Par défaut	Description
		d'une route vers Internet pour extraire l'image du conteneur afin d'exécuter des tests. Par exemple, subnet-7h8i9j0k.
Deuxième sous-réseau existant	<Optional input>	L'ID du deuxième sous-réseau 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 conteneurs 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, la nouvelle région est répertoriée dans le mode Gérer les régions. Vous pouvez utiliser cette région dans un test en la sélectionnant lors de la création du test. La solution crée un élément DynamoDB pour chaque région lancée dans le tableau des scénarios, qui contient

les informations nécessaires concernant les ressources de test de cette région. Vous pouvez trier les résultats des tests par région dans la console Web. En raison des contraintes liées à l'API, vous pouvez afficher les résultats agrégés de toutes les régions dans le cadre d'un test multirégional uniquement en les représentant graphiquement dans les statistiques Amazon CloudWatch . Vous pouvez trouver 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

Si vous avez déjà déployé la solution, suivez cette procédure pour mettre à jour la CloudFormation pile de la solution afin d'obtenir la dernière version du framework 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.

Lors de la mise à jour de versions DLT antérieures à v3.2.6 vers v3.3.x et v3.3.x vers les versions les plus récentes, la mise à jour de la pile échoue

1. Téléchargez le fichier [distributed-load-testing-on-aws.template](https://github.com/aws-samples/elastic-load-balancing-distributed-load-testing-on-aws-template).
2. Ouvrez le modèle et accédez à Conditions : et recherchez DLTCommonResourcesAppRegistryCondition
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 pile supprime les ressources liées au registre des applications de la pile. La mise à jour doit donc être terminée.
7. Effectuez une autre mise à jour de la pile à l'aide de l'URL du modèle le plus récent pour ajouter les ressources de l'application de registre des applications à votre pile.

Note

AWS Systems Manager Application Manager vous donne une vue d'ensemble de cette solution et de ses ressources au niveau de l'application, afin que vous puissiez :

1. Surveillez ses ressources, les coûts des ressources déployées sur les stacks et les comptes AWS, ainsi que les journaux associés à cette solution depuis un emplacement central.

2. Affichez les données opérationnelles relatives aux ressources de cette 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 contient des informations sur l'utilisation de la solution de test de charge distribué sur AWS, notamment les [résultats des tests](#), le [flux de travail de planification des tests](#) et les [données en direct](#).

Résultats du test

Les tests de charge distribués sur AWS s'appuient sur le framework de test de charge pour exécuter des tests d'applications à grande échelle. Lorsqu'un test est terminé, un rapport détaillé contenant les résultats suivants est généré.

- 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 générées par le test.
- 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.
- Percentile : percentile du temps de réponse pour le test. Le temps de réponse maximal est de 100 % ; le temps de réponse minimum est de 0 %.

Note

Les résultats des tests sont affichés dans la console. Vous pouvez consulter les fichiers XML des résultats de test bruts dans le Results dossier du compartiment Scenarios Amazon S3.

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.

Flux de travail de planification des tests

Utilisez la console Web pour planifier un test de charge. Lors de la planification d'un test, le flux de travail suivant s'exécute :

- Lorsqu'un test de charge est créé avec l'option de planification, les paramètres de planification sont envoyés à l'API de la solution via Amazon API Gateway.
- L'API transmet ensuite les paramètres à une fonction Lambda qui crée une règle d' CloudWatch événements, dont l'exécution sera planifiée à la date spécifiée.
- S'il s'agit d'un test ponctuel, la règle CloudWatch des événements s'exécute à la date spécifiée. La fonction `api-services` Lambda exécute un nouveau test via le flux de travail de test.
- S'il s'agit d'un test récurrent, la règle CloudWatch Événements s'active à la date spécifiée. La fonction `api-services` Lambda s'exécute, ce qui supprime la règle CloudWatch Events actuelle et crée une autre règle qui s'exécute immédiatement lors de sa création et de manière récurrente par la suite en fonction de la fréquence de récurrence spécifiée.

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). 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. Facultativement, les ressources du conteneur peuvent être augmentées pour permettre un plus grand nombre d'utilisateurs simultanés. Cependant, cela entraîne un coût plus élevé. Pour plus de détails, reportez-vous à la section [Augmenter les ressources en conteneurs](#) de ce guide.

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

Données en direct

Vous pouvez éventuellement inclure des données en temps réel lors de l'exécution d'un test pour obtenir des informations en temps réel sur ce qui se passe. Le groupe de CloudWatch journaux pour les tâches Fargate contient un filtre d'abonnement pour les résultats des tests incluant l'option Live Data. Si la solution trouve le modèle, elle lance une fonction Lambda qui structure les données et les publie sur une rubrique AWS IoT Core. La console Web s'abonne au sujet, reçoit les données entrantes et représente graphiquement les données agrégées à intervalles d'une seconde. La console Web contient quatre graphiques : temps de réponse moyen, utilisateurs virtuels, succès et échecs.

 Note

Les données sont éphémères et ne sont utilisées que pour voir ce qui se passe pendant le test. Une fois le test terminé, la solution stocke les données de résultats dans DynamoDB et Amazon S3. 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.

Flux de travail d'annulation des tests

Lorsque vous annulez un test de charge depuis la console Web, la solution exécute le flux de travail d'annulation du test suivant.

1. La demande d'annulation est envoyée à l'`microservicesAPI`.
2. L'`microservicesAPI` appelle la fonction `task-canceler` Lambda qui annule les tâches jusqu'à ce que toutes les tâches actuellement lancées soient arrêtées.
3. Si la fonction `task-runner` Lambda continue de s'exécuter après l'appel initial à la fonction `task-canceler` Lambda, les tâches continueront d'être lancées. 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.

Manuel 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 utilisateurs.

Maintenance

Cette solution utilise des images Docker avec des versions fixes correspondant à chaque version de solution si les mises à jour automatiques ne sont pas sélectionnées. L'équipe des solutions AWS utilise le scan amélioré ECR 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 publiera les images corrigées avec le même tag de version à résoudre CVEs, sans compromettre la compatibilité avec la version publiée de la solution. Lorsque des images sont corrigées, si elles se trouvent sur la même version mineure, la balise stable est automatiquement mise à jour et une balise d'image supplémentaire est créée dans ce format `<solution-version>_<date-of-fix>`. Si une version majeure ou mineure est publiée, une mise à jour complète sera nécessaire pour obtenir la dernière version de l'image, car la balise stable sera incrémentée afin que sa version corresponde à la version de la solution. Si vous optez pour les mises à jour automatiques, les modifications apportées à l'image, y compris CVEs les corrections de bogues mineurs, seront automatiquement appliquées à l'image jusqu'à la dernière version mineure correspondante.

Versions

Les clients utilisant la dernière version de la solution recevront automatiquement des correctifs de sécurité et des corrections de bogues mineurs et non révolutionnaires s'ils optent pour les mises à jour automatiques des images. L'image affichera automatiquement la dernière image vers la dernière version mineure correspondante. Afin de verrouiller le conteneur dans une version spécifique, la définition de la tâche peut être modifiée pour spécifier que le conteneur doit utiliser une version d'image spécifique en utilisant la version balisée de l'image. Les mises à jour automatiques peuvent également être désactivées en sélectionnant Non aux mises à jour automatiques CloudFormation lors du lancement de la pile. Cela lancera la version de l'image correspondant à la version de la solution.

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/ecslister.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"
    fi

```

```

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 `ecslister.py` script, tandis que la tâche principale exécute le `ecscontroller.py` script. Le `ecslister.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 sur `testScenario` les paramètres et autres, reportez-vous aux [scénarios](#) et à l'[exemple de charge utile](#) dans le GitHub référentiel.

Scénarios

- [GET /scénarios](#)
- [POST /scénarios](#)
- [OPTIONS/SCÉNARIOS](#)
- [OBTENEZ /scenarios/ {testId}](#)
- [POST /scenarios/ {testId}](#)
- [SUPPRIMER /scenarios/ {testId}](#)
- [OPTIONS /scénarios/ {testId}](#)

Tâches

- [GET /tâches](#)
- [OPTIONS /tâches](#)

Régions

- [GET /régions](#)
- [OPTIONS /régions](#)

GET /scenarios

Description

L'GET /scenariosopération permet de récupérer une liste de scénarios de test.

Réponse

Name (Nom)	Description
data	Une liste de scénarios comprenant l'ID, le nom, la description, le statut et la durée d'exécution de chaque test

POST /scénarios

Description

L'POST /scenariosopération permet de créer ou de planifier un scénario de test.

Corps de la demande

Name (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)

Name (Nom)	Description
fileType	Le type de fichier de téléchargement (par exemple,none,script,zip)
scheduleDate	Date à laquelle un test doit être effectué. 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)
scheduleStep	Étape du processus de planification. Fourni uniquement si vous planifiez un test récurrent . (Les étapes disponibles incluent create etstart)
cronvalue	La valeur cron pour personnaliser la planification récurrente. Le cas échéant, omettez ScheduleDate et ScheduleTime.
cronExpiryDate	Date requise pour que le cron expire et ne s'exécute pas indéfiniment.
recurrence	La récurrence d'un test programmé. Fourni uniquement si vous planifiez un test récurrent (par exemple dailyweekly,,biweekly, oumonthly)

Réponse

Name (Nom)	Description
testId	L'identifiant unique du test
testName	Le nom du test

Name (Nom)	Description
status	État du test

OPTIONS/SCÉNARIOS

Description

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

Réponse

Name (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.

Paramètre de demande

testId

- L'identifiant unique du test

Type : String

Obligatoire : oui

Réponse

Name (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	Le type de fichier chargé (par exemple none, 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
nextRun	La prochaine exécution planifiée (par exemple, 2017-04-22 17:18:00)

Name (Nom)	Description
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

Name (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

Name (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

Name (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 exemplesimple,jmeter)
fileType	Le type de fichier chargé (par exemplenone,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

Name (Nom)	Description
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 /tâches

Description

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

Réponse

Name (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

Name (Nom)	Description
taskIds	Une liste de tâches IDs pour exécuter des tests

GET /régions

Description

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

Réponse

Name (Nom)	Description
testId	L'identifiant de la 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 /regionsopération fournit une réponse à la demande avec les en-têtes de réponse CORS corrects.

Réponse

Name (Nom)	Description
testId	L'identifiant de la 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

Augmenter les ressources en conteneurs

Pour augmenter le nombre d'utilisateurs actuellement pris en charge, augmentez les ressources du conteneur. Cela vous permet d'augmenter la mémoire CPUs et de gérer l'augmentation du nombre d'utilisateurs simultanés.

Création d'une nouvelle révision de définition de tâche

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.
 - 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 en fonction de la mémoire souhaitée.
 - 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 le nom de la nouvelle définition de tâche. Ce nom inclut le numéro de version, par exemple : `[replaceable] <stackName>- EcsTaskDefinition - <system-generated-random-Hash> : [remplaçable]<system-generated-versionNumber>`.

Mettre à jour la table DynamoDB

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 modifié la définition de tâche. Par exemple, `region-<region-name>`.
5. Mettez à jour l'attribut `TaskDefinition` avec la nouvelle définition de tâche.

Référence

Cette section inclut des informations sur une fonctionnalité facultative permettant de collecter des métriques uniques pour cette solution, des pointeurs vers des ressources connexes et une liste des créateurs qui ont contribué à cette solution.

Collecte de données anonymisée

Cette solution inclut une option permettant d'envoyer des métriques opérationnelles anonymisées à AWS. Nous utilisons ces données pour mieux comprendre la façon dont les clients utilisent cette solution et les services et produits associés. Lorsqu'elles sont invoquées, les informations suivantes sont collectées et envoyées à AWS :

- ID de solution : identifiant de solution AWS
- ID unique (UUID) : identifiant unique généré aléatoirement pour chaque déploiement de solution
- Horodatage - Horodatage de la collecte de données
- Type de test : type de test exécuté
- Type de fichier : type de fichier chargé
- Nombre de tâches : nombre de tâches pour chaque test soumis via l'API de la solution
- Durée de la tâche : durée totale d'exécution de toutes les tâches nécessaires à l'exécution d'un test
- Résultat du test : résultat du test effectué

AWS est propriétaire des données collectées dans le cadre de cette enquête. La collecte de données est soumise à la politique de [confidentialité d'AWS](#). Pour désactiver cette fonctionnalité, suivez les étapes ci-dessous avant de lancer le CloudFormation modèle AWS.

1. Téléchargez le [CloudFormation modèle AWS](#) sur votre disque dur local.
2. Ouvrez le CloudFormation modèle AWS dans un éditeur de texte.
3. Modifiez la section de mappage des CloudFormation modèles AWS à partir de :

```
Solution:
  Config:
    SendAnonymousData: "Yes"
```

par :

```
Solution:  
Config:  
  SendAnonymousData: "No"
```

4. Connectez-vous à la [CloudFormation console AWS](#).
5. Sélectionnez Créer une pile.
6. Sur la page Créer une pile, section Spécifier le modèle, sélectionnez Télécharger un fichier modèle.
7. Sous Télécharger un fichier modèle, choisissez Choisir un fichier et sélectionnez le modèle modifié sur votre disque local.
8. Choisissez Next et suivez les étapes décrites dans [Lancer la pile](#) dans la section Déployer la solution de ce guide.

Collaborateurs

- Tom Nightingale
- Fernando Dingler
- Beomseok Lee
- George Lenz
- Erin McGill
- Dimitri López
- Kamyar Ziabari
- Bassem Wanis
- Garvit Singh
- Nikhail Reddy
- Simon Kroll

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](https://www.apache.org/licenses/LICENSE-2.0).

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.