AWS Well-Architected Framework

Pilier Fiabilité

Pilier Fiabilité: AWS Well-Architected Framework

Table of Contents

Pilier Fiabilité – AWS Well-Architected Framework.

Date de publication : 6 novembre 2024 (Révisions du document)

Ce document porte sur le pilier Fiabilité du cadre AWS Well-Architected Framework. Il fournit des conseils pour aider les clients à appliquer les bonnes pratiques de conception, de livraison et de maintenance des environnements Amazon Web Services (AWS).

Introduction

AWS Well-Architected Framework vous aide à mesurer le pour et le contre des options qui se présentent lors de la création de charges de travail sur AWS. En utilisant ce cadre, vous apprendrez les bonnes pratiques architecturales pour concevoir et exploiter des charges de travail fiables, sécurisés, efficaces, économiques et durables dans le cloud. Il permet d'évaluer régulièrement vos architectures par rapport aux bonnes pratiques et d'identifier les axes d'amélioration. Nous pensons qu'une charge de travail bien structurée augmente considérablement les chances de réussite des entreprises.

Le cadre AWS Well-Architected Framework repose sur six piliers :

- Excellence opérationnelle
- Sécurité
- Fiabilité
- Efficacité des performances
- Optimisation des coûts
- Durabilité

Ce livre blanc se concentre sur le pilier Fiabilité et sur la manière de l'appliquer à vos solutions. La fiabilité peut être difficile à atteindre dans les environnements sur site traditionnels, en raison de points de défaillance uniques, d'un manque d'automatisation et d'élasticité. L'adoption des pratiques détaillées dans ce document permettra de construire des architectures résilientes aux fondations solides, avec une gestion cohérente des modifications et des processus éprouvés de reprise en cas de défaillance.

Le présent document est conçu pour ceux et celles qui sont dépositaires de rôles technologiques, comme les directeurs de la technologie, les architectes, les développeurs et les membres de l'équipe

d’exploitation. Après avoir lu ce document, vous comprendrez les bonnes pratiques et les stratégies AWS à utiliser lors de la conception d’architectures cloud fiables. Ce livre blanc comprend des détails d’implémentation de haut niveau et des modèles d’architecture, ainsi que des références à des ressources supplémentaires.

Fiabilité

Le pilier Fiabilité englobe la capacité d'une charge de travail à exécuter sa fonction de manière correcte et cohérente et ce, en temps utile. Cela inclut la possibilité d'exploiter et de tester la charge de travail tout au long de son cycle de vie. Ce livre blanc fournit des bonnes pratiques détaillées pour la mise en œuvre de charges de travail fiables sur AWS.

Rubriques

Modèle de responsabilité partagée pour la résilience

La résilience est une responsabilité partagée entre AWS et vous. Il est important que vous compreniez comment la reprise après sinistre (DR) et la disponibilité, qui font partie de la résilience, fonctionnent dans le cadre de ce modèle partagé.

Responsabilité AWS : résilience du cloud

AWS est responsable de la résilience de l'infrastructure qui fait fonctionner tous les services proposés dans le AWS Cloud. Cette infrastructure est composée du matériel, des logiciels, du réseau et des installations exécutant les services AWS Cloud. AWS déploie des efforts commercialement raisonnables pour rendre ces services AWS Cloud disponibles, en veillant à ce que la disponibilité des services respecte ou dépasse les contrats de niveau de service (SLA) AWS.

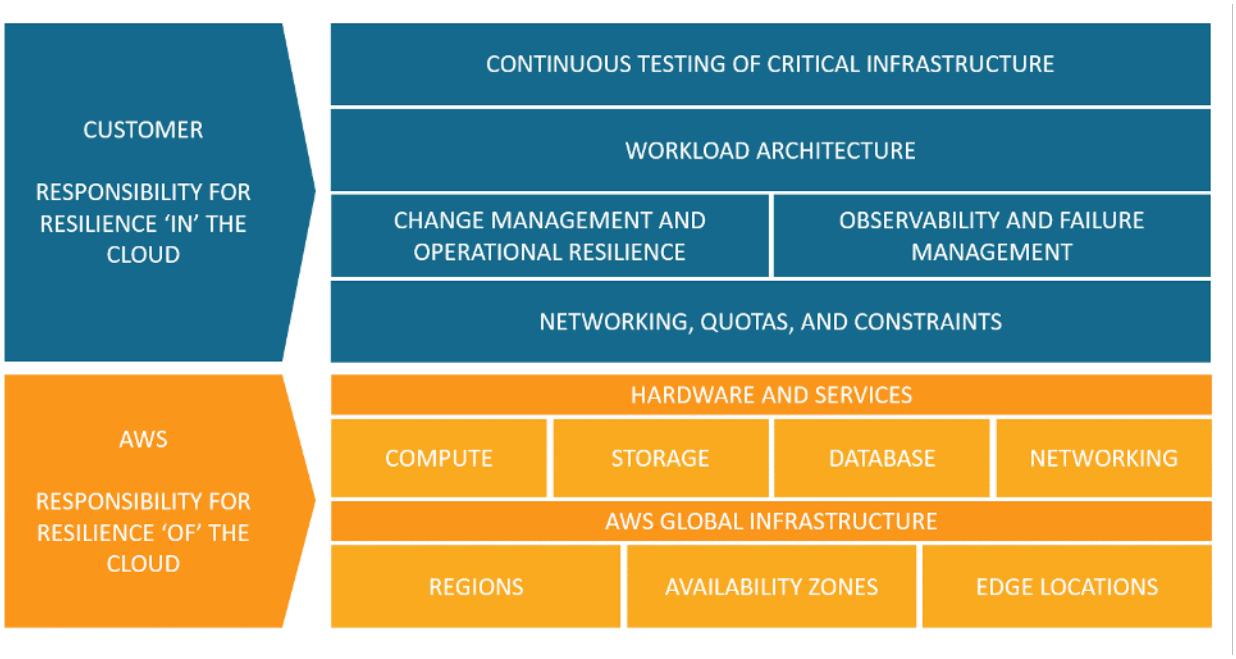
L'infrastructure cloud mondiale AWS est conçue pour permettre aux clients de créer des architectures de charge de travail hautement résilientes. Chaque Région AWS est entièrement isolée et comprend plusieurs zones de disponibilité, qui sont des partitions physiquement isolées de l'infrastructure. Les zones de disponibilité isolent les défaillances susceptibles d'affecter la résilience des charges de travail, en les empêchant d'avoir un impact sur les autres zones de la région. Toutes les zones de disponibilité d'une Région AWS sont interconnectées avec un réseau à large bande passante et à faible latence, qui utilise une fibre métropolitaine dédiée entièrement redondante fournissant un réseau à haut débit et à faible latence entre les AZ. Tout le trafic entre les zones est chiffré. Les performances du réseau sont suffisantes pour réaliser une réplication synchrone entre les zones.

Si une application est partitionnée sur plusieurs AZ, vous êtes mieux isolé et protégé contre les problèmes tels que les pannes de courant, la foudre, les tornades, les tremblements de terre, etc.

Responsabilité des clients : résilience dans le cloud

Votre responsabilité est déterminée par les services AWS Cloud que vous choisissez. Cela détermine la quantité de travail de configuration que vous devez effectuer dans le cadre de vos responsabilités en matière de résilience. Par exemple, un service tel que Amazon Elastic Compute Cloud (Amazon EC2) exige du client qu'il effectue toutes les tâches de configuration et de gestion de la résilience nécessaires. Les clients qui déploient des instances Amazon EC2 sont chargés de déployer des instances Amazon EC2 sur plusieurs sites (tels que les zones de disponibilité AWS), de mettre en œuvre l'autoréparation à l'aide de services tels que Auto Scaling et d'appliquer les bonnes pratiques en matière d'architecture de charge de travail résiliente pour les applications installées sur les instances. Dans le cas des services gérés, comme Amazon S3 et Amazon DynamoDB, AWS exploite la couche infrastructure, le système d'exploitation et les plateformes, et les clients accèdent aux points de terminaison pour stocker et récupérer les données. Vous êtes responsable de la gestion de la résilience de vos données, y compris des stratégies de sauvegarde, de gestion des versions et de réplication.

Le déploiement de votre charge de travail dans plusieurs zones de disponibilité d'une Région AWS fait partie d'une stratégie de haute disponibilité conçue pour protéger les charges de travail en isolant les problèmes dans une zone de disponibilité donnée. Cette stratégie utilise la redondance des autres zones de disponibilité pour continuer à répondre aux requêtes. Une architecture Multi-AZ s'inscrit également dans une stratégie DR conçue pour mieux isoler et protéger les charges de travail contre des problèmes tels que les pannes de courant, la foudre, les tornades, les tremblements de terre, etc. Les stratégies de DR peuvent également faire appel à de multiples Régions AWS. Par exemple, dans une configuration active/passive, le service de la charge de travail passe de sa région active à sa région DR si la région active ne peut plus répondre aux requêtes.



Responsabilité en matière de résilience dans et hors du cloud pour les clients et AWS.

Vous pouvez utiliser les services AWS pour atteindre vos objectifs de résilience. En tant que client, vous êtes responsable de la gestion des aspects suivants de votre système pour atteindre la résilience dans le cloud. Pour plus de détails sur chaque service en particulier, consultez la documentation AWS.

Réseaux, quotas et contraintes

- Les bonnes pratiques pour ce domaine du modèle de responsabilité partagée sont décrites en détail dans la section Fondations.
- Planifiez votre architecture avec une marge de manœuvre suffisante et comprenez les quotas de service et les contraintes des services que vous incluez, en fonction des augmentations de charge attendues, le cas échéant.
- Concevez la topologie de votre réseau pour qu'elle soit hautement disponible, redondante et évolutive.

Gestion des modifications et résilience opérationnelle

- La gestion des modifications inclut la manière d'introduire et de gérer les modifications dans votre environnement. La mise en œuvre du changement nécessite de créer et de maintenir à jour des runbooks ainsi que des stratégies de déploiement pour votre application et votre infrastructure.

- Une stratégie résiliente de surveillance des ressources de charge de travail prend en compte tous les composants, y compris les indicateurs techniques et commerciaux, les notifications, l'automatisation et l'analyse.
- Les charges de travail dans le cloud doivent s'adapter à l'évolution de la demande en réaction à des déficiences ou à des fluctuations d'utilisation.

Observabilité et gestion des pannes

- L'observation des défaillances par le biais de la surveillance est nécessaire pour automatiser la réparation afin que vos charges de travail puissent résister aux défaillances des composants.
- La gestion des défaillances nécessite de sauvegarder les données, d'appliquer les bonnes pratiques pour permettre à votre charge de travail de résister aux défaillances des composants et de planifier la reprise après sinistre.

Architecture de charge de travail

- L'architecture de votre charge de travail inclut la manière dont vous concevez des services autour de domaines commerciaux, appliquez la SOA et concevez des systèmes distribués pour éviter les défaillances, et intégrez des fonctionnalités telles que la limitation, les nouvelles tentatives, la gestion des files d'attente, les délais d'attente et les leviers d'urgence.
- Appuyez-vous sur des solutions AWS éprouvées, Amazon Builders' Library et des modèles sans serveur pour vous aligner sur les bonnes pratiques et accélérer les mises en œuvre.
- Utilisez l'amélioration continue pour décomposer votre système en services distribués afin d'évoluer et d'innover plus rapidement. Utilisez les conseils relatifs aux microservices AWS et les options de services gérés pour simplifier et accélérer votre capacité à introduire le changement et à innover.

Test continu des infrastructures critiques

- Pour tester la fiabilité, il faut effectuer des tests au niveau des fonctionnalités, des performances et du chaos, ainsi qu'adopter l'analyse des incidents et les pratiques des jours de jeu afin de développer une expertise en matière de résolution de problèmes mal compris.
- Pour les applications tout cloud et hybrides, le fait de savoir comment votre application se comporte lorsque des problèmes surviennent ou que des composants tombent en panne permet une reprise rapide et fiable après une panne.

- Créez et documentez des expériences reproductibles pour comprendre comment votre système se comporte lorsque les objets ne fonctionnent pas comme prévu. Ces tests prouveront l'efficacité de votre résilience globale et fourniront une boucle de rétroaction pour vos procédures opérationnelles avant de vous confronter à des scénarios de panne réels.

Principes de conception

Dans le cloud, il existe un certain nombre de principes qui peuvent vous aider à renforcer la fiabilité. Gardez les éléments suivants à l'esprit lorsque nous aborderons les meilleures pratiques :

- Récupération automatique après une panne : en contrôlant les indicateurs clés de performance d'une charge de travail, vous pouvez exécuter l'automatisation en cas de transgression d'un seuil. Ces KPI doivent couvrir la valeur commerciale et non des aspects techniques du fonctionnement du service. Cela permet la création de notifications automatiques, le suivi des pannes et l'exécution de processus de récupération automatique qui contournent ou corrigent les pannes. Une automatisation plus sophistiquée rend possible l'anticipation et la correction des pannes avant qu'elles ne se produisent.
- Test des procédures de récupération : dans un environnement sur site, des tests sont souvent conduits pour prouver que la charge de travail fonctionne dans un scénario particulier. Ces tests ne sont généralement pas utilisés pour valider les stratégies de récupération. Dans le cloud, vous pouvez tester de quelle façon votre charge de travail cesse de fonctionner et valider vos procédures de récupération. Vous pouvez utiliser l'automatisation pour simuler différentes pannes ou recréer les scénarios qui ont déjà conduit à des pannes. Cette approche expose les chemins de défaillance que vous pouvez tester et corriger avant qu'un scénario de défaillance réelle ne se produise et réduire ainsi les risques.
- Mise à l'échelle horizontale pour augmenter la disponibilité de la charge de travail : remplacez une ressource volumineuse par plusieurs petites ressources pour réduire l'impact d'une défaillance unique sur la charge de travail globale. Répartissez les demandes entre plusieurs ressources plus petites pour garantir qu'elles ne partagent pas un point de panne commun.
- Une capacité réellement adaptée à vos besoins : une cause courante de défaillance des charges de travail sur site est la saturation des ressources, lorsque les demandes ciblant une charge de travail en dépassent la capacité (c'est souvent l'objectif des attaques par déni de service). Dans le cloud, vous pouvez contrôler la demande et l'utilisation de la charge de travail. Vous pouvez aussi automatiser l'ajout ou la suppression de ressources afin de maintenir le niveau optimal de satisfaction de la demande sans surallocation ou sous-allocation. Il existe toujours des limites, mais

certaines peuvent être contrôlées et d'autres gérées (Consulter Gestion des quotas de service et des contraintes).

- Gestion des changements avec l'automatisation : les modifications apportées à l'infrastructure doivent être appliquées via l'automatisation. Les modifications qui doivent être gérées incluent celles apportées à l'automatisation et qui peuvent ensuite être suivies et vérifiées.

Définitions

Ce livre blanc aborde la fiabilité dans le cloud et décrit les bonnes pratiques pour ces quatre domaines :

- Fondations
- Architecture de charge de travail
- Gestion des modifications
- Gestion des défaillances

Pour atteindre la fiabilité, vous devez commencer par les fondations : un environnement où les quotas de service et la topologie réseau s'adaptent à la charge de travail. L'architecture de la charge de travail du système distribué doit être conçue pour prévenir et atténuer les défaillances. La charge de travail doit gérer les modifications au niveau de la demande ou des exigences. Elle doit être conçue pour détecter les défaillances et se réparer automatiquement.

Rubriques

- Résilience et composants de la fiabilité
- Disponibilité
- Objectifs de reprise après sinistre

Résilience et composants de la fiabilité

La fiabilité d'une charge de travail dans le cloud dépend de plusieurs facteurs, dont le principal est la résilience :

- La résilience est la capacité d'une charge de travail à récupérer après des perturbations de l'infrastructure ou du service, d'acquérir de manière dynamique les ressources de calcul pour

satisfaire la demande et d'atténuer les perturbations telles que les erreurs de configuration ou les problèmes réseau temporaires.

Les autres facteurs qui affectent la fiabilité de la charge de travail sont les suivants :

- L'excellence opérationnelle qui comprend l'automatisation des modifications, l'utilisation de manuels pour corriger les pannes et les examens de disponibilité opérationnelle pour vérifier que les applications sont prêtes pour les opérations de production.
- La sécurité qui englobe la prévention des dommages causés aux données ou à l'infrastructure par des acteurs malveillants pour prévenir tout impact sur la disponibilité. Par exemple, chiffrez les sauvegardes pour vous sécuriser les données.
- L'efficacité des performances qui intègre la conception pour maximiser les taux de requêtes et réduire au maximum les latences de votre charge de travail.
- L'optimisation des coûts, ce qui implique des compromis, tels que le choix entre une hausse des dépenses sur les instances EC2 pour atteindre la stabilité statique ou le recours à la mise à l'échelle automatique pour augmenter la capacité en fonction des besoins.

La résilience est l'objectif principal de ce livre blanc.

Les quatre autres aspects sont également importants et sont couverts par leurs piliers respectifs du Cadre AWS Well-Architected. Un grand nombre des bonnes pratiques décrites ici traitent également des aspects liés à la fiabilité, mais l'accent est mis sur la résilience.

Disponibilité

La disponibilité (ou disponibilité du service) est à la fois une métrique couramment utilisée pour mesurer quantitativement la résilience, ainsi qu'un objectif de résilience cible.

- La disponibilité correspond au pourcentage de temps pendant lequel une charge de travail est disponible à' l'utilisation.

Disponible à l'utilisation signifie que la charge de travail exécute bien sa fonction chaque fois que nécessaire.

Ce pourcentage se calcule sur une période, par exemple un mois, un an ou les trois dernières années. Dans son interprétation la plus stricte, la disponibilité est réduite chaque fois que l'application

ne fonctionne pas normalement, y compris pendant les interruptions planifiées et non planifiées. Nous définissons la disponibilité comme suit :

$$Availability = \frac{Available\ for\ Use\ Time}{Total\ Time}$$

- La disponibilité est un pourcentage de temps de fonctionnement (par exemple, 99,9 %) sur une période (généralement un mois ou un an)
- Un raccourci courant consiste à parler du « nombre de neuf ». Par exemple, « cinq 9 » correspond à une disponibilité de 99,999 %.
- Certains clients choisissent d'exclure les temps d'arrêt programmés (par exemple, la maintenance planifiée) de la durée totale dans la formule. Cependant, cela n'est pas conseillé, car vos utilisateurs voudront probablement utiliser votre service pendant ces périodes.

Voici un tableau des objectifs courants de disponibilité des applications et de la durée maximale des interruptions qui peuvent se produire sur une année, tout en continuant d'atteindre l'objectif. Le tableau contient des exemples des types d'applications généralement rencontrées à chaque niveau de disponibilité. Tout au long de ce document, nous ferons référence à ces valeurs.

Disponibilité	Indisponibilité maximale (par an)	Catégories d'applications
99 %	3 jours 15 heures	Tâches de traitement par lots, d'extraction de données, de transfert et de chargement
99,9 %	8 heures 45 minutes	Outils internes, tels que la gestion des connaissances, le suivi des projets
99,95 %	4 heures 22 minutes	Commerce en ligne, point de vente
99,99 %	52 minutes	Diffusion vidéo, charges de travail de diffusion

Disponibilité	Indisponibilité maximale (par an)	Catégories d'applications
99,999 %	5 minutes	Transactions de distributeurs, charges de travail de télécommunications

Mesurez la disponibilité en fonction des demandes. Pour votre service, il peut être plus facile de compter les demandes ayant réussi ou échoué au lieu du « temps disponible pour utilisation ». Dans ce cas, le calcul suivant peut être utilisé :

$$Availability = \frac{Successful\ Responses}{Valid\ Requests}$$

Cette mesure porte souvent sur des périodes d'une minute ou de cinq minutes. Un pourcentage de disponibilité mensuelle (mesure de la disponibilité sur la base du temps) peut être calculé à partir de la moyenne de ces périodes. Si aucune demande n'est reçue au cours d'une période donnée, elle est comptée comme étant disponible à 100 % pour cette période.

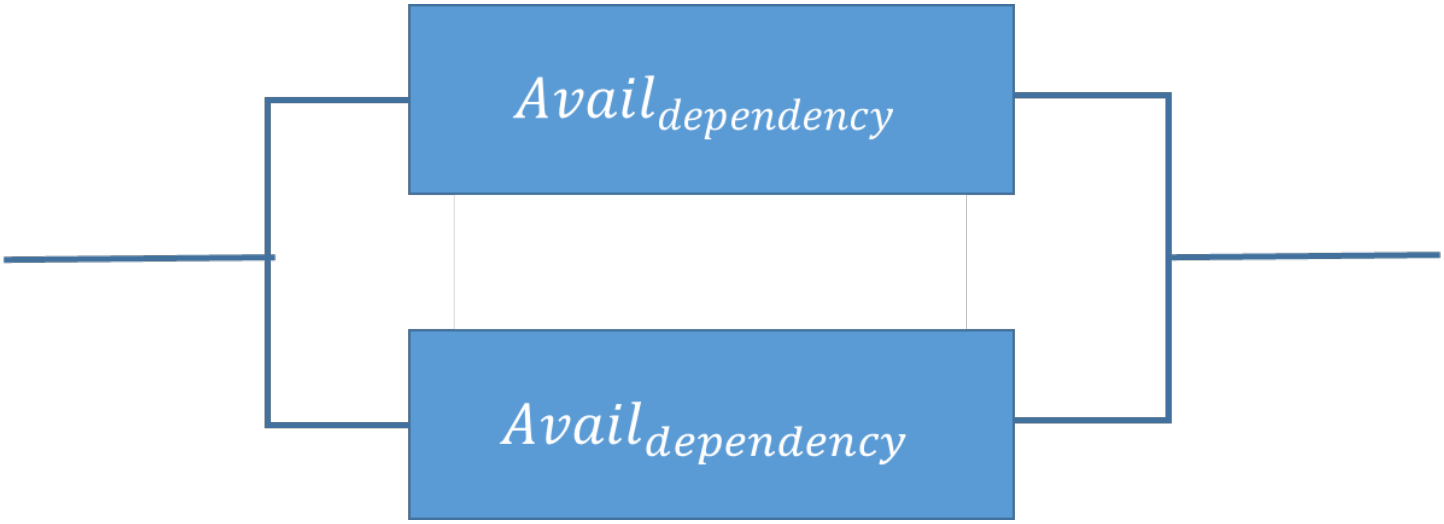
Calcul de disponibilité avec des dépendances strictes. De nombreux systèmes ont des dépendances strictes avec d'autres systèmes. Dans ce cas, une interruption dans un système dépendant se traduit directement par une interruption du système appelant. Cette notion s'oppose à celle de dépendance souple, où une défaillance du système dépendant est compensée dans l'application. Quand de telles dépendances strictes se produisent, la disponibilité du système appelant est le produit de la disponibilité des systèmes dépendants. Par exemple, si un système conçu pour offrir une disponibilité de 99,99 % a une dépendance stricte avec deux autres systèmes indépendants, qui sont chacun conçus pour offrir une disponibilité de 99,99 %, la charge de travail peut théoriquement atteindre 99,97 % de disponibilité :

$Avail_{invok} \times Avail_{dep1} \times Avail_{dep2} = Avail_{workload}$

99,99 % × 99,99 % × 99,99 % = 99,97 %

Il est donc important de comprendre vos dépendances et leurs objectifs de conception de disponibilité dans votre propre calcul de disponibilité.

Calcul de disponibilité avec des composants redondants. Lorsqu'un système implique l'utilisation de composants redondants et indépendants (par exemple, des ressources redondantes dans des zones de disponibilité redondantes), la disponibilité théorique est calculée à hauteur de 100 %, moins le produit des taux de défaillance des composants. Par exemple, si un système utilise deux composants indépendants, chacun avec une disponibilité de 99,9 %, la disponibilité effective de cette dépendance est > 99,9999 % :



$Avail_{effective} = Avail_{MAX} - ((100\% - Avail_{dependency}) \times (100\% - Avail_{dependency}))$

99,9999 % = 100 % − (0,1 % × 0,1 %)

Calcul de raccourci : si les disponibilités de tous les composants de votre calcul contiennent uniquement le chiffre neuf, vous pouvez additionner le nombre de neuf pour obtenir votre réponse. Dans l'exemple ci-dessus, deux composants redondants et indépendants avec trois neuf disponibles donnent six neuf.

Calcul de la disponibilité d'une dépendance. Certaines dépendances fournissent des informations sur leur disponibilité, y compris les objectifs de conception de disponibilité pour de nombreux services AWS Si cette information n'est pas disponible (par exemple, pour un composant où le fabricant ne publie pas les données de disponibilité), un moyen d'estimation consiste à déterminer le temps moyen entre deux pannes (MTBF) et le temps moyen de récupération (MTTR). Une estimation de disponibilité peut être établie par la formule suivante :

$$Avail_{EST} = \frac{MTBF}{MTBF + MTTR}$$

Par exemple, si le MTBF est de 150 jours et que le MTTR est de 1 heure, l'estimation de disponibilité est de 99,97 %.

Pour plus de détails, consultez Availability and Beyond : Understanding and improving the resilience of distributed systems on AWS, qui peut vous aider à calculer votre disponibilité.

Coûts liés à la disponibilité. La conception d'applications pour de plus hauts niveaux de disponibilité est généralement synonyme de coûts accrus. Par conséquent, il est nécessaire d'identifier les vrais besoins de disponibilité avant de démarrer la conception de votre application. Un haut niveau de disponibilité impose des exigences plus strictes pour les tests et la validation dans des scénarios de défaillance exhaustifs. L'automatisation est nécessaire pour la récupération à partir de toutes sortes de défaillances, et tous les aspects des opérations système doivent être conçus et testés selon les mêmes normes. Par exemple, l'ajout ou la suppression de capacité, le déploiement ou la restauration des mises à jour logicielles ou des modifications de configuration ou la migration des données système doivent être réalisées en fonction de l'objectif de disponibilité souhaité. Outre le coût de développement du logiciel, les très hauts niveaux de disponibilité sont un frein à l'innovation, car ils nécessitent de déployer beaucoup plus lentement les systèmes. Il est donc recommandé d'être minutieux dans l'application des normes et dans la sélection d'un objectif de disponibilité adapté pour tout le cycle de vie de l'exploitation du système.

La sélection de dépendances est un autre facteur d'augmentation des coûts pour les systèmes opérant avec des objectifs de conception de disponibilité élevés. Avec ces objectifs plus élevés, l'ensemble de logiciels ou services qui peuvent être choisis en tant que dépendances diminue en fonction des services qui ont bénéficié des investissements importants décrits précédemment. Quand l'objectif de conception de disponibilité augmente, les services multifonctions (par exemple, les bases de données relationnelles) sont moins nombreux par rapport aux services plus spécialisés. Ces derniers sont en effet plus faciles à évaluer, tester et automatiser, et ils sont moins susceptibles de présenter des interactions imprévues avec des fonctionnalités incluses, mais non utilisées.

Objectifs de reprise après sinistre

Outre les objectifs de disponibilité, votre stratégie de résilience doit également inclure des objectifs de reprise après sinistre (DR) basés sur des stratégies de récupération de votre charge de travail en cas de sinistre. La reprise après sinistre se concentre sur des objectifs de reprise ponctuels en réponse à des catastrophes naturelles, des pannes techniques à grande échelle ou des menaces humaines telles qu'une attaque ou une erreur. Elle diffère de la disponibilité qui, elle, mesure la résilience sur une période en réponse aux pannes de composants, aux pics de charge ou aux bogues logiciels.

L'objectif de délai de reprise (RTO) est défini par l'organisation. La RTO correspond au délai maximum acceptable entre l'interruption du service et la restauration du service. Elle détermine ce qui est considéré comme étant un créneau de temps acceptable d'indisponibilité du service.

L'objectif de délai de reprise (RTO) est défini par l'organisation. Le RPO correspond au temps maximal acceptable depuis le dernier point de reprise des données. Il détermine ce qui est considéré comme étant une perte de données acceptable entre le dernier point de reprise et l'interruption du service.

Relation entre le RPO (objectif de point de reprise), la RTO (durée maximale d'interruption admissible) et l'événement de sinistre.

La RTO est similaire au MTTR (temps moyen de reprise) en ce que ces deux valeurs mesurent le délai entre le début d'une panne et la reprise de la charge de travail. Cependant, le MTTR est une valeur moyenne obtenue à partir de plusieurs événements ayant un impact sur la disponibilité au cours d'une période donnée, alors que la RTO est une cible, ou une valeur maximale autorisée, pour un seul événement ayant un impact sur la disponibilité.

Compréhension des besoins en disponibilité

Il est fréquent de considérer initialement la disponibilité d'une application comme un objectif unique à atteindre pour l'application dans son ensemble. Toutefois, en y regardant de plus près, on constate souvent que certains aspects d'une application ou d'un service présentent différentes exigences en

matière de disponibilité. Par exemple, certains systèmes peuvent prioriser la possibilité de recevoir et de stocker de nouvelles données, au détriment de la récupération de données existantes. D'autres systèmes vont privilégier les opérations en temps réel sur les opérations qui modifient la configuration ou l'environnement d'un système. Les services peuvent avoir des exigences de disponibilité très élevées à certains moments de la journée, mais tolérer des périodes de perturbation beaucoup plus longues le reste du temps. Voici quelques-unes des manières dont vous pouvez diviser une même application en différents éléments constitutifs, afin d'évaluer les exigences de disponibilité pour chaque partie. Cette façon de procéder a pour avantage de permettre de concentrer vos efforts (et dépenses) sur la disponibilité en fonction de besoins spécifiques, plutôt que de concevoir l'ensemble du système sur la base de l'exigence la plus stricte.

Recommendation
Évaluez de manière critique les aspects uniques de vos applications et, le cas échéant, différenc iez les objectifs de conception de la disponibilité et de la reprise après sinistre pour refléter les besoins de votre entreprise.

Chez AWS, nous divisons souvent les services en deux, avec un « plan de données » et un « plan de contrôle ». Le plan de données vise à fournir un service en temps réel, tandis que les plans de contrôle servent à configurer l'environnement. Par exemple, les instances Amazon EC2, les bases de données Amazon RDS et les opérations de lecture/écriture sur les tables Amazon DynamoDB sont autant d'opérations qui relèvent du plan de données. En revanche, le lancement de nouvelles instances EC2 ou bases de données RDS, ou l'ajout ou la modification des métadonnées de table dans DynamoDB sont considérés comme des opérations de plan de contrôle. Tandis que de hauts niveaux de disponibilité sont importants pour l'ensemble de ces fonctionnalités, les plans de données ont généralement des objectifs de conception de disponibilité plus élevés que les plans de contrôle. Par conséquent, les charges de travail avec des exigences de haute disponibilité doivent éviter la dépendance d'exécution vis-à-vis des opérations du plan de contrôle.

La plupart des clients AWS adoptent une approche similaire pour évaluer leurs applications avec un esprit critique et identifier les sous-composants avec différents besoins de disponibilité. Les objectifs de conception de disponibilité sont ensuite adaptés aux différents aspects et les efforts de travail appropriés sont mis en œuvre pour concevoir le système. AWS a accumulé une expérience importante dans l'ingénierie d'applications avec un éventail d'objectifs de conception de disponibilité, y compris des services avec une disponibilité de 99,999 % ou plus. AWS Les architectes de solution peuvent vous aider à concevoir de manière appropriée en fonction de vos objectifs de disponibilité.

Impliquer AWS de manière précoce dans votre processus de conception nous permet de vous aider à atteindre vos objectifs de disponibilité. La planification pour la disponibilité ne se fait pas uniquement avant le lancement de votre charge de travail. Elle s'effectue également en continu de manière à affiner votre conception à mesure que vous accumulez de l'expérience opérationnelle, tirez des enseignements des événements réels et êtes confronté à différents types de défaillances. Vous pouvez ensuite appliquer l'effort de travail approprié pour améliorer votre mise en œuvre.

Les besoins en disponibilité requis pour une charge de travail doivent être alignés sur les besoins et la criticité de l'entreprise. En commençant par définir un cadre de criticité commerciale avec une RTO, un RPO et une disponibilité spécifiques, vous pouvez évaluer chaque charge de travail. Une telle approche exige que les personnes impliquées dans la mise en œuvre de la charge de travail connaissent le cadre et l'impact de leur charge de travail sur les besoins de l'entreprise.

Fondations

Les exigences de base sont celles dont le champ d'application s'étend au-delà d'une seule charge de travail ou d'un seul projet. Avant de concevoir l'architecture d'un système, les exigences de base qui influent sur la fiabilité doivent être mises en place. Par exemple, vous devez avoir une bande passante du réseau suffisante pour votre centre de données.

Dans un environnement sur site, ces exigences peuvent entraîner de longs délais d'attente en raison des dépendances et, par conséquent, doivent être intégrées lors de la planification initiale. Toutefois, avec AWS, la plupart de ces exigences en matière de fondation sont déjà intégrées ou peuvent être satisfaites en fonction des besoins. Le cloud est conçu pour être presque illimité. Il est donc de la responsabilité d'AWS de satisfaire l'exigence d'une capacité suffisante de mise en réseau et de calcul, ce qui vous laisse la liberté de modifier la taille des ressources et les allocations à la demande.

Les sections suivantes présentent les bonnes pratiques qui se concentrent sur ces considérations relatives à la fiabilité.

Rubriques

Gestion des quotas et contraintes de service

Pour les architectures de charge de travail basées sur le Cloud, il existe des quotas de service (également appelés limites de service). Ces quotas permettent d'éviter de fournir accidentellement plus de ressources que nécessaire et de limiter les taux de demande des opérations d'API afin de protéger les services de tout abus. Il existe également des contraintes de ressources, par exemple la vitesse à laquelle les bits peuvent être transmis par un câble à fibre optique ou la quantité de données stockées sur un disque physique.

Si vous utilisez des applications AWS Marketplace, vous devez en comprendre les limitations. De la même manière, si vous utilisez des services Web tiers ou des solutions logicielles en tant que service, vous devez être conscient de leurs limites.

Bonnes pratiques

- REL01-BP03 Prise en compte des quotas de services et des contraintes de service fixes dans l'architecture
- REL01-BP04 Surveiller et gérer les quotas
- REL01-BP05 Automatisation de la gestion des quotas
- REL01-BP06 Garantir un écart suffisant entre les quotas actuels et l'utilisation maximale pour permettre le basculement

REL01-BP01 Connaissance des quotas de service et des contraintes

Connaissez vos quotas par défaut et gérez vos demandes d'augmentation de quota pour votre architecture de charge de travail. Connaissez également les contraintes de ressources, comme le disque ou le réseau, qui sont susceptibles d'avoir un impact.

Résultat escompté : les clients peuvent empêcher la dégradation ou l'interruption de service de leurs Comptes AWS en mettant en œuvre des directives appropriées pour le suivi des métriques clés, des examens de l'infrastructure et des mesures correctives automatisées afin de vérifier que les quotas et les contraintes des services ne sont pas atteints, ce qui pourrait entraîner une dégradation ou une interruption du service.

Anti-modèles courants :

- Déployer une charge de travail sans comprendre les quotas matériels ou logiciels et leurs limites pour les services utilisés.
- Déployer une charge de travail de remplacement sans analyser ni reconfigurer les quotas nécessaires ou contacter d'abord l'assistance.
- Supposer que les services cloud sont sans limite et que les services peuvent être utilisés sans prendre en compte les taux, les limites, les nombres et les quantités.
- Supposer que les quotas augmenteront automatiquement.
- Ne pas connaître le processus et la chronologie des demandes de quotas.
- Supposer que le quota du service cloud pas défaut est le même pour chaque service par rapport à d'autres régions.
- Supposer que les contraintes de service peuvent être enfreintes et que les systèmes se mettront automatiquement à l'échelle ou augmenteront la limite au-delà des contraintes de la ressource.
- Ne pas tester l'application sur des pics de trafic pour tester la résistance de l'utilisation de ces ressources.

- Provisionner les ressources sans analyser la taille de ressource nécessaire.
- Surprovisionner la capacité en choisissant des types de ressources qui vont bien au-delà des besoins réels ou des pics attendus.
- Ne pas évaluer les exigences de capacité pour les nouveaux niveaux de trafic avant un nouvel événement client ou le déploiement d'une nouvelle technologie.

Avantages de l'établissement de cette bonne pratique : la surveillance et la gestion automatisée des quotas de service et des contraintes de ressources peuvent réduire les défaillances de manière proactive. Les changements dans les modèles de trafic d'un service client peuvent entraîner une interruption ou une dégradation si les bonnes pratiques ne sont pas suivies. En surveillant et en gérant les valeurs de quota sur toutes les régions et tous les comptes, les applications peuvent bénéficier d'une meilleure résilience lors d'événements indésirables ou imprévus.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : élevé

Directives d'implémentation

Service Quotas est un service AWS qui vous aide à gérer vos quotas pour plus de 250 services AWS à partir d'un seul emplacement. En plus de rechercher les valeurs de quota, vous pouvez également demander et suivre les augmentations de quota à partir de la console Service Quotas ou via le kit SDK AWS. AWS Trusted Advisor propose un contrôle des quotas de service qui affiche votre utilisation et les quotas de différents aspects de certains services. Les quotas de service par défaut pour chaque service figurent également dans la documentation AWS pour chaque service (par exemple, consultez Amazon VPC Quotas).

Certaines limites de services, comme les limites de taux sur les API limitées sont définies dans Amazon API Gateway en configurant un plan d'utilisation. Parmi les limites qui sont définies en tant que configuration sur leurs services respectifs figurent les IOPS provisionnées, le stockage Amazon RDS alloué et les allocations de volume Amazon EBS. Amazon Elastic Compute Cloud dispose de son propre tableau de bord des limites de service qui peut vous aider à gérer vos limites d'instances, d'Amazon Elastic Block Store et d'adresses IP élastiques. Si vous possédez un cas d'utilisation pour lequel les quotas de service affectent les performances de votre application sans être ajustables à vos besoins, contactez Support pour déterminer si des mesures d'atténuation peuvent être implémentées.

Les quotas de service peuvent être spécifiques à une région ou mondiaux par nature. Un service AWS qui atteint son quota ne se comportera pas comme lors d'une utilisation normale et peut entraîner une interruption ou une dégradation du service. Par exemple, un quota de service limite

le nombre d'instances DL Amazon EC2 utilisées dans une région. Cette limite peut être atteinte lors d'un événement de dimensionnement du trafic à l'aide des groupes Auto Scaling (ASG).

L'utilisation des quotas de service pour chaque compte doit être évaluée régulièrement pour déterminer quelles seraient les limites de service appropriées pour ce compte. Ces quotas de service existent en tant que barrières de protection opérationnelles pour empêcher le provisionnement accidentel de plus de ressources que nécessaire. Ils servent également à limiter les taux de requêtes sur les opérations d'API pour protéger les services des abus.

Les contraintes de service sont différentes des quotas de service. Les contraintes de service représentent les limites d'une ressource spécifique, telles que définies par ce type de ressource. Il peut s'agir de la capacité de stockage (par exemple, gp2 a une limite de 1 Go à 16 To) ou du débit du disque. Il est essentiel qu'une contrainte d'un type de ressource soit optimisée et constamment évaluée par rapport à une utilisation qui pourrait atteindre ses limites. Si une contrainte est atteinte de manière inattendue, les applications ou les services du compte peuvent être dégradés ou interrompus.

S'il existe un cas d'utilisation pour lequel les quotas de service affectent les performances d'une application sans être ajustables à vos besoins, contactez Support pour déterminer si des améliorations sont possibles. Pour plus de détails sur l'ajustement des quotas fixes, consultez REL01-BP03 Prise en compte des quotas de services et des contraintes de service fixes dans l'architecture.

Il existe un grand nombre de services et d'outils AWS pour vous aider à surveiller et gérer Service Quotas. Les services et les outils doivent être exploités pour vérifier automatiquement ou manuellement les niveaux de quotas.

- AWS Trusted Advisor propose un contrôle des quotas de service qui affiche votre utilisation et les quotas de différents aspects de certains services. Il peut aider à identifier des services proches du quota.
- AWS Management Console fournit des méthodes permettant d'afficher les valeurs des quotas de service, de les gérer, de demander de nouveaux quotas, de surveiller le statut des demandes de quotas et d'afficher l'historique des quotas.
- La AWS CLI et les CDK offrent des méthodes par programmation pour gérer et surveiller automatiquement les niveaux et l'utilisation des quotas de service.

Étapes d'implémentation

Pour Service Quotas :

- Passez en revue AWS Service Quotas.
- Pour connaître vos quotas de service existant, déterminez les services (comme IAM Access Analyzer) utilisés. Il existe environ 250 services AWS contrôlés par des quotas de service. Ensuite, déterminez le nom du quota de service spécifique qui pourrait être utilisé au sein de chaque compte et région. Il existe environ 3 000 noms de quotas de service par région.
- Complétez cette analyse des quotas avec AWS Config pour trouver toutes les ressources AWS utilisées dans vos Comptes AWS.
- Utilisez les données AWS CloudFormation pour déterminer les ressources AWS que vous utilisez. Examinez les ressources qui ont été créées dans la AWS Management Console ou avec la commande `list-stack-resources`AWS CLI. Vous pouvez également voir les ressources configurées pour être déployées directement dans le modèle.
- Déterminez tous les services indispensables à votre charge de travail en prenant en compte le code de déploiement.
- Identifiez les quotas de service pertinents. Utilisez les informations accessibles par programme depuis Trusted Advisor et Service Quotas.
- Établissez une méthode de surveillance automatisée (consultez REL01-BP02 Gestion des quotas de services entre les comptes et les régions et REL01-BP04 Surveiller et gérer les quotas) pour alerter et informer si les quotas de services sont proches de leur limite ou ont atteint leur limite.
- Établissez une méthode automatisée et programmatique pour vérifier si un quota de service a été modifié dans une région mais pas dans d'autres régions du même compte (consultez REL01-BP02 Gestion des quotas de services entre les comptes et les régions etREL01-BP04 Surveiller et gérer les quotas).
- Automatisez l'analyse des journaux et des métriques de l'application pour déterminer s'il existe des erreurs de contraintes de quotas ou de services. Si de telles erreurs existent, envoyez des alertes au système de surveillance.
- Établissez des procédures d'ingénierie pour calculer le changement de quota requis (consultez REL01-BP05 Automatisation de la gestion des quotas) une fois qu'il a été déterminé que des quotas plus importants sont nécessaires pour des services spécifiques.
- Créez un flux de travail de provisionnement et d'approbation pour demander des modifications des quotas de service. Cela doit inclure un flux de travail d'exception en cas de refus d'une demande ou d'une approbation partielle.

- Créez une méthode d'ingénierie pour vérifier les quotas de service avant le provisionnement et utilisez de nouveaux services AWS avant le déploiement dans des environnements de production ou chargés (par exemple, un compte de test de charge).

Pour les contraintes de service :

- Établissez des méthodes de surveillance et des métriques pour alerter quand les ressources sont proches de leurs contraintes. Tirez profit de CloudWatch si nécessaire pour la surveillance des métriques ou des journaux.
- Établissez des seuils d'alertes pour chaque ressource ayant une contrainte importante pour l'application ou le système.
- Créez des procédures de gestion des flux de travail et de l'infrastructure pour changer le type de ressource si la contrainte est proche de l'utilisation. Ce flux de travail doit inclure le test de charge comme une bonne pratique pour vérifier que ce nouveau type est le bon type de ressource avec les nouvelles contraintes.
- Procédez à la migration des ressources identifiées vers le nouveau type de ressource recommandé avec les procédures et les processus existants.

Ressources

Bonnes pratiques associées:

- REL01-BP02 Gestion des quotas de services entre les comptes et les régions
- REL01-BP03 Prise en compte des quotas de services et des contraintes de service fixes dans l'architecture
- REL01-BP04 Surveiller et gérer les quotas
- REL01-BP05 Automatisation de la gestion des quotas
- REL01-BP06 Garantir un écart suffisant entre les quotas actuels et l'utilisation maximale pour permettre le basculement
- REL03-BP01 Choisissez comment segmenter votre charge de travail
- REL10-BP01 Déploiement de la charge de travail sur plusieurs emplacements
- REL11-BP01 Surveiller tous les composants de la charge de travail pour détecter les défaillances
- REL11-BP03 Automatiser la réparation sur toutes les couches
- REL12-BP04 Tester la résilience à l'aide de l'ingénierie du chaos

Documents connexes :

- Pilier Fiabilité du cadre AWS Well-Architected : disponibilité
- AWS Service Quotas (anciennement Service Limits)
- Vérifications des bonnes pratiques AWS Trusted Advisor (voir la section Service Limits)
- AWS Limit Monitor sur AWS Answers
- Amazon EC2 Service Limits
- Qu'est-ce que Service Quotas ?
- Comment demander une augmentation de quota
- Points de terminaison et quotas de service
- Guide de l'utilisateur de Service Quotas
- Quota Monitor pour AWS
- AWS Fault Isolation Boundaries
- Disponibilité avec redondance
- AWS pour les données
- Qu'est-ce que l'intégration continue ?
- Qu'est-ce que la livraison continue ?
- Partenaire APN : partenaires facilitant la gestion de la configuration
- Gestion du cycle de vie des comptes dans les environnements SaaS de compte par locataire sur AWS
- Gestion et surveillance de la limitation des API dans vos charges de travail
- Affichage des recommandations AWS Trusted Advisor à l'échelle avec AWS Organizations
- Automatiser l'augmentation des limites de service et le support aux entreprises avec AWS Control Tower

Vidéos connexes :

- AWS Live re:Inforce 2019 - Service Quotas
- Affichage et gestion des quotas de services AWS à l'aide de Service Quotas
- Démo des quotas AWS IAM

Outils associés :

- Amazon CodeGuru Reviewer
- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

REL01-BP02 Gestion des quotas de services entre les comptes et les régions

Si vous utilisez plusieurs comptes ou régions, demandez les quotas appropriés dans tous les environnements où vos charges de travail de production s'exécutent.

Résultat escompté : les services et les applications ne devraient pas être affectés par l'épuisement des quotas de services pour les configurations qui couvrent plusieurs comptes ou régions ou qui ont des conceptions de résilience utilisant le basculement de zone, de région ou de compte.

Anti-modèles courants :

- Laisser l'utilisation des ressources dans une région d'isolement se développer sans aucun mécanisme pour maintenir de la capacité dans les autres zones.
- Définition manuelle de tous les quotas de manière indépendante dans les régions d'isolement.
- Non-prise en considération de l'effet des architectures de résilience (par exemple, actives ou passives) dans les futurs besoins de quotas alors qu'une dégradation est observée dans la région non principale.
- Absence d'évaluation régulière des quotas et des changements qui s'imposent dans chaque région et chaque compte où la charge de travail s'exécute.
- Non-utilisation des modèles de demande de quotas pour demander des augmentations dans plusieurs régions et comptes.

- Absence de mise à jour des quotas de services pensant à tort que l'augmentation de quotas a des répercussions sur les coûts comme les demandes de réservation de capacité de calcul.

Avantages de l'établissement de cette bonne pratique : vérification que vous pouvez gérer votre charge de travail actuelle dans les régions ou les comptes secondaires si les services régionaux ne sont plus disponibles. Cela peut contribuer à limiter le nombre d'erreurs ou les niveaux de dégradations observés lors d'une perte de région.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : élevé

Directives d'implémentation

Les quotas de services sont suivis par compte. Sauf indication contraire, chaque quota est propre à une Région AWS. En plus des environnements de production, gérez également les quotas dans tous les autres environnements applicables de façon à ne pas entraver les tests et le développement. Pour maintenir un haut niveau de résilience, il convient d'évaluer constamment les quotas de services (que ce soit de façon automatisée ou manuelle).

Compte tenu de l'augmentation des charges de travail couvrant les régions en raison de la mise en œuvre de conceptions utilisant les approches active/active, active/passive : à chaud, active/passive : à froid et active/passive : veilleuse, il est essentiel de comprendre tous les niveaux de quotas de région et de compte. Les modèles de trafic passés ne permettent pas toujours de déterminer correctement si le quota de service est bien défini.

Tout aussi important, la valeur limite d'un nom de quota de service n'est pas toujours identique d'une région à l'autre. Ainsi, cette valeur peut être égale à cinq dans une région et à dix dans une autre. La gestion de ces quotas doit englober tous les services, comptes et régions identiques pour offrir une résilience cohérente dans des conditions de charge.

Rapprochez toutes les différences de quotas de services entre les différentes régions (région active ou région passive) et créez des processus permettant de rapprocher constamment ces différences. Les plans de test de basculements de régions passives sont rarement mis à l'échelle pour atteindre une capacité active de pointe, ce qui signifie que les exercices de simulation (« game day ») et les exercices de table (« table top ») ne permettent pas nécessairement d'identifier les différences dans les quotas de services entre les régions et donc de maintenir les limites adéquates.

Il est très important de suivre et d'évaluer la dérive des quotas de services, c'est-à-dire la situation dans laquelle les limites des quotas de services pour un quota nommé spécifique sont modifiées

dans une seule région, et non dans toutes les régions. Il doit être envisagé de changer le quota dans les régions qui présentent du trafic ou qui pourraient potentiellement en véhiculer.

- Sélectionnez les comptes et les régions appropriés en fonction de vos exigences de service, de latence, de réglementation et de reprise après sinistre (DR).
- Identifiez les quotas de services dans l'ensemble des comptes, régions et zones de disponibilité appropriés. Les limites s'appliquent au compte et à la région. Ces valeurs doivent être comparées pour repérer les différences.

Étapes d'implémentation

- Examinez les valeurs de Service Quotas susceptibles d'avoir transgressé le niveau d'utilisation à risque. AWS Trusted Advisor propose des alertes pour les violations de seuil de 80 % et 90 %.
- Examinez les valeurs de quotas de services dans les régions passives (dans une conception de type actif/passif). Vérifiez que la charge de travail s'exécutera correctement dans les régions secondaires en cas de défaillance dans la région principale.
- Automatisez l'évaluation pour identifier une éventuelle dérive de Service Quota entre des régions d'un même compte et agissez en conséquence pour changer les limites.
- Si la structure des unités d'organisation (UO) est prise en charge, les modèles de quotas de services doivent être mis à jour en fonction des changements apportés aux quotas qui doivent s'appliquer à plusieurs régions et comptes.
  - Créez un modèle et associez les régions au changement de quota.
  - Examinez tous les modèles de quotas de services existants pour y apporter les changements nécessaires (région, limites et comptes).

Ressources

Bonnes pratiques associées:

- REL01-BP01 Connaissance des quotas de service et des contraintes
- REL01-BP03 Prise en compte des quotas de services et des contraintes de service fixes dans l'architecture
- REL01-BP04 Surveiller et gérer les quotas
- REL01-BP05 Automatisation de la gestion des quotas

- REL01-BP06 Garantir un écart suffisant entre les quotas actuels et l'utilisation maximale pour permettre le basculement
- REL03-BP01 Choisissez comment segmenter votre charge de travail
- REL10-BP01 Déploiement de la charge de travail sur plusieurs emplacements
- REL11-BP01 Surveiller tous les composants de la charge de travail pour détecter les défaillances
- REL11-BP03 Automatiser la réparation sur toutes les couches
- REL12-BP04 Tester la résilience à l'aide de l'ingénierie du chaos

Documents connexes:

- AWS Pilier Fiabilité du cadre Well-Architected : disponibilité
- AWS Service Quotas (anciennement Service Limits)
- AWS Trusted Advisor Vérifications des bonnes pratiques (voir la section Service Limits)
- AWS Limit Monitor sur AWS Answers
- Amazon EC2 Service Limits
- Qu'est-ce que Service Quotas?
- Comment demander une augmentation de quota
- Points de terminaison et quotas de service
- Guide de l'utilisateur de Service Quotas
- Quota Monitor pour AWS
- AWS Fault Isolation Boundaries
- Disponibilité avec redondance
- AWS pour les données
- Qu'est-ce que l'intégration continue?
- Qu'est-ce que la livraison continue?
- Partenaire APN : partenaires facilitant la gestion de la configuration
- Gestion du cycle de vie des comptes dans les environnements SaaS de compte par locataire sur AWS
- Gestion et surveillance de la limitation des API dans vos charges de travail
- Affichage des recommandations AWS Trusted Advisor à l'échelle avec AWS Organizations

- Automatiser l’augmentation des limites de service et le support aux entreprises avec AWS Control Tower

Vidéos connexes:

- AWS Live re:Inforce 2019 - Service Quotas
- Affichage et gestion des quotas de services AWS à l’aide de Service Quotas
- Démo des quotas AWS IAM

Services connexes :

- Amazon CodeGuru Reviewer
- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

REL01-BP03 Prise en compte des quotas de services et des contraintes de service fixes dans l’architecture

Ayez conscience des quotas de services non modifiables, des contraintes de service et des limites de ressources physiques. Concevez des architectures pour les applications et les services afin d’éviter que ces limites n’aient un impact sur la fiabilité.

Par exemple, la bande passante du réseau, la taille de la charge utile des invocations de fonctions sans serveur, la limitation du taux de salves d’une passerelle API et les connexions simultanées d’utilisateurs à une base de données.

Résultat escompté : l'application ou le service fonctionne comme prévu dans des conditions de trafic normales et intenses. Ils ont été conçus pour fonctionner dans les limites des contraintes fixes ou des quotas de services de cette ressource.

Anti-modèles courants :

- Choix d'une conception qui utilise une ressource d'un service, sans savoir qu'il existe des contraintes de conception qui entraîneront l'échec de cette conception au fil des mises à l'échelle.
- Réalisation d'une évaluation comparative qui n'est pas réaliste et qui atteindra les quotas fixés par le service pendant les tests. Par exemple, l'exécution de tests à une limite de débordement mais pendant une durée prolongée.
- Le choix d'une conception qui ne peut pas se mettre à l'échelle ou être modifiée si des quotas de services fixes doivent être dépassés. Par exemple, une taille de charge utile SQS de 256 Ko.
- L'observabilité n'a pas été conçue et mise en œuvre pour surveiller et alerter sur les seuils des quotas de services qui pourraient être compromis lors d'événements à fort trafic

Avantages de l'établissement de cette bonne pratique : vérifier que l'application s'exécutera sous tous les niveaux de charge de service prévus, sans interruption ni dégradation.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : moyen

Directives d'implémentation

Contrairement aux quotas de services souples ou aux ressources qui peuvent être remplacées par des unités de plus grande capacité, les quotas fixes des services AWS ne peuvent pas être modifiés. Cela signifie que tous ces types de services AWS doivent être évalués en fonction des limites potentielles de capacité matérielle lorsqu'ils sont utilisés dans la conception d'une application.

Les limites strictes sont affichées dans la console Service Quotas. Si les colonnes indiquent `ADJUSTABLE = No`, le service est soumis à une limite stricte. Des limites strictes sont également indiquées dans les pages de configuration de certaines ressources. Par exemple, Lambda possède des limites strictes spécifiques qui ne peuvent pas être ajustées.

À titre d'exemple, lors de la conception d'une application python destinée à être exécutée dans une fonction Lambda, l'application doit être évaluée pour déterminer si Lambda risque de s'exécuter pendant plus de 15 minutes. Si le code peut fonctionner au-delà de cette limite de Service Quota, il faut envisager d'autres technologies ou conceptions. Si cette limite est atteinte après le déploiement

de la production, l'application subira une dégradation et des perturbations jusqu'à ce qu'il soit possible d'y remédier. Contrairement aux quotas souples, il n'existe aucune méthode permettant de passer à ces limites, même en cas d'événements d'urgence de gravité 1.

Une fois que l'application a été déployée dans un environnement de test, il convient d'utiliser des stratégies pour déterminer si des limites strictes peuvent être atteintes. Les tests de résistance, les tests de charge et les tests de chaos doivent faire partie du plan de test d'introduction.

Étapes d'implémentation

- Examinez la liste complète des services AWS qui pourraient être utilisés dans la phase de conception de l'application.
- Examinez les limites de quota flexible et de quota fixe pour tous ces services. Les limites ne sont pas toutes affichées dans la console Service Quotas. Certains services décrivent ces limites dans d'autres lieux.
- Lors de la conception de votre application, examinez les facteurs opérationnels et technologiques de votre charge de travail, tels que les résultats opérationnels, le cas d'utilisation, les systèmes dépendants, les objectifs de disponibilité et les objets de reprise après sinistre. Laissez vos facteurs commerciaux et technologiques guider le processus d'identification du système distribué qui convient à votre charge de travail.
- Analysez la charge de service dans les régions et les comptes. De nombreuses limites strictes se basent sur la région pour les services. Cependant, certaines limites sont basées sur le compte.
- Analysez les architectures de résilience pour l'utilisation des ressources lors d'une panne de zone et d'une panne régionale. Dans la progression des conceptions multirégionales utilisant des approches actives/actives, actives/passives : à chaud, actives/passives : à froid, et actives/passives : veilleuse, ces cas de panne entraîneront une utilisation plus importante. Cela crée un cas d'utilisation potentiel pour atteindre des limites strictes.

Ressources

Bonnes pratiques associées:

- REL01-BP01 Connaissance des quotas de service et des contraintes
- REL01-BP02 Gestion des quotas de services entre les comptes et les régions
- REL01-BP04 Surveiller et gérer les quotas
- REL01-BP05 Automatisation de la gestion des quotas

- REL01-BP06 Garantir un écart suffisant entre les quotas actuels et l'utilisation maximale pour permettre le basculement
- REL03-BP01 Choisissez comment segmenter votre charge de travail
- REL10-BP01 Déploiement de la charge de travail sur plusieurs emplacements
- REL11-BP01 Surveiller tous les composants de la charge de travail pour détecter les défaillances
- REL11-BP03 Automatiser la réparation sur toutes les couches
- REL12-BP04 Tester la résilience à l'aide de l'ingénierie du chaos

Documents connexes:

- AWS Pilier Fiabilité du cadre Well-Architected : disponibilité
- AWS Service Quotas (anciennement Service Limits)
- AWS Trusted Advisor Vérifications des bonnes pratiques (voir la section Service Limits)
- AWS Limit Monitor sur AWS Answers
- Amazon EC2 Service Limits
- Qu'est-ce que Service Quotas?
- Comment demander une augmentation de quota
- Points de terminaison et quotas de service
- Guide de l'utilisateur de Service Quotas
- Quota Monitor pour AWS
- AWS Fault Isolation Boundaries
- Disponibilité avec redondance
- AWS pour les données
- Qu'est-ce que l'intégration continue ?
- Qu'est-ce que la livraison continue ?
- Partenaire APN : partenaires facilitant la gestion de la configuration
- Gestion du cycle de vie des comptes dans les environnements SaaS de compte par locataire sur AWS
- Gestion et surveillance de la limitation des API dans vos charges de travail
- Affichage des recommandations AWS Trusted Advisor à l'échelle avec AWS Organizations
- Automatisation de l'augmentation des limites de service et du support aux entreprises avec AWS Control Tower

- Actions, ressources et clés de condition pour Service Quotas

Vidéos connexes :

- AWS Live re:Inforce 2019 - Service Quotas
- Affichage et gestion des quotas de services AWS à l'aide de Service Quotas
- Démo des quotas AWS IAM
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small

Outils associés :

- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

REL01-BP04 Surveiller et gérer les quotas

Évaluez votre utilisation potentielle et augmentez vos quotas de manière appropriée afin d'assurer une croissance planifiée de l'utilisation.

Résultat escompté : des systèmes actifs et automatisés de gestion et de surveillance ont été déployés. Ces solutions opérationnelles permettent de s'assurer que les seuils d'utilisation des quotas sont sur le point d'être atteints. Les changements de quotas demandés permettraient de remédier à ces problèmes de manière proactive.

Anti-modèles courants :

- Absence de configuration de la surveillance pour vérifier les seuils de quotas de services
- Absence de configuration de la surveillance des limites strictes, même si ces valeurs ne peuvent pas être modifiées.
- En supposant que le délai nécessaire pour demander et obtenir un changement de quota souple soit immédiat ou de courte durée.
- Configuration d'alarmes d'approche des quotas de services, mais sans processus sur la façon de répondre à une alerte.
- Configurez les alarmes uniquement pour les services pris en charge par les AWS Service Quotas et ne surveillez pas les autres AWS services.
- Non-prise en compte de la gestion des quotas pour les conceptions de résilience à régions multiples, comme les approches actives/actives, actives/passives : à chaud, actives/passives : à froid et actives/passives : veilleuse..
- Absence d'évaluation des différences de quotas entre les régions.
- Absence d'évaluation des besoins de chaque région pour une demande spécifique d'augmentation de quota.
- Absence d'utilisation de modèles pour la gestion des quotas multirégionaux.

Avantages de l'établissement de cette meilleure pratique : le suivi automatique des Quotas du AWS Service et le suivi de votre utilisation par rapport à ces quotas vous permettront de savoir quand vous approchez d'une limite de quota. Vous pouvez également utiliser ces données de surveillance pour limiter les dégradations dues à l'épuisement des quotas.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : moyen

Directives d'implémentation

Pour les services pris en charge, vous pouvez surveiller vos quotas en configurant différents services qui peuvent évaluer et ensuite envoyer des alertes ou des alarmes. Cela peut aider à surveiller l'utilisation et vous alerter sur l'approche des quotas. Ces alarmes peuvent être invoquées depuis AWS Config les fonctions Lambda CloudWatch, Amazon ou depuis. AWS Trusted Advisor Vous pouvez également utiliser des filtres métriques sur les CloudWatch journaux pour rechercher et extraire des modèles dans les journaux afin de déterminer si l'utilisation approche les seuils de quota.

Étapes d'implémentation

Pour la surveillance :

- Enregistrez la consommation des ressources actuelles (par exemple, les compartiments, ou les instances). Utilisez API les opérations de service, telles que Amazon EC2 `DescribeInstances`API, pour collecter la consommation actuelle de ressources.
- Saisissez vos quotas actuels qui sont essentiels et applicables aux services utilisant ce qui suit :
  - AWS Quotas de service
  - AWS Trusted Advisor
  - AWS documentation
  - AWS pages spécifiques au service
  - AWS Command Line Interface (AWS CLI)
  - AWS Cloud Development Kit (AWS CDK)
- Utilisez AWS Service Quotas, un AWS service qui vous permet de gérer vos quotas pour plus de 250 AWS services à partir d'un seul emplacement.
- Utilisez les limites de Trusted Advisor service pour surveiller vos limites de service actuelles à différents seuils.
- Utilisez l'historique des quotas de service (console ou AWS CLI) pour vérifier les augmentations régionales.
- Comparez les changements de quotas de services dans chaque région et chaque compte pour créer une équivalence, si nécessaire.

Pour la gestion :

- Automatisé : définissez une règle AWS Config personnalisée pour analyser les quotas de service entre les régions et comparer les différences.
- Automatisé : configurez une fonction Lambda programmée pour analyser les quotas de services dans les régions et comparer les différences.
- Manuel : Scannez les quotas de services par le biais AWS CLI de AWS la console ou analysez les quotas de services entre les régions et comparez les différences. API Signalez les différences.
- Si des différences de quotas sont identifiées entre les régions, demandez un changement de quota, si nécessaire.
- Passez en revue le résultat de toutes les demandes.

Ressources

Bonnes pratiques associées :

- REL01-BP01 Connaissance des quotas de service et des contraintes
- REL01-BP02 Gestion des quotas de services entre les comptes et les régions
- REL01-BP03 Prise en compte des quotas de services et des contraintes de service fixes dans l'architecture
- REL01-BP05 Automatisation de la gestion des quotas
- REL01-BP06 Garantir un écart suffisant entre les quotas actuels et l'utilisation maximale pour permettre le basculement
- REL03-BP01 Choisissez comment segmenter votre charge de travail
- REL10-BP01 Déploiement de la charge de travail sur plusieurs emplacements
- REL11-BP01 Surveiller tous les composants de la charge de travail pour détecter les défaillances
- REL11-BP03 Automatiser la réparation sur toutes les couches
- REL12-BP04 Tester la résilience à l'aide de l'ingénierie du chaos

Documents connexes :

- AWS Le pilier de fiabilité du framework Well-Architected : la disponibilité
- AWS Quotas de service (anciennement appelés limites de service)
- AWS Trusted Advisor Contrôles des meilleures pratiques (voir la section Limites de service)
- AWS limitez le nombre de AWS réponses
- Limites EC2 de service Amazon
- Qu'est-ce que Service Quotas ?
- Comment demander une augmentation de quota
- Points de terminaison et quotas de service
- Guide de l'utilisateur de Service Quotas
- Quota Monitor pour AWS
- AWS Limites d'isolation des défauts
- Disponibilité avec redondance

- AWS pour les données
- Qu’est-ce que l’intégration continue ?
- Qu’est-ce que la livraison continue ?
- APNPartenaire : partenaires qui peuvent vous aider à gérer la configuration
- Gestion du cycle de vie des comptes dans les environnements account-per-tenant SaaS sur AWS
- Gestion et surveillance de la API régulation de vos charges de travail
- Consultez les AWS Trusted Advisor recommandations à grande échelle avec AWS Organizations
- Automatiser l'augmentation des limites de service et le support aux entreprises avec AWS Control Tower
- Actions, ressources et clés de condition pour Service Quotas

Vidéos connexes :

- AWS Live Re:inforce 2019 - Service Quotas
- Afficher et gérer les quotas pour les AWS services à l'aide de quotas de service
- AWS IAMDémo de quotas
- AWS re:Invent 2018 : Boucles serrées et ouverture d'esprit : comment prendre le contrôle des systèmes, grands et petits

Outils associés :

- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

REL01-BP05 Automatisation de la gestion des quotas

Les quotas de service, également appelés limites dans les services AWS, sont les valeurs maximales pour les ressources de votre Compte AWS. Chaque service AWS définit un ensemble de quotas et leurs valeurs par défaut. Pour permettre à votre charge de travail d'accéder à toutes les ressources dont elle a besoin, vous devrez peut-être augmenter les valeurs de vos quotas de service.

L'augmentation de la consommation de AWS ressources par la charge de travail peut menacer la stabilité de la charge de travail et avoir un impact sur l'expérience utilisateur en cas de dépassement des quotas. Mettez en œuvre des outils qui vous alerteront quand votre charge de travail approchera des limites et envisagez de créer automatiquement des demandes d'augmentation de quotas.

Résultat escompté : les quotas sont configurés de manière appropriée pour les charges de travail exécutées dans chaque Compte AWS et chaque région.

Anti-modèles courants :

- vous ne tenez pas compte des quotas et ne les ajustez pas de manière appropriée pour répondre aux exigences de charge de travail.
- Vous suivez les quotas et l'utilisation à l'aide de méthodes qui peuvent devenir obsolètes, comme les feuilles de calcul.
- Vous ne mettez à jour les limites de service que lors de planifications périodiques.
- Votre organisation ne dispose pas de processus opérationnels permettant de revoir les quotas existants et de demander des augmentations de quotas de service si nécessaire.

Avantages liés au respect de cette bonne pratique :

- Résilience améliorée de la charge de travail : vous évitez les erreurs causées par le dépassement des quotas de ressources AWS.
- Reprise après sinistre simplifiée : vous pouvez réutiliser les mécanismes de gestion automatique des quotas intégrés dans la région principale lors de la configuration de la reprise après sinistre dans une autre Région AWS.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : moyen

Directives d'implémentation

Consultez les quotas actuels et suivez la consommation continue des quotas via des mécanismes tels que la console AWS Service Quotas, AWS Command Line Interface (AWS CLI) et les kits AWS SDK. Vous pouvez également intégrer vos bases de données de gestion des configurations (CMDB) et vos systèmes de gestion des services informatiques (ITSM) avec les API AWS Service Quotas.

Générez des alertes automatiques si l'utilisation des quotas atteint les seuils que vous avez définis, et définissez un processus pour soumettre les demandes d'augmentation de quotas lorsque vous recevez des alertes. Si la charge de travail sous-jacente est essentielle pour votre entreprise, vous pouvez automatiser les demandes d'augmentation de quotas, mais testez soigneusement l'automatisation pour éviter le risque d'une action incontrôlée, telle qu'une boucle de rétroaction sur la croissance.

Les petites augmentations de quotas sont souvent automatiquement approuvées. Les demandes de quotas plus importants peuvent nécessiter un traitement manuel par AWS Support, et leur examen et leur traitement peuvent prendre plus de temps. Prévoyez du temps supplémentaire pour traiter plusieurs demandes ou des demandes d'augmentation importante.

Étapes d'implémentation

- Mettez en œuvre une surveillance automatisée des quotas de service et émettez des alertes si la croissance de l'utilisation des ressources de votre charge de travail approche de vos limites de quotas. Par exemple, Quota Monitor for AWS peut assurer la surveillance automatisée des quotas de service. Cet outil s'intègre à AWS Organizations et se déploie à l'aide de Cloudformation StackSets, de sorte que les nouveaux comptes sont automatiquement surveillés à leur création.
- Utilisez des fonctionnalités telles que les modèles de demande de quotas de service ou AWS Control Tower pour simplifier la configuration de Service Quotas pour les nouveaux comptes.
- Créez des tableaux de bord de l'utilisation actuelle de vos quotas de service dans tous les Comptes AWS et toutes les régions, et faites-y référence si nécessaire pour éviter de dépasser vos quotas. Le tableau de bord Trusted Advisor Organizational (TAO), qui fait partie du framework Cloud Intelligence Dashboards, peut vous aider à rapidement prendre en main un tel tableau de bord.
- Effectuez le suivi des demandes d'augmentation de limite de service. La page Consolidated Insights from Multiple Accounts (CIMA) peut fournir une vue de toutes vos demandes au niveau de l'organisation.

- Testez la génération d'alertes et l'automatisation de toute demande d'augmentation de quotas en définissant des seuils de quotas inférieurs dans les comptes hors production. N'effectuez pas ces tests dans un compte de production.

Ressources

Bonnes pratiques associées :

- OPS10-BP07 Automatisation des réponses aux événements

Documents connexes :

- Partenaire APN : partenaires facilitant la gestion de la configuration
- AWS Marketplace : produits CMDB facilitant le suivi des limites
- AWS Service Quotas (anciennement Service Limits)
- Vérifications des bonnes pratiques AWS Trusted Advisor (voir la section Service Limits)
- Solution Quota Monitor sur AWS – Solution AWS
- Qu'est-ce que Service Quotas ?
- Que sont les modèles de demande de quotas de service ?

Vidéos connexes :

- AWS Live re:Inforce 2019 - Service Quotas
- Automatisation de l'augmentation des limites de service et du support aux entreprises avec AWS Control Tower

Outils associés :

- Quota Monitor for AWS

REL01-BP06 Garantir un écart suffisant entre les quotas actuels et l'utilisation maximale pour permettre le basculement

Cet article explique comment maintenir l'espace entre le quota de ressources et votre utilisation, et comment cela peut être bénéfique pour votre organisation. Une fois que vous avez fini d'utiliser

une ressource, le quota d'utilisation peut continuer à prendre en compte cette ressource. Cela peut entraîner une défaillance ou une inaccessibilité de la ressource. Empêchez la défaillance de ressource en vérifiant que vos quotas couvrent le chevauchement des ressources inaccessibles et leurs remplacements. Prenez en compte les cas d'utilisation tels que les pannes de réseau, la panne de la zone de disponibilité ou les pannes régionales lorsque vous calculez cet écart.

Résultat escompté : les défaillances mineures ou importantes en matière de ressources ou d'accessibilité des ressources peuvent être couvertes dans les limites des seuils de service actuels. Les pannes de zone, les pannes de réseau, voire les pannes régionales ont été prises en compte dans la planification des ressources.

Anti-modèles courants :

- Définition de quotas de service en fonction des quotas actuels sans tenir compte des scénarios de basculement.
- Ne pas tenir compte des principes de stabilité statique lors du calcul du quota de pointe pour un service.
- Ne pas tenir compte du potentiel des ressources inaccessibles dans le calcul du quota total nécessaire pour chaque région.
- Ne pas prendre en compte les limites d'isolement des pannes de service AWS pour certains services et leurs potentiels schémas d'utilisation anormaux.

Avantages liés au respect de cette bonne pratique : lorsque des interruptions de service ont un impact sur la disponibilité des applications, utilisez le cloud pour mettre en œuvre des stratégies de reprise après ces événements. Un exemple de stratégie consiste à créer des ressources supplémentaires pour remplacer les ressources inaccessibles afin de s'adapter aux conditions de basculement sans épuiser votre limite de service.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : moyen

Directives d'implémentation

Lors de l'évaluation d'une limite de quotas, il faut tenir compte des cas de basculement qui pourraient survenir en raison d'une certaine dégradation. Prenez en compte les cas de basculement suivants.

- VPC perturbé ou inaccessible.
- Sous-réseau inaccessible.
- Zone de disponibilité dégradée qui a un impact sur l'accessibilité des ressources.

- Des itinéraires de mise en réseau ou des points d'entrée et de sortie sont bloqués ou modifiés.
- Région dégradée qui a un impact sur l'accessibilité des ressources.
- Sous-ensemble de ressources affectées par une défaillance dans une région ou une zone de disponibilité.

La décision de basculement est unique selon la situation, car l'impact sur l'entreprise peut varier. Abordez la planification de la capacité des ressources sur le site de basculement et les quotas des ressources avant de décider de basculer une application ou un service.

Tenez compte des pics d'activité supérieurs à la normale lors de l'examen des quotas pour chaque service. Ces pics peuvent être liés à des ressources inaccessibles en raison de la mise en réseau ou des autorisations, mais toujours actives. Les ressources actives non résiliées comptent dans la limite du quota de service.

Étapes d'implémentation

- Maintenez un espace entre votre quota de service et votre utilisation maximale pour faire face à un basculement ou à une perte d'accessibilité.
- Déterminez vos quotas de service. Tenez compte des modèles de déploiement typiques, des exigences de disponibilité et de la croissance de la consommation.
- Demandez des augmentations de quota si nécessaire Prévoyez un temps d'attente pour la demande d'augmentation de quota.
- Déterminez vos exigences de fiabilité (également connues sous le nom de « nombre de neuf »).
- Comprenez les scénarios de panne potentiels tels que la perte d'un composant, d'une zone de disponibilité ou d'une région.
- Définissez votre méthodologie de déploiement (par exemple, canary, bleu/vert, rouge/noir et roulement).
- Incluez une mémoire tampon appropriée pour la limite actuelle de quota. Un exemple de tampon pourrait être de 15 %.
- Ajoutez les calculs de stabilité statique (zonale et régionale), le cas échéant.
- Anticipez la croissance de la consommation et surveillez vos tendances de consommation.
- Songez à l'impact de la stabilité statique pour vos charges de travail les plus critiques. Évaluez les ressources conformes à un système statiquement stable dans toutes les régions et zones de disponibilité.

- Envisagez l'utilisation de réserves de capacité à la demande pour programmer la capacité avant tout basculement. L'implémentation de cette stratégie est utile pour les calendriers d'activité critiques afin de réduire les risques potentiels liés à l'obtention de la bonne quantité et du bon type de ressources lors du basculement.

Ressources

Bonnes pratiques associées:

- REL01-BP01 Connaissance des quotas de service et des contraintes
- REL01-BP02 Gestion des quotas de services entre les comptes et les régions
- REL01-BP03 Prise en compte des quotas de services et des contraintes de service fixes dans l'architecture
- REL01-BP04 Surveiller et gérer les quotas
- REL01-BP05 Automatisation de la gestion des quotas
- REL03-BP01 Choisissez comment segmenter votre charge de travail
- REL10-BP01 Déploiement de la charge de travail sur plusieurs emplacements
- REL11-BP01 Surveiller tous les composants de la charge de travail pour détecter les défaillances
- REL11-BP03 Automatiser la réparation sur toutes les couches
- REL12-BP04 Tester la résilience à l'aide de l'ingénierie du chaos

Documents connexes:

- AWS Pilier Fiabilité du cadre Well-Architected : disponibilité
- AWS Service Quotas (anciennement Service Limits)
- AWS Trusted Advisor Vérifications des bonnes pratiques (voir la section Service Limits)
- AWS Limit Monitor sur AWS Answers
- Amazon EC2 Service Limits
- Qu'est-ce que Service Quotas?
- Comment demander une augmentation de quota
- Points de terminaison et quotas de service
- Guide de l'utilisateur de Service Quotas

- Quota Monitor pour AWS
- AWS Fault Isolation Boundaries
- Disponibilité avec redondance
- AWS pour les données
- Qu’est-ce que l’intégration continue?
- Qu’est-ce que la livraison continue?
- Partenaire APN : partenaires facilitant la gestion de la configuration
- Gestion du cycle de vie des comptes dans les environnements SaaS de compte par locataire sur AWS
- Gestion et surveillance de la limitation des API dans vos charges de travail
- Affichage des recommandations AWS Trusted Advisor à l’échelle avec AWS Organizations
- Automatisation de l’augmentation des limites de service et du support aux entreprises avec AWS Control Tower
- Actions, ressources et clés de condition pour Service Quotas

Vidéos connexes:

- AWS Live re:Inforce 2019 - Service Quotas
- Affichage et gestion des quotas de services AWS à l’aide de Service Quotas
- AWS Démo des quotas IAM
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small

Outils associés:

- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor

- AWS CDK
- AWS Systems Manager
- AWS Marketplace

Planification de votre topologie réseau

Les charges de travail existent souvent dans plusieurs environnements. Il s'agit notamment de plusieurs environnements Cloud (accessibles au public et privés) et éventuellement de votre infrastructure de centre de données existante. Les plans doivent tenir compte de facteurs liés au réseau : connectivité intrasystème et intersystème, gestion des adresses IP publiques, gestion des adresses IP privées et résolution des noms de domaine.

Lorsque vous concevez des systèmes à l'aide de réseaux basés sur des adresses IP, vous devez planifier la topologie du réseau et l'adressage en prévision d'éventuelles pannes et pour faire face à une future croissance et à l'intégration à d'autres systèmes à leurs réseaux.

Virtual Private Cloud (VPC) Amazon vous permet de mettre en service une section du cloud AWS isolée et privée, dans laquelle vous pouvez lancer des ressources AWS dans un réseau virtuel.

Bonnes pratiques

- REL02-BP01 Utiliser une connectivité réseau hautement disponible pour vos points de terminaison publics de charge de travail
- REL02-BP02 Mettre en service une connectivité redondante entre les réseaux privés dans le cloud et les environnements sur site
- REL02-BP03 S'assurer que l'allocation des sous-réseaux IP tient compte de l'expansion et de la disponibilité
- REL02-BP04 Préférer les topologies en étoile au maillage « many-to-many »
- REL02-BP05 Appliquer des plages d'adresses IP privées sans chevauchement dans tous les espaces d'adressage privés où elles sont connectées

REL02-BP01 Utiliser une connectivité réseau hautement disponible pour vos points de terminaison publics de charge de travail

La mise en place d'une connectivité réseau hautement disponible aux points de terminaison publics de vos charges de travail peut vous aider à réduire les temps d'arrêt dus à la perte de connectivité

et à améliorer la disponibilité et le SLA de votre charge de travail. Pour ce faire, utilisez le DNS hautement disponible, les réseaux de diffusion de contenu (CDN), des passerelles API, l'équilibrage de charge ou les proxys inverses.

Résultat escompté : il est essentiel de planifier, de créer et de rendre opérationnelle une connectivité réseau hautement disponible pour vos points de terminaison publics. Si votre charge de travail devient inaccessible en raison d'une perte de connectivité, même si elle est en cours d'exécution et disponible, vos clients verront votre système comme étant en panne. En combinant une connectivité réseau hautement disponible et résiliente pour les points de terminaison publics de votre charge de travail, ainsi qu'une architecture résiliente pour votre charge de travail elle-même, vous pouvez offrir la meilleure disponibilité et le meilleur niveau de service possible à vos clients.

AWS Global Accelerator, Amazon CloudFront, Amazon API Gateway, les URL de fonction AWS Lambda, les API AWS AppSync et Elastic Load Balancing (ELB) fournissent tous des points de terminaison publics hautement disponibles. Amazon Route 53 fournit un service DNS hautement disponible pour la résolution des noms de domaine afin de vérifier que vos adresses de point de terminaison publiques peuvent être résolues.

Vous pouvez également évaluer des appliances logicielles AWS Marketplace pour l'équilibrage de charge et les proxys.

Anti-modèles courants :

- Concevoir une charge de travail hautement disponible sans planifier le DNS et la connectivité réseau pour la haute disponibilité.
- Utiliser des adresses Internet publiques sur des instances ou des conteneurs individuels et gestion de la connectivité à ces adresses avec le DNS.
- Utiliser des adresses IP au lieu des noms de domaine pour localiser les services.
- Ne pas tester des scénarios où la connectivité à vos points de terminaison publics est perdue.
- Ne pas analyser les besoins en débit du réseau et les modèles de distribution.
- Ne pas tester et planifier des scénarios dans lesquels la connectivité du réseau Internet aux points de terminaison publics de votre charge de travail pourrait être interrompue.
- Fournir du contenu (comme les pages web, les ressources statiques ou les fichiers multimédias) à une grande zone géographique sans utiliser un réseau de diffusion de contenu.
- Ne pas se préparer aux attaques par déni de service distribué (DDoS). Les attaques DDoS risquent d'interrompre le trafic légitime et de réduire la disponibilité pour vos utilisateurs.

Avantages du respect de cette bonne pratique : la conception d'une connectivité réseau hautement disponible et résiliente garantit que votre charge de travail est accessible et disponible pour vos utilisateurs.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : élevé

Directives d'implémentation

Le routage du trafic est au cœur de la mise en place d'une connectivité réseau hautement disponible pour vos points de terminaison publics. Pour vérifier que votre trafic est en mesure d'atteindre les points de terminaison, le DNS doit être capable de résoudre les noms de domaine à leurs adresses IP correspondantes. Utilisez un système de nom de domaine (DNS) hautement disponible et évolutif tel qu'Amazon Route 53 pour gérer les enregistrements DNS de votre domaine. Vous pouvez également utiliser les surveillances de l'état fournies par Amazon Route 53. Les surveillances de l'état permettent de s'assurer que votre application est accessible, disponible et fonctionnelle. Elles peuvent être configurées de manière à imiter le comportement de l'utilisateur, comme la demande d'une page web ou d'une URL spécifique. En cas de panne, Amazon Route 53 répond aux demandes de résolution DNS et dirige uniquement le trafic vers les points de terminaison en bonne santé. Vous pouvez également envisager d'utiliser les fonctionnalités de Geo DNS et de routage basé sur la latence offertes par Amazon Route 53.

Pour vérifier que votre charge de travail elle-même est hautement disponible, utilisez Elastic Load Balancing (ELB). Amazon Route 53 peut être utilisé pour cibler le trafic vers ELB, qui le distribue aux instances de calcul cibles. Vous pouvez également utiliser Amazon API Gateway avec AWS Lambda pour une solution sans serveur. Les clients peuvent également exécuter des charges de travail dans plusieurs Régions AWS. Avec un modèle actif/actif multisite, la charge de travail peut desservir le trafic provenant de plusieurs régions. Avec un schéma actif/passif multisite, la charge de travail sert le trafic provenant de la région active tandis que les données sont répliquées vers la région secondaire et deviennent actives en cas de panne dans la région principale. Les surveillances de l'état de Route 53 peuvent ensuite être utilisées pour contrôler le basculement DNS depuis n'importe quel point de terminaison d'une région principale vers un point de terminaison d'une région secondaire, en vérifiant que votre charge de travail est accessible et disponible pour vos utilisateurs.

Amazon CloudFront fournit une API simple pour distribuer du contenu avec une faible latence et des taux de transfert de données élevés en répondant aux demandes à l'aide d'un réseau d'emplacements périphériques dans le monde entier. Les réseaux de diffusion de contenu (CDN) desservent les clients en proposant un contenu situé ou mis en cache à un endroit proche de l'utilisateur. Cela améliore également la disponibilité de votre application, car la charge de contenu est transférée de vos serveurs vers les emplacements périphériques de CloudFront. Les

emplacements périphériques et les caches périphériques régionaux conservent des copies en cache de votre contenu à proximité de vos utilisateurs, ce qui permet une récupération rapide et augmente l'accessibilité et la disponibilité de votre charge de travail.

Pour les charges de travail avec des utilisateurs dispersés géographiquement, AWS Global Accelerator améliore la disponibilité et les performances des applications. AWS Global Accelerator fournit des adresses IP statiques Anycast qui servent de point d'entrée fixe à votre application hébergée dans une ou plusieurs Régions AWS. Cela permet au trafic d'entrer sur le réseau mondial AWS aussi près que possible de vos utilisateurs, améliorant ainsi l'accessibilité et la disponibilité de votre charge de travail. AWS Global Accelerator surveille également l'état de santé des points de terminaison de vos applications en utilisant la surveillance de l'état TCP, HTTP et HTTPS. Toute modification de l'état ou de la configuration de vos points de terminaison permet la redirection du trafic utilisateur vers des points de terminaison sains qui offrent les meilleures performances et la meilleure disponibilité à vos utilisateurs. De plus, AWS Global Accelerator est conçu pour être isolé des pannes et utilise deux adresses IPv4 statiques qui sont desservies par des zones réseau indépendantes, ce qui augmente la disponibilité de vos applications.

Pour aider à protéger les clients contre les attaques DDoS, AWS fournit AWS Shield Standard. Shield Standard est activé automatiquement et protège contre les attaques d'infrastructure courantes (couches 3 et 4) telles que les inondations SYN/UDP et les attaques par réflexion afin de garantir la haute disponibilité de vos applications sur AWS. Pour bénéficier de protections supplémentaires contre des attaques plus sophistiquées et plus importantes (comme les inondations UDP), les attaques par épuisement d'état (comme les inondations TCP SYN), et pour aider à protéger vos applications fonctionnant sur Amazon Elastic Compute Cloud (Amazon EC2), Elastic Load Balancing (ELB), Amazon CloudFront, AWS Global Accelerator et Route 53, vous pouvez envisager d'utiliser AWS Shield Advanced. Pour se protéger contre les attaques au niveau de la couche application, comme les inondations HTTP POST ou GET, utilisez AWS WAF. AWS WAF peut utiliser les adresses IP, les en-têtes HTTP, le corps HTTP, les chaînes URI, l'injection SQL et les conditions de script intersite pour déterminer si une requête doit être bloquée ou autorisée.

Étapes d'implémentation

1. Définir un DNS hautement disponible : Amazon Route 53 est un service web de système de nom de domaine (DNS) hautement disponible et évolutif. Route 53 connecte les demandes des utilisateurs aux applications Internet exécutées sur AWS ou sur site. Pour plus d'informations, consultez Configuration d'Amazon Route 53 en tant que service DNS.
2. Configurez la surveillance de l'état : lorsque vous utilisez Route 53, vérifiez que seules les cibles saines sont résolubles. Commencez par la création de surveillance de l'état Route 53 et la

configuration du basculement DNS. Il est important de tenir compte des aspects suivants lors de la mise en place des surveillances de l'état :

a. Comment Amazon Route 53 détermine si une surveillance de l'état est saine

b. Création, mise à jour et suppression de surveillances de l'état

c. Surveillance du statut de la surveillance de l'état et obtention de notifications

d. Bonnes pratiques relatives à Amazon Route 53 DNS

3. Connectez votre service DNS à vos points de terminaison.

   a. Lorsque vous utilisez Elastic Load Balancing comme cible pour votre trafic, créez un enregistrement d'alias à l'aide d'Amazon Route 53 qui pointe vers le point de terminaison régional de votre équilibreur de charge. Pendant la création de l'enregistrement de l'alias, réglez l'option « Évaluer l'état de la cible » sur « Oui ».

   b. Pour les charges de travail sans serveur ou les API privées lorsqu'une passerelle API est utilisée, utilisez Route 53 pour diriger le trafic vers la passerelle API.

4. Choisissez un réseau de diffusion de contenu.

   a. Pour diffuser du contenu en utilisant des emplacements périphériques plus proches de l'utilisateur, commencez par comprendre comment CloudFront diffuse le contenu.

   b. Démarrez avec une distribution CloudFront simple. CloudFront comprend alors l'endroit d'où vous souhaitez que le contenu soit diffusé, ainsi que les détails concernant le suivi et la gestion de la diffusion du contenu. Il est important de comprendre et de prendre en compte les aspects suivants lors de la mise en place de la distribution CloudFront :

      i. Fonctionnement de la mise en cache avec les emplacements périphériques CloudFront

      ii. Augmentation de la proportion de demandes servies directement à partir des caches CloudFront (taux d'accès au cache)

      iii. Utilisation d'Amazon CloudFront Origin Shield

      iv. Optimisation de la haute disponibilité avec le basculement d'origine CloudFront

5. Configurez la protection de la couche d'application : AWS WAF vous aide à vous protéger contre les exploits et les bots web courants qui peuvent affecter la disponibilité, compromettre la sécurité ou consommer des ressources excessives. Pour mieux comprendre, découvrez comment AWS WAF fonctionne et quand vous serez prêt à mettre en œuvre des protections contre les inondations HTTP POST AND GET de la couche d'application, consultez Démarrer avec AWS WAF. Vous pouvez également utiliser AWS WAF avec CloudFront. Consultez la documentation sur le fonctionnement de AWS WAF avec des fonctionnalités d'Amazon CloudFront.

6. Configurez une protection DDoS supplémentaire : par défaut, tous les clients AWS bénéficient d'une protection contre les attaques DDoS les plus fréquentes au niveau de la couche réseau et de la couche transport qui ciblent votre site web ou votre application avec AWS Shield Standard, et ce sans frais supplémentaires. Pour une protection supplémentaire des applications connectées à Internet exécutées sur Amazon EC2, Elastic Load Balancing, Amazon CloudFront, AWS Global Accelerator et Amazon Route 53, vous pouvez prendre en compte AWS Shield Advanced et examiner des exemples d'architectures résilientes aux attaques DDoS. Pour protéger votre charge de travail et vos points de terminaison publics contre les attaques DDoS, consultez Démarrer avec AWS Shield Advanced.

Ressources

Bonnes pratiques associées:

- REL10-BP01 Déploiement de la charge de travail sur plusieurs emplacements
- REL11-BP04 S'appuyer sur le plan de données et non sur le plan de contrôle pendant la récupération
- REL11-BP06 Envoyer des notifications lorsque des événements affectent la disponibilité

Documents connexes :

- Partenaire APN : partenaires pouvant vous aider à planifier votre mise en réseau
- AWS Marketplace pour l'infrastructure réseau
- Qu'est-ce que AWS Global Accelerator?
- Qu'est-ce qu'Amazon CloudFront ?
- Qu'est-ce qu'Amazon Route 53 ?
- Qu'est-ce qu'Elastic Load Balancing ?
- Capacité de connectivité réseau : établir les bases de votre cloud
- Qu'est-ce qu'Amazon API Gateway ?
- Que sont AWS WAF, AWS Shield et AWS Firewall Manager ?
- Qu'est-ce qu'Amazon Application Recovery Controller ?
- Configurer des surveillances de l'état personnalisées pour le basculement DNS

Vidéos connexes :

- AWS re:Invent 2022 - Improve performance and availability with AWS Global Accelerator
- AWS re:Invent 2020: Global traffic management with Amazon Route 53
- AWS re:Invent 2022 - Operating highly available Multi-AZ applications
- AWS re:Invent 2022 - Dive deep on AWS networking infrastructure
- AWS re:Invent 2022 - Building resilient networks

Exemples connexes :

- Reprise après sinistre avec Amazon Application Recovery Controller (ARC)
- AWS Global Accelerator Atelier

REL02-BP02 Mettre en service une connectivité redondante entre les réseaux privés dans le cloud et les environnements sur site

Mettez en œuvre une redondance dans vos connexions entre les réseaux privés dans le cloud et les environnements sur site pour obtenir la résilience de la connectivité. Cela peut se faire en déployant au moins deux liens et chemins de trafic, afin de préserver la connectivité en cas de défaillance du réseau.

Anti-modèles courants :

- Vous dépendez d'une seule connexion réseau, ce qui crée un point de défaillance unique.
- Vous utilisez un seul tunnel VPN ou plusieurs tunnels qui aboutissent à la même zone de disponibilité.
- Vous dépendez d'un seul fournisseur de services Internet (FSI) pour la connectivité VPN, ce qui peut entraîner des défaillances complètes en cas de panne de ce dernier.
- Vous n'implémentez pas de protocoles de routage dynamique tels que BGP, qui sont essentiels pour rediriger le trafic lors de perturbations du réseau.
- Vous ignorez les limites de bande passante des tunnels VPN et surestimez leurs capacités de secours.

Avantages du respect de cette bonne pratique : l'implémentation d'une connectivité redondante entre votre environnement cloud et votre environnement d'entreprise/sur site permet aux services dépendants entre les deux environnements de communiquer de manière fiable.

Niveau d’exposition au risque si cette bonne pratique n’est pas respectée : élevé

Directives d’implémentation

Lorsque vous utilisez AWS Direct Connect pour connecter votre réseau sur site à AWS, vous pouvez obtenir une résilience réseau maximale (SLA de 99,99 %) en utilisant des connexions distinctes qui mènent à des appareils distincts dans plusieurs emplacements sur site et plusieurs emplacements AWS Direct Connect. Cette topologie offre une résilience contre les pannes d’appareils, les problèmes de connectivité et les pannes complètes d’un site. Vous pouvez également atteindre une résilience élevée (SLA de 99,9 %) en utilisant deux connexions individuelles à plusieurs emplacements (chaque emplacement sur site étant connecté à un seul emplacement Direct Connect). Cette approche assure une protection contre les interruptions de connectivité causées par des coupures de fibre ou des pannes d’appareils, et contribue à pallier des défaillances complètes d’un site. Direct Connect Resiliency Toolkit peut vous aider à concevoir votre topologie AWS Direct Connect.

Vous pouvez également envisager que AWS Site-to-Site VPN mène à une passerelle AWS Transit Gateway faisant office de solution de secours à moindre coût de votre connexion AWS Direct Connect principale. Cette configuration permet un routage ECMP (Equal cost multi-path) via plusieurs tunnels VPN, permettant d’obtenir un débit allant jusqu’à 50 Gbit/s, bien que chaque tunnel VPN soit plafonné à 1,25 Gbit/s. Il est toutefois important de noter qu’AWS Direct Connect constitue toujours le choix le plus efficace pour réduire au maximum les perturbations du réseau et assurer une connectivité stable.

Lorsque vous utilisez des VPN sur Internet pour connecter votre environnement cloud à votre centre de données sur site, configurez deux tunnels VPN dans le cadre d’une connexion VPN unique de site à site. Chaque tunnel doit mener à une zone de disponibilité différente pour garantir la haute disponibilité, et utiliser du matériel redondant pour éviter une panne d’appareil sur site. En outre, envisagez plusieurs connexions Internet provenant de différents fournisseurs de services Internet (FSI) à votre emplacement sur site pour éviter une interruption complète de la connectivité VPN en cas de panne au niveau d’un seul FSI. Pour garantir une connectivité à haute disponibilité, il convient de sélectionner des fournisseurs de services Internet offrant un routage et une infrastructure diversifiés, notamment avec des chemins physiques distincts vers les points de terminaison AWS.

Outre la redondance physique avec plusieurs connexions AWS Direct Connect et plusieurs tunnels VPN (ou une combinaison des deux), il est essentiel de mettre en œuvre un routage dynamique via le protocole de passerelle frontière (BGP). Le protocole BGP dynamique permet de rediriger automatiquement le trafic d’un chemin vers un autre en fonction des conditions en temps réel du réseau et des politiques configurées. Ce comportement dynamique est particulièrement utile pour

maintenir la disponibilité du réseau et la continuité des services en cas de panne de liaison ou de réseau. Il sélectionne rapidement des chemins alternatifs, améliorant ainsi la résilience et la fiabilité du réseau.

Étapes d'implémentation

- Bénéficiez d'une connectivité hautement disponible entre AWS et votre environnement sur site.
  - Utilisez plusieurs connexions AWS Direct Connect ou tunnels VPN entre des réseaux privés déployés séparément.
  - Utilisez plusieurs emplacements Direct Connect pour une haute disponibilité.
  - Si vous utilisez plusieurs Régions AWS, mettez en œuvre la redondance dans au moins deux d'entre elles.
- Utilisez AWS Transit Gateway, dans la mesure du possible, pour mettre fin à votre connexion VPN.
- Évaluez les appliances AWS Marketplace permettant de mettre fin aux VPN ou d'étendre votre SD-WAN à AWS. Si vous utilisez des appliances AWS Marketplace, déployez des instances redondantes pour une plus haute disponibilité dans différentes zones de disponibilité.
- Assurez-vous que vous disposez d'une connexion redondante à votre environnement sur site.
  - Il se peut que vous ayez besoin de connexions redondantes à plusieurs Régions AWS pour répondre à vos besoins en matière de disponibilité.
  - Utilisez la boîte à outils de résilience Direct Connect pour démarrer.

Ressources

Documents connexes :

- Recommandations de résilience AWS Direct Connect
- Utilisation de connexions Site-to-Site VPN redondantes pour fournir un basculement
- Stratégies de routage et communautés BGP (Border Gateway Protocol)
- Configurations actives/actives et actives/passives dans AWS Direct Connect
- Partenaire APN : partenaires pouvant vous aider à planifier votre mise en réseau
- AWS Marketplace pour l'infrastructure réseau
- Livre blanc sur les options de connectivité d'Amazon Virtual Private Cloud
- Création d'une infrastructure réseau AWS multi-VPC évolutive et sécurisée
- Utilisation de connexions Site-to-Site VPN redondantes pour fournir un basculement

- Utiliser la boîte à outils de résilience Direct Connect pour démarrer
- Points de terminaison d'un VPC et services de point de terminaison d'un VPC (AWS PrivateLink)
- Qu'est-ce qu'Amazon VPC ?
- Qu'est-ce qu'une passerelle de transit ?
- Qu'est-ce qu AWS Site-to-Site VPN?
- Utilisation des passerelles Direct Connect

Vidéos connexes :

- AWS re:Invent 2018: Advanced VPC Design and New Capabilities for Amazon VPC
- AWS re:Invent 2019: AWS Transit Gateway reference architectures for many VPCs

REL02-BP03 S'assurer que l'allocation des sous-réseaux IP tient compte de l'expansion et de la disponibilité

Les plages d'adresses IP d'Amazon VPC doivent être assez grandes pour répondre aux exigences de charge de travail, y compris en prévision d'une future expansion ou allocation d'adresses IP aux sous-réseaux sur les zones de disponibilité. Cela inclut les équilibreurs de charge, les instances EC2 et les applications basées sur conteneur.

Lorsque vous planifiez votre topologie de réseau, la première étape consiste à définir l'espace d'adressage IP lui-même. Des plages d'adresses IP privées (conformément aux directives RFC 1918) doivent être allouées pour chaque VPC. Vous devez remplir les exigences suivantes dans le cadre de ce processus :

- Autorisez un espace d'adressage IP pour plus d'un VPC par région.
- Au sein d'un VPC, prévoyez de l'espace pour plusieurs sous-réseaux afin de pouvoir couvrir plusieurs zones de disponibilité.
- Pensez à laisser de l'espace de bloc CIDR non utilisé au sein d'un VPC pour une future expansion.
- Assurez-vous qu'il existe un espace d'adressage IP pour répondre aux besoins de tout parc transitoire d'instances Amazon EC2 que vous pourriez utiliser, comme les parcs d'instances Spot pour Machine Learning, les clusters Amazon EMR ou Amazon Redshift. Une attention similaire doit être accordée aux clusters Kubernetes, tels qu'Amazon Elastic Kubernetes Service (Amazon EKS), car chaque pod Kubernetes se voit attribuer une adresse routable depuis le bloc CIDR du VPC par défaut.

- Remarque : les quatre premières adresses IP et la dernière adresse IP dans le bloc CIDR de chaque sous-réseau sont réservées et ne peuvent pas être utilisées.
- Notez que le bloc CIDR initial du VPC alloué à votre VPC ne peut pas être modifié ou supprimé, mais vous pouvez ajouter des blocs CIDR non superposés au VPC. Les CIDR IPv4 de sous-réseau ne sont pas modifiables, mais les CIDR IPv6 le sont.
- Le plus grand bloc d'adresse CIDR VPC possible est un /16 et le plus petit est un /28.
- Envisagez d'autres réseaux connectés (VPC, sur site ou autres fournisseurs de cloud) et veillez à ce que l'espace d'adressage IP ne se chevauche pas. Pour plus d'informations, consultez REL02-BP05 Appliquer des plages d'adresses IP privées sans chevauchement dans tous les espaces d'adressage privés où elles sont connectées.

Résultat escompté : un sous-réseau IP évolutif peut vous aider à faire face à la croissance future et à éviter tout gaspillage inutile.

Anti-modèles courants :

- Ignorer la croissance future et utiliser des blocs CIDR trop petits, qui requièrent une reconfiguration, ce qui peut entraîner des temps d'arrêt.
- Estimation incorrecte du nombre d'adresses IP qu'un Elastic Load Balancer peut utiliser.
- Déploiement de nombreux équilibreurs de charge à trafic élevé dans les mêmes sous-réseaux
- Utilisation de mécanismes de dimensionnement automatisés sans surveiller la consommation d'adresses IP.
- La définition de plages CIDR excessivement étendues va bien au-delà des prévisions de croissance futures, ce qui peut entraîner des difficultés pour l'appairage avec d'autres réseaux dont les plages d'adresses se chevauchent.

Avantages de respecter cette bonne pratique : ces tailles sont la garantie que vous pouvez prendre en charge la croissance de vos charges de travail et continuer à fournir une disponibilité lors de l'augmentation verticale.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : moyen

Directives d'implémentation

Planifiez votre réseau en prévision de votre croissance, de la conformité réglementaire et de son intégration avec d'autres composants. La croissance peut être sous-estimée, la conformité

réglementaire peut changer, et les acquisitions ou les connexions à des réseaux privés peuvent être difficiles à implémenter sans une planification appropriée.

- Sélectionnez les régions et Comptes AWS pertinents en fonction de vos exigences de services, de la latence, des exigences réglementaires et de reprise après sinistre (DR).
- Identifiez vos besoins pour les déploiements VPC régionaux.
- Identifiez la taille des VPC.
  - Déterminez si vous allez déployer une connectivité multi-VPC.
    - Qu'est-ce qu'une passerelle de transit ?
    - Connectivité multi-VPC dans une seule région
  - Déterminez si vous avez besoin d'une mise en réseau séparée pour les exigences réglementaires.
  - Créez des VPC avec des blocs CIDR de taille appropriée pour répondre à vos besoins actuels et futurs.
    - Si vos prévisions de croissance sont inconnues, il peut être préférable d'opter pour des blocs CIDR plus importants afin de réduire le risque de reconfiguration future
  - Envisagez d'utiliser l'adressage IPv6 pour les sous-réseaux dans le cadre d'un VPC à double pile. IPv6 convient parfaitement à une utilisation dans des sous-réseaux privés contenant des flottes d'instances éphémères ou des conteneurs qui nécessiteraient autrement un grand nombre d'adresses IPv4.

Ressources

Bonnes pratiques Well-Architected connexes :

- REL02-BP05 Appliquer des plages d'adresses IP privées sans chevauchement dans tous les espaces d'adressage privés où elles sont connectées

Documents connexes :

- Partenaire APN : partenaires pouvant vous aider à planifier votre mise en réseau
- AWS Marketplace pour l'infrastructure réseau
- Livre blanc sur les options de connectivité d'Amazon Virtual Private Cloud
- Connectivité au réseau à haute disponibilité de plusieurs centres de données
- Connectivité multi-VPC dans une seule région

- Qu'est-ce qu'Amazon VPC ?
- IPv6 sur AWS
- IPv6 sur des architectures de référence
- Amazon Elastic Kubernetes Service lance le support IPv6
- Recommandations pour votre VPC – Classic Load Balancers
- Sous-réseaux de zone de disponibilité – Application Load Balancers
- Zones de disponibilité – Network Load Balancers

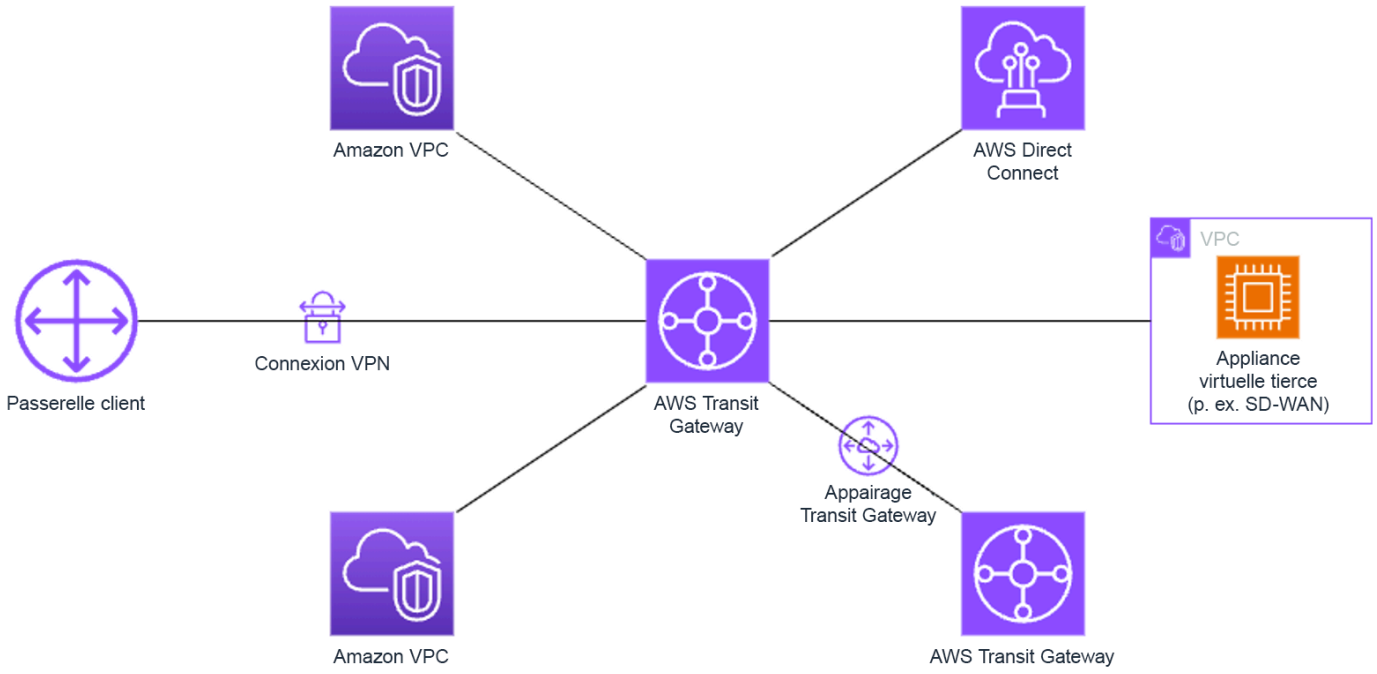
Vidéos connexes :

- AWS re:Invent 2018: Advanced VPC Design and New Capabilities for Amazon VPC (NET303)
- AWS re:Invent 2019: AWS Transit Gateway reference architectures for many VPCs (NET406-R1)
- AWS re:Invent 2023: AWS Ready for what's next? Designing networks for growth and flexibility (NET310)

REL02-BP04 Préférer les topologies en étoile au maillage « many-to-many »

Lorsque vous connectez plusieurs réseaux privés, tels que des clouds privés virtuels (VPC) et des réseaux sur site, optez pour une topologie en étoile plutôt qu'une topologie maillée. Contrairement aux topologies maillées, où chaque réseau se connecte directement aux autres et augmente la complexité et les frais de gestion, l'architecture en étoile centralise les connexions via un hub unique. Cette centralisation simplifie la structure du réseau et améliore son opérabilité, sa capacité de mise à l'échelle et son contrôle.

AWS Transit Gateway est un service géré, évolutif et hautement disponible conçu pour la construction de réseaux en étoile sur AWS. Il fait office de hub central de votre réseau et assure la segmentation du réseau, le routage centralisé et la connexion simplifiée aux environnements cloud et sur site. La figure suivante montre comment utiliser AWS Transit Gateway pour créer une topologie en étoile.



Résultat escompté : vous avez connecté vos clouds privés virtuels (VPC) et vos réseaux sur site via un hub central. Vous configurez vos connexions d'appairage via le hub, qui agit comme un routeur cloud hautement évolutif. Le routage est simplifié car vous n'avez pas à travailler avec des relations d'appairage complexes. Le trafic entre les réseaux est chiffré et vous avez la possibilité d'isoler les réseaux.

Anti-modèles courants :

- Vous établissez des règles d'appairage réseau complexes.
- Vous fournissez des routes entre des réseaux qui ne doivent pas communiquer entre eux (par exemple, des charges de travail distinctes qui n'ont aucune interdépendance).
- La gouvernance de l'instance du hub est inefficace.

Avantages du respect de cette bonne pratique : à mesure que le nombre de réseaux connectés augmente, la gestion et le développement de la connectivité maillée deviennent de plus en plus difficiles. Une architecture maillée introduit des défis supplémentaires, tels que des composants d'infrastructure, des exigences de configuration et des considérations de déploiement supplémentaires. Le maillage introduit également une surcharge supplémentaire pour gérer et surveiller les composants du plan de données et du plan de contrôle. Vous devez réfléchir

à la manière d'assurer la haute disponibilité de l'architecture maillée, de surveiller l'état et les performances du maillage et de gérer les mises à niveau des composants du maillage.

Un modèle en étoile (hub and spoke), quant à lui, établit un routage centralisé du trafic sur plusieurs réseaux. Il fournit une approche plus simple de la gestion et de la surveillance des composants du plan de données et du plan de contrôle.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : moyen

Directives d'implémentation

Créez un compte de services réseau s'il n'en existe pas. Placez le hub dans le compte de services réseau de l'organisation. Cette approche permet au hub d'être géré de manière centralisée par les ingénieurs réseau.

Le hub du modèle en étoile (hub and spoke) agit en tant que routeur virtuel pour le trafic circulant entre vos clouds privés virtuels (VPC) et les réseaux sur site. Cette approche réduit la complexité du réseau et facilite la résolution des problèmes de mise en réseau.

Tenez compte de la conception de votre réseau, notamment des VPC, d'AWS Direct Connect et des connexions VPN de site à site que vous souhaitez interconnecter.

Envisagez d'utiliser un sous-réseau distinct pour chaque attachement de VPC de passerelle de transit. Pour chaque sous-réseau, utilisez un petit CIDR (par exemple /28), afin d'avoir plus d'espace d'adressage pour les ressources de calcul. De plus, créez une liste ACL réseau et associez-la à tous les sous-réseaux qui sont associés au hub. Gardez la liste ACL réseau ouverte dans les directions entrantes et sortantes.

Concevez et implémentez vos tables de routage de telle sorte que les routes ne soient fournies qu'entre les réseaux qui doivent communiquer. Omettez les routes entre des réseaux qui ne doivent pas communiquer entre eux (par exemple, entre des charges de travail distinctes qui n'ont aucune interdépendance).

Étapes d'implémentation

1. Planifiez votre réseau. Déterminez les réseaux que vous souhaitez connecter et vérifiez qu'ils ne partagent pas de plages CIDR superposées.
2. Créez une passerelle AWS Transit Gateway et attachez-lui vos VPC.
3. Si nécessaire, créez des connexions VPN ou des passerelles Direct Connect et associez-les à la passerelle Transit Gateway.

4. Définissez la façon dont le trafic est acheminé entre les VPC connectés et les autres connexions via la configuration de vos tables de routage Transit Gateway.
5. Utilisez Amazon CloudWatch pour surveiller et ajuster les configurations selon les besoins afin d'optimiser les performances et les coûts.

Ressources

Bonnes pratiques associées :

- REL02-BP03 S'assurer que l'allocation des sous-réseaux IP tient compte de l'expansion et de la disponibilité
- REL02-BP05 Appliquer des plages d'adresses IP privées sans chevauchement dans tous les espaces d'adressage privés où elles sont connectées

Documents connexes :

- Qu'est-ce qu'une passerelle de transit?
- Bonnes pratiques pour la conception de passerelles de transit
- Création d'une infrastructure réseau AWS multi-VPC évolutive et sécurisée
- Création d'un réseau global à l'aide de AWS Transit Gateway Inter-Region peering
- Amazon Virtual Private Cloud Connectivity Options
- Partenaire APN : partenaires pouvant vous aider à planifier votre mise en réseau
- AWS Marketplace pour l'infrastructure réseau

Vidéos connexes:

- AWS re:Invent 2023 - AWS networking foundations
- AWS re:Invent 2023 - Advanced VPC designs and new capabilities

Ateliers connexes :

- Atelier AWS Transit Gateway

REL02-BP05 Appliquer des plages d’adresses IP privées sans chevauchement dans tous les espaces d’adressage privés où elles sont connectées

Les plages d’adresses IP de vos VPC ne doivent pas se chevaucher lorsque les réseaux sont appairés, connectés via Transit Gateway ou connectés via un VPN. Évitez les conflits d’adresses IP entre un VPC et des environnements sur site ou avec d’autres fournisseurs de services cloud que vous utilisez. Vous devez également disposer d’un moyen d’allouer des plages d’adresses IP privées lorsque cela est nécessaire. Un système de gestion des adresses IP (IPAM) peut aider à automatiser cette opération.

Résultat escompté :

- aucun conflit de plage d’adresses IP entre les VPC, les environnements sur site ou d’autres fournisseurs de services cloud.
- Une bonne gestion des adresses IP permet d’adapter plus facilement l’infrastructure réseau à la croissance et à l’évolution des exigences en matière de réseau.

Anti-modèles courants :

- Utilisation dans votre VPC d’une plage d’adresses IP identique à celle que vous utilisez sur site, dans votre réseau d’entreprise ou auprès d’autres fournisseurs de services cloud
- Non-suivi des plages d’adresses IP des VPC utilisés pour déployer vos charges de travail.
- Utilisation de processus manuels de gestion des adresses IP, tels que des feuilles de calcul.
- Surdimensionnement ou sous-dimensionnement des blocs CIDR, entraînant un gaspillage d’adresses IP ou un espace d’adressage insuffisant pour votre charge de travail.

Avantages du respect de cette bonne pratique : la planification active de votre réseau garantit que vous n’avez pas plusieurs occurrences de la même adresse IP dans les réseaux interconnectés. Cela empêche les problèmes de routage de se produire dans certaines parties de la charge de travail qui utilisent les différentes applications.

Niveau d’exposition au risque si cette bonne pratique n’est pas respectée : moyen

Directives d'implémentation

Utilisez un IPAM, tel que le gestionnaire d'adresses IP Amazon VPC, pour surveiller et gérer votre utilisation du CIDR. Plusieurs gestionnaires IPAM sont également disponibles sur AWS Marketplace. Évaluez votre utilisation potentielle sur AWS, ajoutez des plages CIDR à des VPC existants et créez des VPC pour autoriser une croissance d'utilisation planifiée.

Étapes d'implémentation

- Capturez votre consommation CIDR actuelle (par exemple, VPC et sous-réseaux).
  - Utilisez des opérations d'API de service pour collecter la consommation CIDR actuelle.
  - Utilisez le gestionnaire d'adresses IP Amazon VPC pour découvrir des ressources.
- Capturez l'utilisation actuelle de votre sous-réseau.
  - Utilisez des opérations d'API de service pour collecter les sous-réseaux par VPC dans chaque région.
  - Utilisez le gestionnaire d'adresses IP Amazon VPC pour découvrir des ressources.
- Enregistrez l'utilisation actuelle.
- Déterminez si vous avez créé des plages d'adresses IP se chevauchant.
- Calculez la capacité inutilisée.
- Identifiez les plages d'adresses IP qui se chevauchent. Vous pouvez soit migrer vers une nouvelle plage d'adresses, soit envisager d'utiliser des techniques telles que la passerelle NAT privée ou AWS PrivateLink si vous devez connecter les plages qui se chevauchent.

Ressources

Bonnes pratiques associées :

- Protection des réseaux

Documents connexes :

- Partenaire APN : partenaires pouvant vous aider à planifier votre mise en réseau
- AWS Marketplace pour l'infrastructure réseau
- Livre blanc sur les options de connectivité d'Amazon Virtual Private Cloud
- Connectivité au réseau à haute disponibilité de plusieurs centres de données

- Connexion de réseaux dont les plages d’adresses IP se chevauchent
- Qu’est-ce qu’Amazon VPC ?
- Qu’est-ce qu’IPAM ?

Vidéos connexes :

- AWS re:Invent 2023 - Advanced VPC designs and new capabilities
- AWS re:Invent 2019: AWS Transit Gateway reference architectures for many VPCs
- AWS re:Invent 2023 - Ready for what’s next? Designing networks for growth and flexibility
- AWS re:Invent 2021 - {New Launch} Manage your IP addresses at scale on AWS

Architecture de charge de travail

Pour garantir la fiabilité d'une charge de travail, il faut commencer par choisir le bon logiciel et la bonne infrastructure. Vos choix d'architecture ont un impact sur le comportement des charges de travail sur les cinq piliers Well-Architected. Pour des raisons de fiabilité, vous devez suivre des modèles spécifiques.

Les sections suivantes expliquent les bonnes pratiques à utiliser avec ces modèles de fiabilité.

Rubriques

Conception de l'architecture de votre service de charge de travail

Créez des charges de travail hautement évolutives et fiables à l'aide d'une architecture orientée service (SOA) ou d'une architecture de microservices. L'architecture orientée services (SOA) consiste à rendre les composants logiciels réutilisables via les interfaces de service. L'architecture des microservices va plus loin, en particulier en rendant les composants plus petits et plus simples.

Les interfaces d'architecture orientée service (SOA) utilisent des normes de communication communes permettant leur intégration rapide à de nouvelles charges de travail. La SOA a remplacé la pratique consistant à créer des architectures monolithes, composées d'unités interdépendantes et indivisibles.

Chez AWS, nous avons toujours utilisé la SOA. Nous concevons toutefois désormais nos systèmes à l'aide de microservices. Bien que les micro-services présentent plusieurs qualités attractives, l'avantage le plus important pour la disponibilité est le fait que les microservices sont plus légers et plus simples. Ils vous permettent de différencier la disponibilité requise par les différents services, et ainsi de concentrer plus spécifiquement vos investissements sur les microservices qui présentent les besoins les plus importants en disponibilité. Par exemple, pour fournir des pages d'informations produits sur Amazon.com (« pages de détails »), des centaines de microservices sont appelés pour construire des parties distinctes de la page. Tandis que certains services doivent être disponibles pour fournir le prix et les détails des produits, la grande majorité du contenu de la page peut simplement être exclu si le service n'est pas disponible. Même des éléments tels que les photos et

commentaires ne sont pas requis pour fournir une expérience permettant à un client d'acheter un produit.

Bonnes pratiques

- REL03-BP01 Choisissez comment segmenter votre charge de travail
- REL03-BP02 Créer des services axés sur des domaines d'activité et la fonctionnalité
- REL03-BP03 Fournir des contrats de service par API

REL03-BP01 Choisissez comment segmenter votre charge de travail

La segmentation de la charge de travail est importante lorsqu'il s'agit de déterminer les exigences de résilience de votre application. L'architecture monolithique doit être évitée dans la mesure du possible. À la place, réfléchissez bien aux composants de l'application capables d'être divisés en microservices. Selon les exigences de votre application, il peut s'agir d'une combinaison d'une architecture orientée services (SOA) avec des microservices dans la mesure du possible. Les charges de travail capables d'absence d'état sont davantage en mesure d'être déployées en tant que microservices.

Résultat désiré : les charges de travail doivent être supportables, évolutives et aussi faiblement couplées que possible.

Lorsque vous choisissez comment segmenter votre charge de travail, comparez les avantages aux complexités. Ce qui convient pour un nouveau produit en course pour un premier lancement est différent de ce dont a besoin une charge de travail conçue pour augmenter d'échelle. Lors de la refactorisation d'une architecture monolithique existante, vous devez évaluer comment l'application prendra en charge une décomposition vers l'absence d'état. La division de services en microservices permet aux petites équipes bien définies de les développer et les gérer. Toutefois, les services plus petits peuvent créer des complexités, dont une latence supérieure, un débogage plus complexe et une charge opérationnelle accrue.

Anti-modèles courants :

- Le microservice Death Star est une situation dans laquelle les composants atomiques deviennent si interdépendants que l'échec de l'un d'entre eux résulte en un échec encore plus important, ce qui rend les composants aussi rigides et fragiles qu'une architecture monolithique.

Avantages liés au respect de cette pratique :

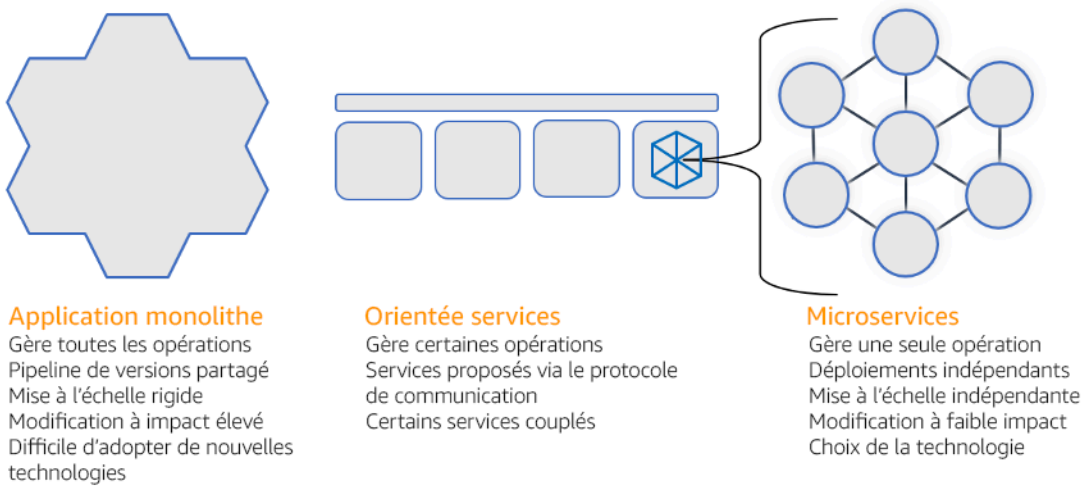
- L’utilisation de segments plus petits permet une plus grande agilité, une plus grande flexibilité organisationnelle et une capacité de mise à l’échelle.
- L’impact réduit des interruptions de service.
- Les composants de l’application peuvent avoir différentes exigences de disponibilité, pouvant être pris en charge par une segmentation plus atomique.
- Des responsabilités bien définies pour les équipes prenant en charge la charge de travail.

Niveau d’exposition au risque si cette bonne pratique n’est pas respectée : élevé

Directives d’implémentation

Choisissez votre type d’architecture en fonction de la façon dont vous segmenterez votre charge de travail. Choisissez une architecture SOA ou une architecture de microservices (ou dans de rares cas, une architecture monolithique). Même si vous choisissez de commencer par une architecture monolithe, vous devez vous assurer qu'elle est modulaire et qu'elle puisse finalement évoluer vers des microservices au fur et à SOA mesure que votre produit évolue avec l'adoption par les utilisateurs. SOAet les microservices offrent respectivement une segmentation plus petite, ce qui est préférable en tant qu'architecture moderne, évolutive et fiable, mais certains compromis doivent être pris en compte, en particulier lors du déploiement d'une architecture de microservices.

Le principal compromis est que vous avez maintenant une architecture pour le calcul distribué qui peut compliquer le respect des exigences en matière de latence des utilisateurs et qui complexifie le suivi et le débogage des interactions des utilisateurs. AWS X-Ray peut vous aider à résoudre ce problème. Un autre effet à prendre en compte est la hausse de la complexité opérationnelle à mesure que vous augmentez le nombre d’applications que vous gérez, ce qui nécessite le déploiement de plusieurs composants indépendants.



Architectures monolithique, orientée services et de microservices

Étapes d'implémentation

- Déterminer l'architecture adaptée pour refactoriser ou créer votre application. SOAet les microservices offrent respectivement une segmentation plus petite, ce qui est préférable en tant qu'architecture moderne, évolutive et fiable. SOApeut être un bon compromis pour réduire la segmentation tout en évitant certaines des complexités des microservices. Pour plus de détails, consultez la section Compromis des microservices.
- Si votre charge de travail est appropriée et que votre organisation peut la prendre en charge, vous devez utiliser une architecture de microservices pour obtenir la meilleure agilité et la meilleure fiabilité. Pour plus de détails, voir Implémentation de microservices sur AWS.
- Envisagez de suivre le modèle Strangler Fig pour refactoriser un monolithe en composants plus petits. Cela implique le remplacement progressif de composants spécifiques de l'application par de nouvelles applications et de nouveaux services. AWS Migration Hub Refactor Spaces sert de point de départ à la refactorisation incrémentielle. Pour plus de détails, consulter Migrer sans interruption vers des charges de travail existantes sur site à l'aide d'un modèle Figuier étrangleur.
- La mise en œuvre de microservices peut nécessiter un mécanisme de découverte de services pour permettre à ces services distribués de communiquer entre eux. AWS App Meshpeut être utilisé avec des architectures orientées services pour permettre une découverte et un accès fiables aux services. AWS Cloud Mappeut également être utilisé pour la découverte de services dynamique DNS basée sur des données.
- Si vous passez d'un monolithe à un bus de service, Amazon SOA MQ peut vous aider à combler cette lacune lors de la refonte d'applications existantes dans le cloud.

- Pour les architectures monolithiques existantes avec une base de données partagée unique, choisissez comment réorganiser les données en segments plus petits. Vous pouvez les réorganiser par unité commerciale, modèle d'accès ou structure de données. À ce stade du processus de refactorisation, vous devez choisir de passer à un type de base de données relationnel ou non relationnel (nonSQL). Pour plus de détails, voir De SQL à Non SQL.

Niveau d'effort du plan d'implémentation : élevé

Ressources

Bonnes pratiques associées :

- REL03-BP02 Créer des services axés sur des domaines d'activité et la fonctionnalité

Documents connexes :

- Amazon API Gateway : Configuration d'une application à REST API l'aide d'Open API
- Qu'est-ce que l'architecture orientée service ?
- Contexte délimité (modèle central dans la conception pilotée par domaine)
- Implémentation de microservices sur AWS
- Compromis des microservices
- Microservices : une définition de ce nouveau terme architectural
- Microservices sur AWS
- Qu'est-ce que c'est AWS App Mesh ?

Exemples connexes :

- Atelier sur la modernisation itérative des applications

Vidéos connexes :

- Garantir l'excellence grâce aux microservices sur AWS

REL03-BP02 Créer des services axés sur des domaines d'activité et la fonctionnalité

Une architecture orientée services (SOA) définit des services avec des fonctions bien déterminées dictées par les besoins métier. Les microservices utilisent des modèles de domaine et un contexte limité pour définir les limites des services en fonction des limites du contexte métier. En se concentrant sur les domaines d'activité et les fonctionnalités, les équipes peuvent définir des exigences de fiabilité indépendantes pour leurs services. Les contextes limités isolent et encapsulent la logique métier, ce qui permet aux équipes de mieux raisonner sur la manière de gérer les défaillances.

Résultat escompté : les ingénieurs et les parties prenantes de l'entreprise définissent conjointement des contextes délimités et les utilisent pour concevoir des systèmes en tant que services remplissant des fonctions commerciales spécifiques. Ces équipes utilisent des pratiques établies telles que l'event storming pour définir les exigences. Les nouvelles applications sont conçues comme des services dont les limites sont bien définies et qui possèdent un couplage faible. Les monolithes existants sont décomposés en contextes délimités et les conceptions des systèmes évoluent vers des architectures SOA ou de microservices. Lorsque les monolithes sont refactorisés, des approches établies telles que les contextes de bulles et les modèles de décomposition des monolithes sont appliquées.

Les services orientés domaine sont exécutés sous la forme d'un ou de plusieurs processus qui ne partagent pas d'état. Ils répondent de manière indépendante aux fluctuations de la demande et gèrent les scénarios de panne à la lumière des exigences spécifiques du domaine.

Anti-modèles courants :

- Les équipes sont constituées autour de domaines techniques spécifiques tels que l'interface utilisateur et l'expérience utilisateur, les intergiciels ou les bases de données plutôt que de domaines commerciaux spécifiques.
- Les applications couvrent des responsabilités de domaine. Les services qui couvrent des contextes limités peuvent être plus difficiles à gérer, nécessiter des efforts de test plus importants et nécessiter la participation de plusieurs équipes de domaine aux mises à jour logicielles.
- Les dépendances de domaine, telles que les bibliothèques d'entités de domaine, sont partagées entre les services de telle sorte que les modifications apportées à un domaine de service nécessitent des modifications apportées à d'autres domaines de service.

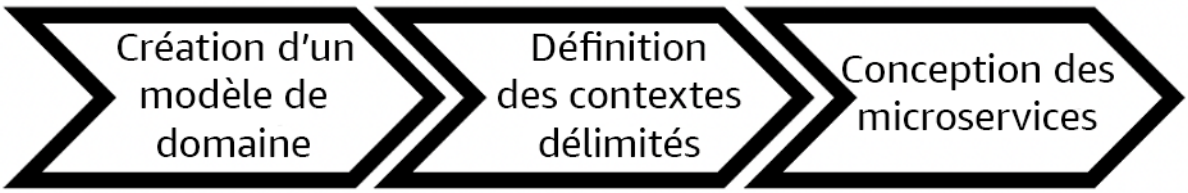
- Les contrats de service et la logique métier n'expriment pas les entités dans un langage de domaine commun et cohérent, ce qui crée des couches de traduction qui compliquent les systèmes et augmentent les efforts de débogage.

Avantages du respect de cette bonne pratique : les applications sont conçues comme des services indépendants délimités par domaines d'activité et utilisent un langage métier commun. Les services peuvent être testés et déployés indépendamment. Les services répondent aux exigences de résilience spécifiques au domaine mis en œuvre.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : élevé

Directives d'implémentation

La conception pilotée par domaine (DDD) est l'approche fondamentale de la conception et de la création de logiciels autour de domaines métier. Il est utile de travailler avec un cadre existant lorsque vous créez des services axés sur des domaines métier. Lorsque vous travaillez avec des applications monolithiques existantes, vous pouvez tirer parti des modèles de décomposition qui fournissent des techniques éprouvées pour moderniser les applications en services.



Conception pilotée par domaine

Étapes d'implémentation

- Les équipes peuvent organiser des ateliers event storming pour identifier rapidement les événements, les commandes, les agrégats et les domaines dans un format léger de notes autocollantes.
- Une fois que les entités et les fonctions de domaine ont été formées dans un contexte de domaine, vous pouvez diviser votre domaine en services à l'aide d'un contexte délimité, dans lequel les entités partageant des caractéristiques et des attributs similaires sont regroupées. La division en contextes permet de faire émerger un modèle de délimitation des microservices.
  - Par exemple, les entités du site Web Amazon.com peuvent inclure le colis, la livraison, le calendrier, le prix, la remise et la devise.

    - Le colis, la livraison et le calendrier sont regroupés dans le contexte d'expédition, tandis que le prix, la remise et la devise sont regroupés dans le contexte de tarification.
- Décomposition des monolithes en microservices décrit les modèles de refactorisation des microservices. L'utilisation de modèles de décomposition par capacité métier, sous-domaine ou transaction s'inscrit parfaitement dans les approches axées sur le domaine.
- Les techniques tactiques telles que le contexte à bulles vous permettent d'introduire le DDD dans des applications existantes ou héritées sans devoir procéder à des réécritures préalables et sans engagement total envers le DDD. Dans une approche basée sur un contexte à bulles, un petit contexte délimité est établi à l'aide d'une cartographie et d'une coordination des services, ou couche anticorruption, qui protège le modèle de domaine nouvellement défini des influences extérieures.

Une fois que les équipes ont effectué une analyse du domaine et défini des entités et des contrats de service, elles peuvent tirer parti des services AWS pour mettre en œuvre leur conception axée sur le domaine sous forme de services basés sur le cloud.

- Commencez votre développement en définissant des tests qui appliquent les règles métier de votre domaine. Le développement piloté par les tests (TDD) et le développement piloté par le comportement (BDD) aident les équipes à concentrer leurs services sur la résolution des problèmes commerciaux.
- Sélectionnez les services AWS qui répondent le mieux aux exigences de votre domaine d'activité et à votre architecture de microservices :
    - AWS sans serveur permet à votre équipe de se concentrer sur une logique de domaine spécifique au lieu de gérer les serveurs et l'infrastructure.
    - Les conteneurs à AWS simplifient la gestion de votre infrastructure afin que vous puissiez vous concentrer sur les exigences de votre domaine.
    - Les bases de données sur mesure vous permettent d'adapter les exigences de votre domaine au type de base de données le mieux adapté.
- Création d'architectures hexagonales sur AWS décrit un cadre permettant d'intégrer une logique métier à des services en procédant de manière rétroactive à partir d'un domaine métier afin de répondre à des exigences fonctionnelles, puis d'associer des adaptateurs d'intégration. Les modèles qui séparent les détails de l'interface de la logique métier avec les services AWS aident les équipes à se concentrer sur les fonctionnalités du domaine et à améliorer la qualité des logiciels.

Ressources

Bonnes pratiques associées:

- REL03-BP01 Choisissez comment segmenter votre charge de travail
- REL03-BP03 Fournir des contrats de service par API

Documents connexes :

- Microservices AWS
- Implémentation des microservices sur AWS
- Comment convertir un monolithe en microservices
- Premiers pas avec DDD en présence de systèmes hérités
- Conception axée sur le domaine : aborder la complexité au cœur du logiciel
- Création d’architectures hexagonales sur AWS
- Décomposition des monolithes en microservices
- Event Storming
- Messages entre contextes limités
- Microservices
- Développement piloté par les tests
- Développement axé sur le comportement

Exemples connexes :

- Conception de microservices natifs cloud sur AWS (extrait de DDD/EventStormingWorkshop)

Outils associés :

- AWS CloudBases de données
- Sans serveur activé sur AWS
- Conteneurs à AWS

REL03-BP03 Fournir des contrats de service par API

Les contrats de service sont des accords documentés entre API producteurs et consommateurs définis dans une définition lisible par API machine. Une stratégie de gestion des versions des contrats permet aux consommateurs de continuer à utiliser l'existant API et de migrer leurs applications vers une version plus récente API lorsqu'elles sont prêtes. Le déploiement du producteur peut avoir lieu à tout moment tant que le contrat est respecté. Les équipes de service peuvent utiliser la pile technologique de leur choix pour exécuter le API contrat.

Résultat escompté : les applications conçues avec des architectures orientées services ou microservices sont capables de fonctionner de manière indépendante tout en intégrant une dépendance à l'environnement d'exécution. Les modifications apportées à un API consommateur ou à un producteur n'interrompent pas la stabilité de l'ensemble du système lorsque les deux parties suivent un API contrat commun. Les composants qui communiquent via le service APIs peuvent exécuter des versions fonctionnelles indépendantes, mettre à niveau les dépendances d'exécution ou basculer vers un site de reprise après sinistre (DR) avec peu ou pas d'impact les uns sur les autres. En outre, les services discrets sont capables d’évoluer de manière indépendante en absorbant la demande de ressources sans que les autres services soient réduits horizontalement à l’unisson.

Anti-modèles courants :

- Création d'un service APIs sans schémas fortement typés. Il en résulte APIs que cela ne peut pas être utilisé pour générer des API liaisons et des charges utiles qui ne peuvent pas être validées par programmation.
- Ne pas adopter de stratégie de gestion des versions, qui oblige API les consommateurs à effectuer des mises à jour et à publier ou à échouer lorsque les contrats de service évoluent.
- Messages d’erreur qui divulguent les détails de l’implémentation du service sous-jacent au lieu de décrire les échecs d’intégration dans le contexte et le langage du domaine.
- Ne pas utiliser de API contrats pour développer des scénarios de test et API des mises en œuvre fictives afin de permettre des tests indépendants des composants du service.

Avantages de l'établissement de cette meilleure pratique : les systèmes distribués composés de composants communiquant par le biais de contrats de API service peuvent améliorer la fiabilité. Les développeurs peuvent détecter les problèmes potentiels dès le début du processus de développement en vérifiant le type lors de la compilation afin de vérifier que les demandes et les réponses respectent le API contrat et que les champs obligatoires sont présents. APIles contrats

fournissent une interface autodocumentée claire APIs et fournissent une meilleure interopérabilité entre les différents systèmes et langages de programmation.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : moyen

Directives d'implémentation

Une fois que vous avez identifié les domaines commerciaux et déterminé la segmentation de votre charge de travail, vous pouvez développer votre serviceAPIs. Définissez d'abord des contrats de service lisibles par machine pourAPIs, puis implémentez une stratégie de gestion des API versions. Lorsque vous êtes prêt à intégrer des services via des protocoles courants tels que REST GraphQL ou des événements asynchrones, vous pouvez intégrer des AWS services dans votre architecture afin d'intégrer vos composants à l'aide de contrats bien typés. API

AWS services pour les API contrats de service

Intégrez AWS des services tels qu'Amazon API Gateway et Amazon EventBridge dans votre architecture pour utiliser des contrats de API service dans votre application. AWS AppSync Amazon API Gateway vous permet d'intégrer directement des AWS services natifs et d'autres services Web. APIGateway prend en charge la APIspécification Open et le versionnement. AWS AppSync est un point de terminaison GraphQL géré que vous configurez en définissant un schéma GraphQL pour définir une interface de service pour les requêtes, les mutations et les abonnements. Amazon EventBridge utilise des schémas d'événements pour définir des événements et générer des liaisons de code pour vos événements.

Étapes d'implémentation

- Tout d'abord, définissez un contrat pour votreAPI. Un contrat exprimera les capacités d'un API et définira des objets de données et des champs fortement typés pour l'APIentrée et la sortie.
- Lorsque vous configurez APIs dans API Gateway, vous pouvez importer et exporter des API spécifications ouvertes pour vos points de terminaison.
  - L'importation d'une API définition ouverte simplifie la création de votre API et peut être intégrée à AWS l'infrastructure sous forme d'outils de code tels que le AWS Serverless Application Modelet AWS Cloud Development Kit (AWS CDK).
  - L'exportation d'une API définition simplifie l'intégration avec API les outils de test et fournit aux consommateurs de services une spécification d'intégration.

- Vous pouvez définir et gérer GraphQL AWS AppSync en définissant APIs un fichier de schéma GraphQL pour générer votre interface de contrat et simplifier l'interaction avec des REST modèles complexes, plusieurs tables de base de données ou des services existants.
- AWS Amplifyles projets intégrés AWS AppSync génèrent des fichiers de JavaScript requêtes fortement typés à utiliser dans votre application, ainsi qu'une bibliothèque cliente AWS AppSync GraphQL pour les tables Amazon DynamoDB.
- Lorsque vous consommez des événements de service d'Amazon EventBridge, les événements adhèrent aux schémas qui existent déjà dans le registre des schémas ou que vous définissez avec l'Open API Spec. Avec un schéma défini dans le registre, vous pouvez également générer des liaisons client à partir du contrat de schéma afin d'intégrer votre code aux événements.
- Extension ou version de votreAPI. L'extension d'une API est une option plus simple lorsque vous ajoutez des champs qui peuvent être configurés avec des champs facultatifs ou des valeurs par défaut pour les champs obligatoires.
  - JSONles contrats basés sur des protocoles tels que REST GraphQL peuvent être une bonne solution pour l'extension de contrat.
  - XMLdes contrats basés sur des protocoles tels que SOAP ceux qui devraient être testés auprès des consommateurs de services afin de déterminer la faisabilité d'une prolongation du contrat.
- Lors du versionnement d'unAPI, envisagez d'implémenter le versionnage par proxy dans le cadre duquel une façade est utilisée pour prendre en charge les versions afin que la logique puisse être maintenue dans une base de code unique.
  - Avec API Gateway, vous pouvez utiliser les mappages de demandes et de réponses pour simplifier l'absorption des modifications de contrat en établissant une façade pour fournir des valeurs par défaut pour les nouveaux champs ou pour supprimer les champs supprimés d'une demande ou d'une réponse. Avec cette approche, le service sous-jacent peut gérer une base de code unique.

Ressources

Bonnes pratiques associées :

- REL03-BP01 Choisissez comment segmenter votre charge de travail
- REL03-BP02 Créer des services axés sur des domaines d'activité et la fonctionnalité
- REL04-BP02 Implémenter des dépendances faiblement couplées
- REL05-BP03 Contrôler et limiter les appels de nouvelle tentative
- REL05-BP05 Définir les délais d'expiration des clients

Documents connexes :

- Qu'est-ce qu'une API (interface de programmation d'applications) ?
- Implémentation de microservices sur AWS
- Compromis des microservices
- Microservices : une définition de ce nouveau terme architectural
- Microservices sur AWS
- Utilisation des extensions de API passerelle pour ouvrir API
- Spécification ouverte API
- GraphQL : schémas et types
- liaisons EventBridge de code Amazon

Exemples connexes :

- Amazon API Gateway : Configuration d'une application à REST API l'aide d'Open API
- Amazon API Gateway vers l'application Amazon CRUD DynamoDB à l'aide d'Open API
- Modèles modernes d'intégration des applications à l'ère du sans serveur : intégration des services de API passerelle
- Implémentation du versionnement des API passerelles basé sur les en-têtes avec Amazon CloudFront
- AWS AppSync : création d’une application client

Vidéos connexes :

- Utilisation d'Open API in AWS SAM pour gérer API Gateway

Outils associés :

- APIPasserelle Amazon
- AWS AppSync
- Amazon EventBridge

Concevoir des interactions dans un système distribué pour éviter les défaillances

Les systèmes distribués s'appuient sur des réseaux de communication pour interconnecter les composants, comme les serveurs ou les services. Votre charge de travail doit fonctionner de manière fiable malgré la perte de données ou la latence sur ces réseaux. Les composants du système distribué doivent fonctionner de manière à ne pas avoir d'impact négatif sur les autres composants ou sur la charge de travail. Ces bonnes pratiques permettent d'éviter les défaillances et d'améliorer le temps moyen entre défaillances (MTBF).

Bonnes pratiques

- REL04-BP01 Identifier le type de systèmes distribués dont vous dépendez
- REL04-BP02 Implémenter des dépendances faiblement couplées
- REL04-BP03 Faire un travail constant
- REL04-BP04 Rendre les opérations de mutation idempotentes

REL04-BP01 Identifier le type de systèmes distribués dont vous dépendez

Les systèmes distribués peuvent être synchrones, asynchrones ou par lots. Les systèmes synchrones doivent traiter les demandes le plus rapidement possible et communiquer entre eux en effectuant des appels de demande et de réponse synchrones à l'aide des protocoles HTTP/S, REST ou RPC (Remote Procedure Call). Les systèmes asynchrones communiquent entre eux en échangeant des données de manière asynchrone via un service intermédiaire sans coupler des systèmes individuels. Les systèmes par lots reçoivent un volume important de données d'entrée, exécutent des processus de données automatisés sans intervention humaine et génèrent des données de sortie.

Résultat escompté : concevez une charge de travail qui interagit efficacement avec les dépendances synchrones, asynchrones et par lots.

Anti-modèles courants :

- La charge de travail attend indéfiniment une réponse de la part de ses dépendances, ce qui peut entraîner une expiration des clients de la charge de travail, qui ne savent pas si leur demande a été reçue.

- La charge de travail utilise une chaîne de systèmes dépendants qui s'appellent les uns les autres de manière synchrone. Cela nécessite que chaque système soit disponible et traite correctement une demande avant que l'ensemble de la chaîne puisse aboutir, ce qui entraîne un comportement et une disponibilité globale potentiellement fragiles.
- La charge de travail communique avec ses dépendances de manière asynchrone et repose sur le concept de livraison garantie unique des messages, alors qu'il est souvent encore possible de recevoir des messages dupliqués.
- La charge de travail n'utilise pas les outils de planification par lots appropriés et permet l'exécution simultanée de la même tâche de traitement par lots.

Avantages du respect de cette bonne pratique : il est courant qu'une charge de travail donnée implémente un ou plusieurs styles de communication entre synchrone, asynchrone et par lots. Cette bonne pratique vous aide à identifier les différents compromis associés à chaque style de communication afin que votre charge de travail soit capable de tolérer les interruptions liées à toutes ses dépendances.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : élevé

Directives d'implémentation

Les sections suivantes contiennent des instructions de mise en œuvre générales et spécifiques pour chaque type de dépendance.

General guidance

- Assurez-vous que les objectifs de niveau de service (SLO) offerts par vos dépendances en matière de performance et de fiabilité répondent aux exigences de performance et de fiabilité de votre charge de travail.
- Utilisez les services d'observabilité AWS pour surveiller les temps de réponse et les taux d'erreur afin de vous assurer que votre dépendance fournit des services aux niveaux requis par votre charge de travail.
- Identifiez les défis potentiels auxquels votre charge de travail peut être confrontée lors de la communication avec ses dépendances. Les systèmes distribués présentent un large éventail de défis susceptibles d'accroître la complexité architecturale, la charge opérationnelle et les coûts. Les défis courants incluent la latence, les perturbations du réseau, la perte de données, la mise à l'échelle et le retard de réplication des données.

- Mettez en œuvre une gestion des erreurs et une journalisation robustes pour vous aider à résoudre les problèmes lorsque votre dépendance rencontre des problèmes.

Dépendance synchrone

Dans les communications synchrones, votre charge de travail envoie une demande à sa dépendance et bloque l'opération en attente de réponse. Lorsque sa dépendance reçoit la demande, elle essaie de la traiter le plus rapidement possible et renvoie une réponse à la charge de travail. L'un des principaux défis liés à la communication synchrone est qu'elle entraîne un couplage temporel, ce qui nécessite que la charge de travail et ses dépendances soient disponibles en même temps. Lorsque la charge de travail doit communiquer de manière synchrone avec ses dépendances, suivez les conseils ci-dessous :

- Votre charge de travail ne doit pas reposer sur plusieurs dépendances synchrones pour exécuter une seule fonction. Cette chaîne de dépendances augmente la fragilité globale, car toutes les dépendances sur le chemin doivent être disponibles pour que la demande soit traitée correctement.
- Lorsqu'une dépendance n'est pas saine ou n'est pas disponible, déterminez vos stratégies de gestion des erreurs et de nouvelles tentatives. Évitez d'utiliser un comportement bimodal. On parle de comportement bimodal lorsque la charge de travail présente un comportement différent en mode normal et en mode d'échec. Pour plus de détails sur le comportement bimodal, voir REL11-BP05 Utiliser la stabilité statique pour empêcher le comportement bimodal.
- N'oubliez pas qu'il vaut mieux échouer rapidement que faire attendre la charge de travail. Par exemple, le Guide du développeur AWS Lambda explique comment gérer les tentatives et les échecs lorsque vous invoquez des fonctions Lambda.
- Définissez des délais d'expiration lorsque la charge de travail appelle sa dépendance. Cette technique permet d'éviter d'attendre trop longtemps ou d'attendre indéfiniment une réponse. Vous trouverez une discussion utile sur ce sujet dans la section Réglage des paramètres de demande HTTP du SDK Java AWS pour les applications Amazon DynamoDB sensibles à la latence.
- Réduisez le nombre d'appels passés entre la charge de travail et sa dépendance pour répondre à une seule demande. Le fait d'avoir trop d'appels entre elles augmente le couplage et la latence.

Dépendance asynchrone

Pour pouvoir découpler temporellement la charge de travail de sa dépendance, elles doivent communiquer de manière asynchrone. En utilisant une approche asynchrone, la charge de travail

peut poursuivre tout autre traitement sans avoir à attendre que sa dépendance, ou sa chaîne de dépendances, envoie une réponse.

Lorsque la charge de travail doit communiquer de manière asynchrone avec sa dépendance, suivez les conseils ci-dessous :

- Déterminez s'il convient d'utiliser la messagerie ou le streaming d'événements en fonction de votre cas d'utilisation et de vos exigences. La messagerie permet à votre charge de travail de communiquer avec ses dépendances en envoyant et en recevant des messages par le biais d'un agent de messages. Le streaming d'événements permet à votre charge de travail et à ses dépendances d'utiliser un service de streaming pour publier et s'abonner à des événements, diffusés sous forme de flux de données continus, qui doivent être traités dès que possible.

- La messagerie et le streaming d'événements traitent les messages différemment. Vous devez donc faire un choix en fonction des éléments suivants :
  - Priorité des messages : les agents de messages peuvent traiter les messages prioritaires avant les messages normaux. Dans le cadre du streaming d'événements, tous les messages ont la même priorité.
  - Consommation de messages : les agents de messages veillent à ce que les consommateurs reçoivent le message. Les consommateurs qui diffusent des événements doivent suivre le dernier message qu'ils ont lu.
  - Ordre des messages : avec la messagerie, la réception des messages dans l'ordre exact dans lequel ils sont envoyés n'est pas garantie, sauf si vous utilisez une approche « premier entré, premier sorti » (FIFO). Le streaming d'événements préserve toujours l'ordre dans lequel les données ont été produites.
  - Suppression du message : dans le cas de la messagerie, le consommateur doit supprimer le message après l'avoir traité. Le service de streaming d'événements ajoute le message à un flux et y reste jusqu'à l'expiration de la période de conservation du message. Cette politique de suppression rend le streaming d'événements adapté à la rediffusion de messages.
- Définissez comment la charge de travail comprend que sa dépendance a mené à bien sa tâche. Par exemple, lorsque votre charge de travail invoque une fonction Lambda de manière asynchrone, Lambda place l'événement dans une file d'attente, et renvoie une réponse de succès sans plus d'informations. Une fois le traitement terminé, la fonction Lambda peut envoyer le résultat à une destination configurable en fonction du succès ou de l'échec.
- Augmentez votre charge de travail pour gérer les messages dupliqués en tirant parti de l'idempotence. L'idempotence signifie que les résultats de la charge de travail ne changent pas

même si elle est générée plusieurs fois pour le même message. Il est important de souligner que les services de messagerie ou de streaming redistribueront un message en cas de panne du réseau ou en l'absence de réception d'un accusé de réception.

- Si la charge de travail n'obtient pas de réponse de sa dépendance, elle doit soumettre à nouveau la demande. Envisagez de limiter le nombre de tentatives pour préserver le processeur, la mémoire et les ressources réseau de votre charge de travail afin de gérer d'autres demandes. La documentation AWS Lambda montre comment gérer les erreurs lors d'une invocation asynchrone.
- Tirez parti des outils d'observabilité, de débogage et de suivi appropriés pour gérer et exploiter la communication asynchrone de la charge de travail avec ses dépendances. Vous pouvez utiliser Amazon CloudWatch pour surveiller les services de messagerie et de streaming d'événements. Vous pouvez également utiliser votre charge de travail avec AWS X-Ray pour obtenir rapidement des informations permettant de résoudre les problèmes.

Dépendance par lots

Les systèmes par lots prennent les données d'entrée, lancent une série de tâches pour les traiter et produisent certaines données de sortie, sans intervention manuelle. En fonction de la taille des données, les tâches peuvent s'exécuter pendant une durée allant de quelques minutes à, dans certains cas, plusieurs jours. Lorsque la charge de travail communique avec sa dépendance par lots, suivez les conseils ci-dessous :

- Définissez la fenêtre de temps pendant laquelle la charge de travail doit exécuter le traitement par lots. La charge de travail peut configurer un modèle de récurrence pour invoquer un système de traitement par lots (par exemple, toutes les heures ou à la fin de chaque mois).
- Déterminez l'emplacement de l'entrée des données et de la sortie des données traitées. Choisissez un service de stockage, tel qu'Amazon Simple Storage Services (Amazon S3), Amazon Elastic File System (Amazon EFS) et Amazon FSx pour Lustre, qui permet à votre charge de travail de lire et d'écrire des fichiers à l'échelle.
- Si votre charge de travail doit invoquer plusieurs tâches par lots, vous pouvez en tirer parti de AWS Step Functions pour simplifier l'orchestration des tâches par lots exécutées dans AWS ou sur site. Cet exemple de projet illustre l'orchestration de traitements par lots à l'aide de Step Functions, AWS Batch et Lambda.
- Surveillez les tâches par lots pour détecter d'éventuelles anomalies, telles qu'une tâche qui prend plus de temps que prévu. Vous pouvez utiliser des outils tels que CloudWatch Container Insights pour surveiller les environnements AWS Batch et les tâches. Dans ce cas, la charge de travail empêcherait le début de la tâche suivante et informerait le personnel concerné de l'exception.

Ressources

Documents connexes :

- Opérations : Surveillance et observabilité AWS Cloud
- L'Amazon Builders' Library : défis liés aux systèmes distribués
- REL11-BP05 Utiliser la stabilité statique pour éviter les comportements bimodaux
- Guide du développeur AWS Lambda : Gestion des erreurs et tentatives automatiques dans AWS Lambda
- Réglage des paramètres de demande HTTP d'AWS SDK Java pour les applications Amazon DynamoDB sensibles à la latence
- Messagerie AWS
- Qu'est-ce que le streaming de données ?
- Guide du développeur AWS Lambda : Invocation asynchrone
- FAQ Amazon Simple Queue Service : Files d'attente FIFO
- Guide du développeur Amazon Kinesis Data Streams : Gestion des enregistrements en double
- Guide du développeur Amazon Simple Queue Service : Métriques CloudWatch disponibles pour Amazon SQS
- Guide du développeur Amazon Kinesis Data Streams : Surveillance du service Amazon Kinesis Data Streams avec Amazon CloudWatch
- Guide du développeur AWS X-Ray : Concepts AWS X-Ray
- Exemples AWS sur GitHub : Application AWS Step Functions Complex Orchestrator
- Guide de l'utilisateur AWS Batch : AWS Batch CloudWatch Container Insights

Vidéos connexes :

- AWS Summit SF 2022 - Full-stack observability and application monitoring with AWS (COP310)

Outils associés :

- Amazon CloudWatch
- Amazon CloudWatch Logs
- AWS X-Ray
- Amazon Simple Storage Service (Amazon S3)

- Amazon Elastic File System (Amazon EFS)
- Amazon FSx pour Lustre
- AWS Step Functions
- AWS Batch

REL04-BP02 Implémenter des dépendances faiblement couplées

Des dépendances telles que des systèmes de file d'attente, des systèmes de streaming, des flux de travail et des équilibreurs de charge sont couplées faiblement. Le couplage faible permet d'isoler le comportement d'un composant des autres composants qui en dépendent, ce qui augmente la résilience et l'agilité.

Le découplage des dépendances, telles que les systèmes de file d'attente, les systèmes de streaming et les flux de travail, permet de minimiser l'impact des modifications ou des défaillances sur un système. Cette séparation empêche le comportement d'un composant d'affecter les autres qui en dépendent, améliorant ainsi la résilience et l'agilité.

Dans les systèmes couplés fortement, la modification d'un composant peut nécessiter de modifier d'autres composants qui en dépendent, ce qui entraîne une dégradation des performances de tous les composants. Le couplage faible rompt cette dépendance de sorte que les composants dépendants n'ont besoin que de connaître l'interface publiée et sa version. La mise en œuvre d'un couplage faible entre les dépendances permet d'isoler une défaillance dans l'une afin de ne pas en impacter une autre.

Le couplage faible vous permet de modifier le code ou d'ajouter des fonctionnalités à un composant tout en minimisant les risques pour les autres composants qui en dépendent. Il offre également une résilience granulaire au niveau des composants, ce qui vous permet d'augmenter horizontalement voire de modifier la mise en œuvre sous-jacente de la dépendance.

Pour améliorer encore la résilience par un couplage faible, dans la mesure du possible, rendez asynchrones les interactions des composants. Ce modèle convient à toute interaction qui ne nécessite pas une réponse immédiate et pour laquelle une confirmation de l'enregistrement d'une requête suffira. Il implique un composant qui génère des événements et un autre qui les consomme. Les deux composants ne s'intègrent pas par point-to-point interaction directe, mais généralement par le biais d'une couche de stockage durable intermédiaire, telle qu'une SQS file d'attente Amazon, une plateforme de données de streaming telle qu'Amazon Kinesis ou. AWS Step Functions

Figure 4 : Les dépendances telles que des systèmes de file d’attente et des équilibreurs de charge sont couplées faiblement

Amazon fait la SQS queue et ce ne AWS Step Functions sont que deux moyens d'ajouter une couche intermédiaire pour un couplage souple. Des architectures axées sur les événements peuvent également être créées à l'aide d' AWS Cloud Amazon EventBridge, qui peut isoler les clients (producteurs d'événements) des services sur lesquels ils comptent (consommateurs d'événements). Amazon Simple Notification Service (AmazonSNS) est une solution efficace lorsque vous avez besoin d'une messagerie push à haut débit. many-to-many À l'aide d'Amazon SNS Topics, les systèmes de vos éditeurs peuvent diffuser les messages vers un grand nombre de points de terminaison d'abonnés pour un traitement parallèle.

Bien que les files d’attente offrent plusieurs avantages, dans la plupart des systèmes en temps réel stricts, les requêtes antérieures à un seuil (souvent en secondes) sont considérées comme obsolètes (le client a abandonné et n’attend plus de réponse). En conséquence, elles ne sont pas traitées. De cette façon, les requêtes plus récentes (et probablement toujours valides) peuvent être traitées à la place.

Résultat souhaité : la mise en œuvre de dépendances faiblement couplées vous permet de minimiser la surface de défaillance au niveau du composant, ce qui permet de diagnostiquer et de résoudre les problèmes. Elle simplifie également les cycles de développement en permettant aux équipes de mettre en œuvre des modifications à un niveau modulaire sans affecter les performances des autres composants qui en dépendent. Avec cette approche, il est possible d’augmenter horizontalement un composant en fonction des besoins en ressources et de l’utilisation de ce composant, ce qui contribue à améliorer la rentabilité.

Anti-modèles courants :

- Déploiement d’une charge de travail monolithique.
- Invocation directe APIs entre les niveaux de charge de travail sans possibilité de basculement ou de traitement asynchrone de la demande.
- Couplage fort à l’aide de données partagées. Les systèmes couplés faiblement évitent de partager des données par le biais de bases de données partagées ou d’autres formes de stockage de données couplées fortement, qui peuvent réintroduire un couplage fort et entraver la capacité de mise à l’échelle.
- Ignorer la contre-pression. Votre charge de travail doit être capable de ralentir ou d’arrêter les données entrantes lorsqu’un composant ne peut pas les traiter au même rythme.

Avantages du respect de cette bonne pratique : le couplage faible permet d'isoler le comportement d'un composant des autres composants qui en dépendent, ce qui augmente la résilience et l'agilité. La défaillance d'un composant est isolée des autres.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : élevé

Directives d'implémentation

Implémentez des dépendances couplées faiblement. Différentes solutions permettent de créer des applications couplées faiblement. Il s'agit notamment de services permettant de mettre en œuvre des files d'attente entièrement gérées, des flux de travail automatisés, de réaction aux événements, etc., qui peuvent aider à isoler le comportement des composants par rapport aux autres composants, augmentant ainsi la résilience et l'agilité. APIs

- Créez des architectures pilotées par les événements : Amazon vous EventBridge aide à créer des architectures pilotées par les événements faiblement couplées et distribuées.
- Implémenter des files d'attente dans les systèmes distribués : vous pouvez utiliser Amazon Simple Queue Service (AmazonSQS) pour intégrer et découpler les systèmes distribués.
- Conteneuriser les composants sous forme de microservices : les microservices permettent aux équipes de créer des applications composées de petits composants indépendants qui communiquent via des canaux bien définis. APIs Amazon Elastic Container Service (AmazonECS) et Amazon Elastic Kubernetes Service (EKSAmazon) peuvent vous aider à démarrer plus rapidement avec les conteneurs.
- Gérez les flux de travail avec Step Functions : Step Functions vous aide à coordonner plusieurs AWS services dans des flux de travail flexibles.
- Tirez parti des architectures de messagerie publish-subscribe (pub/sub) : Amazon Simple Notification Service (AmazonSNS) assure la transmission des messages des éditeurs aux abonnés (également appelés producteurs et consommateurs).

Étapes d'implémentation

- Les composants d'une architecture basée sur les événements sont initiés par des événements. Les événements sont des actions qui se produisent dans un système (par exemple, un utilisateur ajoute un article à un panier). Lorsque l'action aboutit, un événement est généré et active le composant suivant du système.
  - Création d'applications basées sur les événements avec Amazon EventBridge

    - AWS re:Invent 2022 - Conception d'intégrations pilotées par des événements à l'aide d'Amazon EventBridge
- Les systèmes de messagerie distribuée comportent trois parties principales qui doivent être mises en œuvre pour une architecture basée sur des files d'attente. Ils incluent les composants du système distribué, la file d'attente utilisée pour le découplage (distribuée sur les SQS serveurs Amazon) et les messages de la file d'attente. Dans un système classique, les producteurs envoient le message dans la file d'attente et le consommateur reçoit le message de la file d'attente. La file d'attente stocke les messages sur plusieurs SQS serveurs Amazon à des fins de redondance.
    - SQSArchitecture Amazon de base
    - Envoyer des messages entre des applications distribuées avec Amazon Simple Queue Service
- Lorsqu'ils sont bien utilisés, les microservices améliorent la maintenabilité et la capacité de mise à l'échelle, car les composants couplés faiblement sont gérés par des équipes indépendantes. Ils permettent également d'isoler les comportements d'un composant en cas de changement.
    - Implémentation de microservices sur AWS
    - Let's Architect! Architecting microservices with containers
- AWS Step Functions Vous pouvez notamment créer des applications distribuées, automatiser des processus, orchestrer des microservices. L'orchestration de plusieurs composants dans un flux de travail automatisé vous permet de découpler des dépendances dans votre application.
    - Créez un flux de travail sans serveur avec et AWS Step FunctionsAWS Lambda
    - Commencer avec AWS Step Functions

Ressources

Documents connexes :

- Amazon EC2 : garantir l'impuissance
- L'Amazon Builders' Library : défis liés aux systèmes distribués
- L'Amazon Builders' Library : fiabilité, travail constant et une bonne tasse de café
- Qu'est-ce qu'Amazon EventBridge ?
- Qu'est-ce qu'Amazon Simple Queue Service ?
- Rompre avec votre monolithe
- Orchestrez des microservices basés sur des files d'attente avec Amazon et Amazon AWS Step Functions SQS

- SQSArchitecture Amazon de base
- Architecture basée sur des files d'attente

Vidéos connexes :

- AWS Sommet de New York 2019 : introduction aux architectures événementielles et à Amazon EventBridge (05) MAD2
- AWS re:Invent 2018 : Boucles serrées et ouverture d'esprit : comment prendre le contrôle des systèmes, grands et petits ARC337 (y compris le couplage lâche, le travail constant, la stabilité statique)
- AWS re:Invent 2019 : Passage à des architectures pilotées par les événements (08) SVS3
- AWS re:Invent 2019 : applications évolutives pilotées par des événements sans serveur utilisant Amazon et Lambda SQS
- AWS re:Invent 2022 - Conception d'intégrations pilotées par des événements à l'aide d'Amazon EventBridge
- AWS re:Invent 2017 : Elastic Load Balancing : analyse approfondie et meilleures pratiques

REL04-BP03 Faire un travail constant

Les systèmes peuvent échouer en cas de modifications importantes et rapides de la charge. Par exemple, si votre charge de travail effectue une surveillance de l'état de milliers de serveurs, elle doit envoyer chaque fois une charge utile de la même taille (un instantané complet de l'état actuel). Qu'aucun des serveurs ne présente de problème ou qu'ils en connaissent tous, le système de surveillance de l'état effectue un travail constant sans modifications importantes ni rapides.

Par exemple, si le système de surveillance de l'état surveille 100 000 serveurs, la charge sur celui-ci est nominale avec le taux de défaillance normalement faible du serveur. En revanche, si un événement majeur rendait la moitié de ces serveurs défectueux, le système de surveillance de l'état serait submergé en tentant de mettre à jour les systèmes de notification et de communiquer l'état à ses clients. Le système de surveillance de l'état devrait donc envoyer un instantané complet de l'état actuel à chaque fois. 100 000 états de santé du serveur, chacun représenté par un octet, ne représenteraient qu'une charge utile de 12,5 Ko. Qu'aucun des serveurs ne présente de problème ou qu'ils en connaissent tous, le système de surveillance de l'état effectue un travail constant, et les modifications importantes et rapides ne menacent pas la stabilité du système. C'est ainsi qu'Amazon Route 53 gère les surveillances de l'état des points de terminaison (tels que les adresses IP) pour déterminer comment les utilisateurs finaux sont acheminés vers eux.

Niveau d’exposition au risque si cette bonne pratique n’est pas respectée : faible

Directives d’implémentation

- Effectuez un travail constant : les systèmes peuvent échouer lorsque la charge connaît des changements rapides et importants.
- Implémentez des dépendances couplées faiblement. Des dépendances telles que des systèmes de file d’attente, des systèmes de streaming, des flux de travail et des équilibreurs de charge sont couplées faiblement. Le couplage faible permet d’isoler le comportement d’un composant des autres composants qui en dépendent, ce qui augmente la résilience et l’agilité.
  - L’Amazon Builders’ Library : fiabilité, travail constant et une bonne tasse de café
  - AWS re:Invent 2018 : Boucles serrées et ouverture d'esprit : comment prendre le contrôle des systèmes, grands et petits ARC337 (y compris un travail constant)
    - Pour l’exemple d’un système de surveillance de l’état surveillant 100 000 serveurs, concevez les charges de travail de manière à ce que les tailles de charge utile restent constantes, quel que soit le nombre de réussites ou d’échecs.

Ressources

Documents connexes :

- Amazon EC2 : garantir l'impuissance
- L’Amazon Builders’ Library : défis liés aux systèmes distribués
- L’Amazon Builders’ Library : fiabilité, travail constant et une bonne tasse de café

Vidéos connexes :

- AWS Sommet de New York 2019 : introduction aux architectures événementielles et à Amazon EventBridge (05) MAD2
- AWS re:Invent 2018 : Boucles serrées et ouverture d'esprit : comment prendre le contrôle des systèmes, grands et petits ARC337 (y compris un travail constant)
- AWS re:Invent 2018 : Boucles serrées et ouverture d'esprit : comment prendre le contrôle des systèmes, grands et petits ARC337 (y compris le couplage lâche, le travail constant, la stabilité statique)
- AWS re:Invent 2019 : Passage à des architectures pilotées par les événements (08) SVS3

REL04-BP04 Rendre les opérations de mutation idempotentes

Un service idempotent garantit que chaque demande est traitée une seule fois, de sorte que la soumission de plusieurs demandes identiques ait le même effet que la soumission d'une seule demande. Il est ainsi plus facile pour un client d'implémenter de nouvelles tentatives sans craindre qu'une demande soit traitée plusieurs fois par erreur. Pour ce faire, les clients peuvent émettre des demandes d'API avec un jeton d'idempotence, qui est utilisé chaque fois que la demande est répétée. Une API de service idempotente utilise le jeton pour renvoyer une réponse identique à la réponse qui a été renvoyée la première fois que la demande a été traitée, même si l'état sous-jacent du système a changé.

Dans un système distribué, il est relativement simple d'effectuer une action au plus une fois (le client soumet une seule demande) ou au moins une fois (le client continue à soumettre des demandes jusqu'à ce qu'il reçoive une confirmation de succès). Il est plus difficile de garantir qu'une action est exécutée exactement une fois, de sorte que la soumission de plusieurs demandes identiques a le même effet qu'une seule demande. En utilisant des jetons d'idempotence dans les API, les services peuvent recevoir une demande de mutation une ou plusieurs fois sans avoir besoin de créer des enregistrements en double ou des effets secondaires.

Résultat escompté : vous disposez d'une approche cohérente, bien documentée et largement adoptée pour garantir l'idempotence sur l'ensemble des composants et services.

Anti-modèles courants :

- Vous appliquez l'idempotence sans distinction, même lorsque cela n'est pas nécessaire.
- Vous introduisez une logique trop complexe pour implémenter l'idempotence.
- Vous utilisez les horodatages comme des clés pour l'idempotence. Cela peut entraîner des inexactitudes en raison d'un décalage d'horloge ou du fait que plusieurs clients utilisent les mêmes horodatages pour appliquer les modifications.
- Vous stockez des données utiles complètes à des fins d'idempotence. Dans cette approche, vous enregistrez des données utiles complètes pour chaque demande et vous les remplacez à chaque nouvelle demande. Cela peut dégrader les performances et affecter la capacité de mise à l'échelle.
- Vous générez des clés de manière incohérente entre les services. En l'absence de clés cohérentes, les services peuvent ne pas reconnaître les demandes en double, ce qui peut conduire à des résultats indésirables.

Avantages liés au respect de cette bonne pratique :

- Capacité de mise à l'échelle accrue : le système peut gérer les nouvelles tentatives et les demandes en double sans avoir à appliquer une logique supplémentaire ni à gérer des états complexes.
- Fiabilité améliorée : l'idempotence aide les services à traiter plusieurs demandes identiques de manière cohérente, ce qui réduit le risque d'effets secondaires indésirables ou de doublons d'enregistrements. Cela est particulièrement important dans les systèmes distribués, où les défaillances du réseau et les nouvelles tentatives sont communes.
- Cohérence des données améliorée : étant donné qu'une même demande produit la même réponse, l'idempotence permet de maintenir la cohérence des données dans les systèmes distribués. Cela est essentiel pour maintenir l'intégrité des transactions et des opérations.
- Gestion des erreurs : les jetons d'idempotence facilitent la gestion des erreurs. Si un client ne reçoit pas de réponse en raison d'un problème, il peut renvoyer la demande en toute sécurité avec le même jeton d'idempotence.
- Transparence opérationnelle : l'idempotence permet une meilleure surveillance et une meilleure journalisation. Les services peuvent consigner les demandes avec leurs jetons d'idempotence, ce qui facilite le suivi et le débogage des problèmes.
- Contrat d'API simplifié : il peut simplifier le contrat entre les systèmes côté client et serveur et réduire la crainte d'un traitement erroné des données.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : moyen

Directives d'implémentation

Dans un système distribué, il est relativement simple d'effectuer une action au plus une fois (le client soumet une seule demande) ou au moins une fois (le client continue à soumettre des demandes jusqu'à ce que le succès soit confirmé). Toutefois, il est difficile de mettre en œuvre un comportement exactement une fois. Pour ce faire, vos clients doivent générer et fournir un jeton d'idempotence pour chaque demande.

En utilisant des jetons d'idempotence, un service peut faire la distinction entre de nouvelles demandes et des demandes répétées. Lorsqu'un service reçoit une demande contenant un jeton d'idempotence, il vérifie si le jeton a déjà été utilisé. Si le jeton a été utilisé, le service extrait et retourne la réponse stockée. Si le jeton est nouveau, le service traite la demande, stocke la réponse avec le jeton, puis retourne la réponse. Ce mécanisme rend toutes les réponses idempotentes, ce qui améliore la fiabilité et la cohérence du système distribué.

L'idempotence est également un comportement important des architectures axées sur les événements. Ces architectures s'appuient généralement sur une file d'attente de messages telle qu'Amazon SQS, Amazon MQ, Amazon Kinesis Streams ou Amazon Managed Streaming for Apache Kafka (MSK). Dans certaines circonstances, un message qui n'a été publié qu'une seule fois peut être envoyé accidentellement plusieurs fois. Lorsqu'un diffuseur de publication génère et inclut des jetons d'idempotence dans des messages, il demande que le traitement de tout message reçu en double ne donne pas lieu à une action répétée pour le même message. Les consommateurs doivent suivre chaque jeton reçu et ignorer les messages contenant des jetons dupliqués.

Les services et les consommateurs doivent également transmettre le jeton d'idempotence reçu à tous les services en aval qu'il appelle. Chaque service en aval de la chaîne de traitement est de la même manière responsable de la mise en œuvre de l'idempotence afin d'éviter l'effet secondaire consistant à traiter un message plusieurs fois.

Étapes d'implémentation

1. Identification des opérations idempotentes

   Déterminez quelles opérations nécessitent l'idempotence. Il s'agit généralement des méthodes HTTP POST, PUT et DELETE et des opérations d'insertion, de mise à jour ou de suppression de base de données. Les opérations qui n'entraînent pas de mutation d'état, telles que les requêtes en lecture seule, ne nécessitent généralement pas l'idempotence, sauf si elles ont des effets secondaires.
2. Utilisation d'identifiants uniques

   Incluez un jeton unique dans chaque demande d'opération idempotente envoyée par l'expéditeur, soit directement dans la demande, soit au sein de ses métadonnées (par exemple, un en-tête HTTP). Cela permet au destinataire de reconnaître et de traiter les demandes ou opérations dupliquées. Les identifiants couramment utilisés pour les jetons incluent les identifiants uniques universels (UUID) et les identifiants KSUID (K-Sortable Unique Identifiers).
3. Suivi et gestion de l'état

   Tenez à jour l'état de chaque opération ou demande dans votre charge de travail. Pour ce faire, vous pouvez stocker le jeton d'idempotence et l'état correspondant (en attente, terminé ou échec) dans une base de données, un cache ou un autre stockage permanent. Ces informations d'état permettent à la charge de travail d'identifier et de traiter les demandes ou opérations en double.

   Maintenez la cohérence et l'atomicité en utilisant des mécanismes de contrôle de simultanéité appropriés si nécessaire, tels que des verrous, des transactions ou des contrôles de simultanéité

optimiste. Cela inclut le processus d'enregistrement du jeton idempotent et d'exécution de toutes les opérations de mutation associées au traitement de la demande. Cela contribue à éviter la survenue de conditions de concurrence et vérifie que les opérations idempotentes se déroulent correctement.

Supprimez régulièrement les anciens jetons d'idempotence de l'entrepôt de données pour gérer le stockage et les performances. Si votre système de stockage les prend en charge, pensez à utiliser des horodatages d'expiration pour les données (souvent appelés « durée de vie » ou valeurs TTL). La probabilité de réutilisation des jetons d'idempotence diminue avec le temps.

Les options de stockage AWS courantes généralement utilisées pour le stockage des jetons d'idempotence et de l'état associé incluent :

- Amazon DynamoDB : DynamoDB est un service de base de données NoSQL qui fournit des performances à faible latence et une haute disponibilité, ce qui le rend parfaitement adapté au stockage de données liées à l'idempotence. Le modèle de données clé-valeur et document de DynamoDB permet de stocker et d'extraire efficacement les jetons d'idempotence et les informations d'état associées. DynamoDB peut également faire expirer automatiquement les jetons d'idempotence si votre application définit une valeur TTL à leur insertion.
- Amazon ElastiCache : ElastiCache peut stocker des jetons d'idempotence à haut débit, à faible latence et à faible coût. ElastiCache (Redis) et ElastiCache (Memcached) peuvent tous les deux également faire expirer automatiquement les jetons d'idempotence si votre application définit une valeur TTL à leur insertion.
- Amazon Relational Database Service (RDS) : vous pouvez utiliser Amazon RDS pour stocker les jetons d'idempotence et les informations d'état associées, en particulier si votre application utilise déjà une base de données relationnelle à d'autres fins.
- Amazon Simple Storage Service (S3) : Amazon S3 est un service de stockage d'objets hautement évolutif et durable qui peut être utilisé pour stocker les jetons d'idempotence et les métadonnées associées. Les capacités de gestion des versions de S3 peuvent être particulièrement utiles pour maintenir l'état des opérations idempotentes. Le choix du service de stockage dépend généralement de facteurs tels que le volume de données liées à l'idempotence, les caractéristiques de performances requises, le besoin de durabilité et de disponibilité, et la manière dont le mécanisme d'idempotence s'intègre dans l'architecture globale de la charge de travail.

4. Mise en œuvre des opérations idempotentes

Concevez vos composants d'API et de charge de travail de manière à ce qu'ils soient idempotents. Incorporez des vérifications d'idempotence dans vos composants de charge de travail. Avant de traiter une demande ou d'effectuer une opération, vérifiez si l'identifiant unique a déjà été traité. Si tel est le cas, renvoyez le résultat précédent au lieu de réexécuter l'opération. Par exemple, si un client envoie une demande de création d'utilisateur, vérifiez si un utilisateur possédant le même identifiant unique existe déjà. Si un tel utilisateur existe, ses informations doivent être renvoyées au lieu de créer un nouvel utilisateur. De même, si un consommateur de file d'attente reçoit un message contenant un jeton d'idempotence dupliqué, le consommateur doit ignorer le message.

Créez des suites de tests complètes qui valident l'idempotence des demandes. Elles doivent couvrir un large éventail de scénarios, tels que des demandes réussies, des demandes ayant échoué et des demandes dupliquées.

Si votre charge de travail tire parti des fonctions AWS Lambda, envisagez d'utiliser Powertools for AWS Lambda. Powertools for AWS Lambda est une boîte à outils pour développeurs permettant de mettre en œuvre les bonnes pratiques en matière de technologies sans serveur et d'augmenter la rapidité des développeurs dans le cadre de l'utilisation des fonctions AWS Lambda. En particulier, il fournit un utilitaire pour convertir vos fonctions Lambda en opérations idempotentes qu'il est possible de réessayer en toute sécurité.

5. Communication claire de l'idempotence

   Documentez vos composants d'API et de charge de travail afin de communiquer clairement la nature idempotente des opérations. Cela permet aux clients de comprendre le comportement attendu et de savoir comment interagir de manière fiable avec votre charge de travail.

6. Surveillance et audit

   Mettez en œuvre des mécanismes de surveillance et d'audit pour détecter tout problème lié à l'idempotence des réponses, tel que des variations de réponse inattendues ou un traitement excessif des demandes dupliquées. Cela peut vous aider à détecter et à étudier tout problème ou comportement inattendu lié à votre charge de travail.

Ressources

Bonnes pratiques associées :

- REL05-BP03 Contrôler et limiter les appels de nouvelle tentative
- REL06-BP01 Surveiller tous les composants de la charge de travail (génération)

- REL06-BP03 Envoyer des notifications (traitement et alarmes en temps réel)
- REL08-BP02 Intégrer les tests fonctionnels dans le cadre de votre déploiement

Documents connexes :

- Amazon Builders' Library : sécurisation des nouvelles tentatives avec des API idempotentes
- L'Amazon Builders' Library : défis liés aux systèmes distribués
- L'Amazon Builders' Library : fiabilité, travail constant et une bonne tasse de café
- Amazon Elastic Container Service : garantie de l'idempotence
- Comment puis-je rendre ma fonction Lambda idempotente ?
- Procédure pour garantir l'idempotence dans les demandes d'API Amazon EC2

Vidéos connexes :

- Création d'applications distribuées avec une architecture axée sur les événements – AWS Online Tech Talks
- AWS re:Invent 2023 - Building next-generation applications with event-driven architecture
- AWS re:Invent 2023 - Advanced integration patterns & trade-offs for loosely coupled systems
- AWS re:Invent 2023 - Advanced event-driven patterns with Amazon EventBridge
- AWS re:Invent 2018 - Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (inclut les concepts de couplage faible, travail constant et stabilité statique)
- AWS re:Invent 2019 - Moving to event-driven architectures (SVS308)

Outils associés :

- Idempotence avec Powertools AWS Lambda (Java)
- Idempotence avec Powertools AWS Lambda (Python)
- Page GitHub de Powertools AWS Lambda

Conception des interactions dans un système distribué pour résister aux défaillances ou les atténuer

Les systèmes distribués s'appuient sur des réseaux de communication pour interconnecter des composants (tels que des serveurs ou des services). Votre charge de travail doit fonctionner de manière fiable malgré la perte de données ou la latence sur ces réseaux. Les composants du système distribué doivent fonctionner de manière à ne pas avoir d'impact négatif sur les autres composants ou sur la charge de travail. Ces bonnes pratiques permettent aux charges de travail de résister aux contraintes ou aux défaillances, de s'en remettre plus rapidement et d'atténuer l'impact de ces altérations. Il en résulte une amélioration du temps moyen de récupération (MTTR).

Ces bonnes pratiques permettent d'éviter les défaillances et d'améliorer le temps moyen entre défaillances (MTBF).

Bonnes pratiques

- REL05-BP01 Implémenter une dégradation appropriée pour transformer les dépendances matérielles applicables en dépendances logicielles
- REL05-BP02 Limiter les demandes
- REL05-BP03 Contrôler et limiter les appels de nouvelle tentative
- REL05-BP04 Procéder à une interruption immédiate et limiter les files d'attente
- REL05-BP05 Définir les délais d'expiration des clients
- REL05-BP06 Rendre les systèmes apatrides dans la mesure du possible
- REL05-BP07 Mettre en œuvre des leviers de secours

REL05-BP01 Implémenter une dégradation appropriée pour transformer les dépendances matérielles applicables en dépendances logicielles

Les composants de l'application doivent continuer à exécuter leur fonction principale même si les dépendances deviennent indisponibles. Ils peuvent fournir des données légèrement obsolètes, des données alternatives ou même aucune donnée. Cela garantit que le fonctionnement global du système n'est que très peu entravé par des défaillances localisées tout en fournissant une valeur commerciale centrale.

Résultat escompté : lorsque les dépendances d'un composant ne sont pas en bon état, le composant lui-même peut continuer de fonctionner, mais de manière dégradée. Les modes de défaillance des

composants doivent être considérés comme un fonctionnement normal. Les flux de travail doivent être conçus de manière à ce que ces défaillances n'aboutissent pas à une défaillance complète ou qu'elles aboutissent au moins à des états prévisibles et récupérables.

Anti-modèles courants :

- Ne pas identifier les fonctionnalités métier essentielles nécessaires. Ne pas tester le fonctionnement des composants, même en cas de défaillance des dépendances.
- Aucune donnée n'est diffusée en cas d'erreur ou lorsqu'une seule dépendance parmi plusieurs n'est pas disponible et que des résultats partiels peuvent toujours être renvoyés.
- Création d'un état incohérent lorsqu'une transaction échoue partiellement.
- Ne pas disposer d'un autre moyen d'accéder à un magasin de paramètres central.
- Invalider ou vider l'état local à la suite d'un échec d'actualisation sans prendre en compte les conséquences d'une telle opération.

Avantages du respect de cette bonne pratique : une dégradation progressive améliore la disponibilité du système dans son ensemble et maintient la fonctionnalité des fonctions les plus importantes même en cas de panne.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : élevé

Directives d'implémentation

La mise en œuvre d'une dégradation progressive permet de minimiser l'impact des défaillances de dépendance sur le fonctionnement des composants. Idéalement, un composant détecte les défaillances liées aux dépendances et les contourne de manière à avoir un impact minimal sur les autres composants ou les clients.

L'architecture permettant une dégradation progressive implique de prendre en compte les modes de défaillance potentiels lors de la conception des dépendances. Pour chaque mode de défaillance, déterminez un moyen de fournir la plupart des fonctionnalités, ou les plus critiques d'entre elles, du composant aux appelants ou aux clients. Ces considérations peuvent devenir des exigences supplémentaires qui peuvent être testées et vérifiées. Idéalement, un composant est capable d'exécuter sa fonction principale de manière acceptable, même en cas de défaillance d'une ou de plusieurs dépendances.

Il s'agit tout autant d'une discussion commerciale que technique. Toutes les exigences commerciales sont importantes et doivent être satisfaites dans la mesure du possible. Cependant, il est tout de

même logique de se demander ce qui doit se passer lorsque toutes les exigences ne peuvent pas être satisfaites. Un système peut être conçu pour être disponible et cohérent, mais lorsqu'une exigence doit être supprimée, laquelle est la plus importante ? Pour le traitement des paiements, il peut s'agir de la cohérence. Pour une application en temps réel, il peut s'agir de la disponibilité. Pour un site Web orienté client, la réponse peut dépendre des attentes du client.

Ce que cela signifie dépend des exigences du composant et de ce qui doit être considéré comme sa fonction principale. Par exemple :

- un site Web d'e-commerce peut afficher des données provenant de plusieurs systèmes différents, par exemple des recommandations personnalisées, les produits les mieux classés et l'état des commandes des clients sur la page de destination. Lorsqu'un système en amont est défaillant, il est tout de même judicieux d'afficher tout le reste au lieu d'afficher une page d'erreur à un client.
- Un composant effectuant des écritures par lots peut toujours continuer à traiter un lot si l'une des opérations individuelles échoue. La mise en œuvre d'un mécanisme de nouvelle tentative doit être simple. Cela peut être fait en renvoyant à l'appelant des informations indiquant quelles opérations ont réussi, lesquelles ont échoué et pourquoi elles ont échoué, ou en plaçant les demandes ayant échoué dans une file d'attente de lettres mortes pour implémenter des tentatives asynchrones. Les informations relatives aux opérations ayant échoué doivent également être consignées.
- Un système qui traite les transactions doit vérifier que toutes les mises à jour individuelles sont exécutées ou qu'aucune d'entre elles ne l'est. Pour les transactions distribuées, le modèle Saga peut être utilisé pour annuler les opérations précédentes en cas d'échec d'une opération ultérieure de la même transaction. Ici, la fonction principale est de maintenir la cohérence.
- Les systèmes soumis à des contraintes de temps doivent être en mesure de gérer les dépendances qui ne répondent pas en temps voulu. Dans ces cas de figure, le modèle du disjoncteur peut être utilisé. Lorsque les réponses d'une dépendance commencent à expirer, le système peut passer à un état fermé où aucun appel supplémentaire n'est effectué.
- Une application peut lire des paramètres à partir d'un magasin de paramètres. Il peut être utile de créer des images de conteneur avec un ensemble de paramètres par défaut et de les utiliser si le magasin de paramètres n'est pas disponible.

Notez que les chemins empruntés en cas de défaillance d'un composant doivent être testés et doivent être nettement plus simples que le chemin principal. Généralement, les stratégies de repli doivent être évitées.

Étapes d’implémentation

Identifiez les dépendances externes et internes. Déterminez quels types de défaillances peuvent y survenir. Réfléchissez à des moyens de minimiser l’impact négatif sur les systèmes en amont et en aval, ainsi que sur les clients lors de ces défaillances.

Vous trouverez ci-dessous une liste des dépendances et la manière de les dégrader de façon appropriée en cas d’échec :

1. Défaillance partielle des dépendances : un composant peut adresser plusieurs demandes à des systèmes en aval, soit sous la forme de demandes multiples adressées à un système, soit sous celle d’une demande adressée à plusieurs systèmes. Selon le contexte métier, différentes méthodes de gestion peuvent être appropriées (pour plus de détails, voir les exemples précédents dans le guide de mise en œuvre).
2. Un système en aval est incapable de traiter les demandes en raison d’une charge élevée : si les demandes adressées à un système en aval échouent régulièrement, il n’est pas logique de continuer à réessayer. Cela peut créer une charge supplémentaire sur un système déjà surchargé et rendre la récupération plus difficile. Le modèle du disjoncteur peut être utilisé ici afin de surveiller les appels en échec vers un système en aval. Si un grand nombre d’appels échouent, il cessera d’envoyer d’autres demandes au système en aval et n’autorisera les appels qu’occasionnellement pour vérifier si le système en aval est à nouveau disponible.
3. Aucun magasin de paramètres n’est disponible : pour transformer un magasin de paramètres, vous pouvez utiliser la mise en cache des dépendances souples ou des valeurs par défaut saines incluses dans les images de conteneur ou de machine. Notez que ces valeurs par défaut doivent être tenues à jour et incluses dans les suites de tests.
4. Aucun service de surveillance ou autre dépendance non fonctionnelle n’est disponible : si un composant ne peut pas envoyer par intermittence des journaux, des métriques ou des traces à un service de surveillance central, il est souvent préférable de continuer à exécuter les fonctions métier comme d’habitude. Il est souvent inacceptable de ne pas enregistrer ni de pousser des métriques pendant une longue période. En outre, certains cas d’utilisation peuvent nécessiter des entrées d’audit complètes pour répondre aux exigences de conformité.
5. Il est possible qu’une instance principale d’une base de données relationnelle ne soit pas disponible : Amazon Relational Database Service, comme presque toutes les bases de données relationnelles, ne peut avoir qu’une seule instance de rédacteur principal. Cela crée un point de défaillance unique pour les charges de travail d’écriture et complique la mise à l’échelle. Ce problème peut être partiellement atténué en utilisant une configuration multi-AZ pour une haute disponibilité ou Amazon Aurora sans serveur pour une meilleure mise à l’échelle. Pour

des exigences de très haute disponibilité, il peut être judicieux de ne pas se fier du tout au rédacteur principal. Pour les requêtes qui se limitent à la lecture, des répliques de lecture peuvent être utilisées, ce qui assure la redondance et la possibilité d'une augmentation horizontale, et pas seulement d'une augmentation verticale. Les écritures peuvent être mises en mémoire tampon, par exemple dans une file d'attente Amazon Simple Queue Service, afin que les demandes d'écriture des clients puissent toujours être acceptées même si le serveur principal est temporairement indisponible.

Ressources

Documents connexes :

- Amazon API Gateway : Limitation des demandes d'API pour améliorer le débit
- Coupe-circuit (présentation du coupe-circuit, ouvrage « Release It! »)
- Nouvelles tentatives après erreur et backoff exponentiel dans AWS
- Michael Nygard « Release It! Design and Deploy Production-Ready Software »
- L'Amazon Builders' Library : éviter le basculement dans les systèmes distribués
- L'Amazon Builders' Library : éviter les retards de file d'attente insurmontables
- L'Amazon Builders' Library : défis et stratégies de mise en cache
- Amazon Builders' Library : délais d'attente, nouvelles tentatives et backoff avec instabilité

Vidéos connexes :

- Retry, backoff, and jitter: AWS re:Invent 2019: Introducing The Amazon Builders' Library (DOP328)

REL05-BP02 Limiter les demandes

Limitez les demandes pour atténuer l'épuisement des ressources en cas d'augmentation imprévue de la demande. Les demandes inférieures aux taux de limitation sont traitées tandis que celles dépassant la limite définie sont rejetées avec un message de retour indiquant que la demande a dépassé la limite.

Résultat escompté : les pics de volume importants, qu'ils soient dus à une augmentation soudaine du trafic client, à des inondations ou à des tempêtes de nouvelles tentatives, sont atténués par la limitation des demandes, ce qui permet aux charges de travail de poursuivre le traitement normal du volume de demandes pris en charge.

Anti-modèles courants :

- Les limitations des points de terminaison de l'API ne sont pas implémentées ou leurs valeurs par défaut sont conservées sans tenir compte des volumes attendus.
- Les points de terminaison de l'API ne sont pas testés en termes de charge ou les limites de régulation ne sont pas testées.
- Limiter les taux de demandes sans tenir compte de la taille ou de la complexité des demandes.
- Tester les taux de demande maximaux ou la taille maximale des demandes, mais pas les deux simultanément.
- Les ressources ne sont pas provisionnées selon les mêmes limites établies lors des tests.
- Aucun plan d'utilisation n'a été configuré ni envisagé pour les utilisateurs d'API d'application à application (A2A).
- Les utilisateurs de files d'attente qui mettent à l'échelle horizontalement ne disposent pas de paramètres de simultanéité maximaux configurés.
- La limitation du débit par adresse IP n'a pas été mise en œuvre.

Avantages du respect de cette bonne pratique : les charges de travail qui fixent des limites peuvent fonctionner normalement et traiter correctement le chargement des demandes acceptées en cas de pics de volume inattendus. Les pics soudains ou soutenus de demandes adressées aux API et aux files d'attente sont limités et n'épuisent pas les ressources de traitement des demandes. Les limites de débit limitent les requêtes individuelles afin que les volumes élevés de trafic provenant d'une seule adresse IP ou d'un seul utilisateur d'API n'épuisent pas les ressources et n'aient pas d'impact sur les autres consommateurs.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : élevé

Directives d'implémentation

Les services doivent être conçus pour traiter une capacité connue de demandes ; cette capacité peut être établie par des tests de charge. Si les taux d'arrivée des demandes dépassent les limites, la réponse appropriée indique qu'une demande a été limitée. Cela permet au consommateur de gérer l'erreur et de réessayer ultérieurement.

Lorsque votre service nécessite une implémentation de limitation, pensez à implémenter l'algorithme du compartiment à jetons, dans lequel un jeton compte pour une demande. Les jetons sont rechargés à une vitesse limitée par seconde et vidés de manière asynchrone à raison d'un jeton par demande.

L’algorithme du compartiment à jetons.

Amazon API Gateway implémente l’algorithme du compartiment à jetons en fonction des limites du compte et de la région et il peut être configuré par client avec des plans d’utilisation. En outre, Amazon Simple Queue Service (Amazon SQS) et Amazon Kinesis peuvent mettre en mémoire tampon les demandes afin de réduire le taux de demandes et de permettre des taux de limitation plus élevés pour les demandes pouvant être traitées. Enfin, vous pouvez implémenter une limitation de débit avec AWS WAF pour limiter les consommateurs d’API spécifiques qui génèrent une charge anormalement élevée.

Étapes d’implémentation

Vous pouvez configurer API Gateway avec des limites de régulation pour vos API et renvoyer des erreurs `429 Too Many Requests` lorsque les limites sont dépassées. Vous pouvez utiliser AWS WAF avec votre AWS AppSync et vos points de terminaison API Gateway pour activer la limitation du débit par adresse IP. En outre, lorsque votre système peut tolérer un traitement asynchrone, vous pouvez placer les messages dans une file d’attente ou un flux pour accélérer les réponses aux clients du service, ce qui vous permet d’atteindre des taux de limitation plus élevés.

Avec le traitement asynchrone, lorsque vous avez configuré Amazon SQS comme source d’événements pour AWS Lambda, vous pouvez configurer une simultanéité maximale afin d’éviter que des taux d’événements élevés ne consomment le quota d’exécution simultanée du compte disponible nécessaire pour les autres services de votre charge de travail ou de votre compte.

Bien qu’API Gateway propose une implémentation gérée du compartiment à jetons, lorsque vous ne pouvez pas utiliser API Gateway, vous pouvez tirer parti des implémentations open source

spécifiques à la langue (voir les exemples associés dans Ressources) du compartiment à jetons pour vos services.

- Comprenez et configurez les limites de limitation d'API Gateway au niveau du compte par région, de l'API par étape et de la clé d'API par niveau de plan d'utilisation.
- Appliquez des règles de limitation de débit AWS WAF à API Gateway et aux points de terminaison AWS AppSync pour vous protéger contre les inondations et bloquer les adresses IP malveillantes. Les règles de limitation de débit peuvent également être configurées sur les clés d'API AWS AppSync pour les consommateurs A2A.
- Déterminez si vous avez besoin d'un contrôle plus limitant qu'une limitation du débit pour les API AWS AppSync et, si c'est le cas, configurez un API Gateway devant votre point de terminaison AWS AppSync.
- Lorsque les files d'attente Amazon SQS sont configurées comme déclencheurs pour les consommateurs de files d'attente Lambda, définissez la simultanéité maximale sur une valeur qui traite suffisamment pour atteindre vos objectifs de niveau de service, mais qui ne respecte pas les limites de simultanéité ayant un impact sur les autres fonctions Lambda. Envisagez de définir une simultanéité réservée pour d'autres fonctions Lambda du même compte et de la même région lorsque vous utilisez des files d'attente avec Lambda.
- Utilisez API Gateway avec des intégrations de services natives vers Amazon SQS ou Kinesis pour mettre des demandes en mémoire tampon.
- Si vous ne pouvez pas utiliser API Gateway, examinez les bibliothèques spécifiques à la langue pour implémenter l'algorithme de compartiment à jetons adapté à votre charge de travail. Consultez la section des exemples et faites vos propres recherches pour trouver une bibliothèque appropriée.
- Testez les limites que vous envisagez de définir ou d'autoriser à augmenter, et documentez les limites testées.
- N'augmentez pas les limites au-delà de ce que vous avez établi lors des tests. Lorsque vous augmentez une limite, vérifiez que les ressources provisionnées sont déjà équivalentes ou supérieures à celles des scénarios de test avant d'appliquer l'augmentation.

Ressources

Bonnes pratiques associées :

- REL04-BP03 Faire un travail constant
- REL05-BP03 Contrôler et limiter les appels de nouvelle tentative

Documents connexes :

- Amazon API Gateway : Limitation des demandes d'API pour améliorer le débit
- AWS WAF : instruction de règle fréquentielle
- Introduction d'une simultanéité maximale de AWS Lambda en utilisant Amazon SQS comme source d'événements
- AWS Lambda : simultanéité maximale

Exemples connexes :

- Les trois principales règles AWS WAF basées sur le débit
- Bucket4j Java
- Jeton-compartiment Python
- Nœud jeton-compartiment
- Limitation du débit de threading du système .NET

Vidéos connexes :

- Implementing GraphQL API security best practices with AWS AppSync

Outils associés :

- Amazon API Gateway
- AWS AppSync
- Amazon SQS
- Amazon Kinesis
- AWS WAF
- Salle d'attente virtuelle sur AWS

REL05-BP03 Contrôler et limiter les appels de nouvelle tentative

Utilisez le backoff exponentiel pour relancer les demandes à des intervalles de plus en plus longs entre chaque nouvelle tentative. Introduisez un décalage entre les tentatives afin de randomiser les intervalles entre les tentatives. Limitez le nombre maximal de tentatives.

Résultat escompté : Les composants typiques d'un système logiciel distribué incluent les serveurs, les équilibreurs de charge, les bases de données et DNS les serveurs. Pendant le fonctionnement normal, ces composants peuvent répondre aux demandes par des erreurs temporaires ou limitées, ainsi que par des erreurs qui persisteraient indépendamment des nouvelles tentatives. Lorsque des clients adressent des demandes à des services, celles-ci consomment des ressources, notamment de la mémoire, des threads, des connexions, des ports ou toute autre ressource limitée. Le contrôle et la limitation des nouvelles tentatives constituent une stratégie visant à libérer et à minimiser la consommation de ressources afin que les composants du système soumis à des contraintes ne soient pas surchargés.

Lorsque le client demande une expiration du délai ou reçoit des réponses d'erreur, il doit décider de réessayer ou non. S'il recommence, il le fait avec un backoff exponentiel avec une instabilité et une valeur de nouvelle tentative maximale. Par conséquent, les services et processus back-end sont moins sollicités et le temps nécessaire pour s'autoréparer est réduit, ce qui se traduit par une récupération plus rapide et un traitement efficace des demandes.

Anti-modèles courants :

- Implémentation de nouvelles tentatives sans ajouter de valeurs de backoff exponentiel, d'instabilité et de nouvelles tentatives maximales. Le backoff et l'instabilité permettent d'éviter les pics de trafic artificiels dus à des tentatives involontaires coordonnées à intervalles réguliers.
- Implémentation de nouvelles tentatives sans tester leurs effets ou en supposant que les nouvelles tentatives sont déjà intégrées ou SDK sans test de scénarios de nouvelle tentative.
- Incapacité à comprendre les codes d'erreur publiés à partir des dépendances, ce qui entraîne une nouvelle tentative pour toutes les erreurs, y compris celles dont la cause claire indique un manque d'autorisation, une erreur de configuration ou toute autre condition qui, comme on pouvait s'y attendre, ne sera pas résolue sans intervention manuelle.
- Ne pas aborder les pratiques d'observabilité, notamment la surveillance et l'envoi d'alertes en cas de pannes de service répétées afin que les problèmes sous-jacents soient connus et puissent être résolus.
- Développement de mécanismes de nouvelle tentative personnalisés lorsque des fonctionnalités de nouvelle tentative intégrées ou tierces suffisent.
- Réessayer à plusieurs couches de votre pile d'applications d'une manière qui complique les nouvelles tentatives et augmente la consommation de ressources lors d'une tempête de nouvelles tentatives. Assurez-vous de comprendre comment ces erreurs affectent votre application et les dépendances sur lesquelles vous vous appuyez, puis implémentez les nouvelles tentatives à un seul niveau.

- Réessayer les appels de service qui ne sont pas idempotents, ce qui peut entraîner des effets secondaires inattendus tels que des résultats dupliqués.

Avantages du respect de cette bonne pratique : les nouvelles tentatives aident les clients à obtenir les résultats souhaités lorsque les requêtes échouent, mais elles font également perdre plus de temps au serveur pour obtenir les réponses souhaitées. Lorsque les défaillances sont rares ou transitoires, les nouvelles tentatives fonctionnent bien. Lorsque les défaillances sont causées par une surcharge de ressources, les nouvelles tentatives peuvent aggraver la situation. L'ajout d'un backoff exponentiel avec instabilité aux nouvelles tentatives des clients permet aux serveurs de se rétablir en cas de défaillance provoquée par une surcharge de ressources. L'instabilité permet d'éviter l'alignement des demandes en pics, tandis que le backoff réduit l'escalade de charge provoquée par l'ajout de nouvelles tentatives au chargement normal des demandes. Enfin, il est important de configurer un nombre maximal de nouvelles tentatives ou un temps écoulé afin d'éviter de créer des backlogs susceptibles d'entraîner des échecs métastables.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : élevé

Directives d'implémentation

Contrôler et limiter les appels de nouvelle tentative. Utilisez le backoff exponentiel pour réessayer après des intervalles progressivement plus longs. Introduisez l'instabilité pour randomiser les intervalles de nouvelle tentative et limiter le nombre maximal de nouvelles tentatives.

Certains AWS SDKs implémentent les nouvelles tentatives et le recul exponentiel par défaut. Utilisez ces AWS implémentations intégrées, le cas échéant, dans votre charge de travail. Implémentez une logique similaire dans votre charge de travail lorsque vous appelez des services qui sont idempotents et où les nouvelles tentatives améliorent la disponibilité de vos clients. Déterminez quels sont les délais d'expiration et quand les nouvelles tentatives doivent s'arrêter en fonction de votre cas d'utilisation. Créez et mettez en pratique des scénarios de test pour ces cas d'utilisation impliquant de nouvelles tentatives.

Étapes d'implémentation

- Déterminez la couche optimale de votre pile d'applications pour implémenter de nouvelles tentatives pour les services sur lesquels repose votre application.
- Tenez compte des stratégies de relance éprouvées SDKs qui mettent en œuvre des stratégies de relance éprouvées avec un retard et une instabilité exponentiels dans la langue de votre choix,

et privilégiez ces stratégies par rapport à l'écriture de vos propres implémentations de nouvelle tentative.

- Vérifiez que les services sont idempotents avant d'implémenter de nouvelles tentatives. Une fois les nouvelles tentatives mises en œuvre, assurez-vous qu'elles sont à la fois testées et régulièrement mises en œuvre en production.
- Lorsque vous appelez le AWS serviceAPIs, utilisez les options de configuration AWS SDKsAWS CLIet et comprenez la nouvelle tentative. Déterminez si les valeurs par défaut conviennent à votre cas d'utilisation, testez-les et ajustez-les si nécessaire.

Ressources

Bonnes pratiques associées :

- REL04-BP04 Rendre les opérations de mutation idempotentes
- REL05-BP02 Limiter les demandes
- REL05-BP04 Procéder à une interruption immédiate et limiter les files d'attente
- REL05-BP05 Définir les délais d'expiration des clients
- REL11-BP01 Surveiller tous les composants de la charge de travail pour détecter les défaillances

Documents connexes :

- Rétentatives d'erreur et retard exponentiel dans AWS
- Amazon Builders' Library : délais d'attente, nouvelles tentatives et backoff avec instabilité
- Instabilité et backoff exponentiel
- Sécuriser les nouvelles tentatives avec idempotent APIs

Exemples connexes :

- Nouvelle tentative Spring
- Nouvelle tentative Resilience4j

Vidéos connexes :

- Réessayez, attendez et agitez : AWS re:Invent 2019 : Introducing The Amazon Builders' Library () DOP328

Outils associés :

- AWS SDKset outils : comportement des nouvelles tentatives
- AWS Command Line Interface: nouvelles AWS CLI tentatives

REL05-BP04 Procéder à une interruption immédiate et limiter les files d'attente

Lorsqu'un service n'est pas en mesure de répondre correctement à une demande, procédez à son interruption immédiate. Cela permet la libération des ressources associées à une demande et donne la possibilité au service de récupérer s'il lui manque des ressources. L'interruption immédiate est un modèle de conception logicielle bien établi qui peut être exploité pour créer des charges de travail hautement fiables dans le cloud. La mise en file d'attente est également un modèle d'intégration d'entreprise bien établi qui permet de faciliter le chargement et de permettre aux clients de libérer des ressources lorsque le traitement asynchrone peut être toléré. Lorsqu'un service est capable de répondre correctement dans des conditions normales, mais échoue lorsque le taux de demandes est trop élevé, utilisez une file d'attente pour mettre les demandes en mémoire tampon. Toutefois, ne permettez pas l'accumulation de longs backlogs de files d'attente susceptibles d'entraîner le traitement de demandes obsolètes auxquelles un client a déjà renoncé.

Résultat escompté : lorsque les systèmes sont confrontés à des problèmes de ressources, à des dépassements de délai, à des exceptions ou à des pannes grises qui rendent les objectifs de niveau de service irréalisables, les stratégies d'interruption immédiate permettent d'accélérer la récupération du système. Les systèmes qui doivent absorber les pics de trafic et peuvent prendre en charge le traitement asynchrone peuvent améliorer la fiabilité en permettant aux clients de lancer rapidement des demandes grâce à l'utilisation des files d'attente pour mettre en mémoire tampon les demandes envoyées aux services back-end. Lors de la mise en mémoire tampon des demandes dans des files d'attente, des stratégies de gestion des files d'attente sont mises en œuvre pour éviter des backlogs insurmontables.

Anti-modèles courants :

- Mise en œuvre de files de messages sans configurer de files d'attente de lettres mortes (DLQ) ni d'alarmes sur les volumes DLQ pour détecter les défaillances d'un système.
- Il ne s'agit pas de mesurer l'ancienneté des messages dans une file d'attente, mais de mesurer la latence pour comprendre quand les utilisateurs prennent du retard ou si des erreurs entraînent de nouvelles tentatives.

- Conservation des messages en attente dans une file d'attente, alors qu'il n'est plus utile de traiter ces messages si l'entreprise n'en a plus besoin.
- La configuration de files d'attente du premier entré, premier sorti (FIFO) au moment du dernier entré, premier sorti (LIFO) permettrait de mieux répondre aux besoins des clients, par exemple lorsqu'un ordre strict n'est pas requis et que le traitement du backlog retarde toutes les nouvelles demandes urgentes, ce qui entraîne une violation des niveaux de service pour tous les clients.
- Exposition des files d'attente internes aux clients au lieu d'exposer les API qui gèrent la prise de travail et placent les demandes dans les files d'attente internes.
- La combinaison d'un trop grand nombre de types de demandes de travail dans une seule file d'attente peut aggraver les problèmes de backlog en répartissant la demande de ressources entre les types de demandes.
- Traitement de demandes complexes et simples dans la même file d'attente, malgré la nécessité d'une surveillance, de délais d'expiration et d'allocations de ressources différents.
- Absence de validation des entrées ou utilisation des assertions pour implémenter des mécanismes d'interruption immédiate dans les logiciels qui génèrent des exceptions vers des composants de niveau supérieur capables de gérer les erreurs de façon appropriée.
- Absence de suppression des ressources défectueuses du routage des requêtes, en particulier lorsque les défaillances sont grises, ce qui indique à la fois des réussites et des échecs en raison d'un plantage et d'un redémarrage, d'une panne de dépendance intermittente, d'une capacité réduite ou d'une perte de paquets réseau.

Avantages du respect de cette bonne pratique : les systèmes avec interruption immédiate sont plus faciles à déboguer et à corriger, et présentent souvent des problèmes de codage et de configuration avant la publication des versions en production. Les systèmes qui intègrent des stratégies de mise en file d'attente efficaces offrent une résilience et une fiabilité accrues face aux pics de trafic et aux pannes intermittentes du système.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : élevé

Directives d'implémentation

Les stratégies d'interruption immédiate peuvent être codées dans des solutions logicielles ou configurées dans l'infrastructure. En plus de leur capacité d'interruption immédiate, les files d'attente constituent une technique architecturale simple mais puissante qui permet de découpler les composants du système en douceur. Amazon CloudWatch fournit des fonctionnalités de surveillance et d'alerte en cas de défaillance. Une fois que l'on sait qu'un système est défaillant, des stratégies

d'atténuation peuvent être invoquées, notamment en cas de défaillance de ressources altérées. Quand des systèmes implémentent des files d'attente avec Amazon SQS et d'autres technologies de file d'attente pour faciliter le chargement, ils doivent tenir compte de la gestion des backlogs de files d'attente, ainsi que des défaillances de consommation de messages.

Étapes d'implémentation

- Implémentez des assertions programmatiques ou des métriques spécifiques dans votre logiciel et utilisez-les pour signaler explicitement les problèmes du système. Amazon CloudWatch vous aide à créer des métriques et des alarmes reposant sur le modèle de journal des applications et l'instrumentation du SDK.
- Utilisez les métriques et les alarmes CloudWatch pour éviter les problèmes liés à l'altération des ressources qui ajoutent de la latence au traitement ou qui échouent à plusieurs reprises à traiter les demandes.
- Utilisez le traitement asynchrone en concevant des API pour accepter les demandes et les ajouter aux files d'attente internes en utilisant Amazon SQS, puis en répondant au client émetteur du message par un message de réussite afin que le client puisse libérer des ressources et passer à autre chose pendant que les utilisateurs de la file d'attente back-end traitent les demandes.
- Mesurez et surveillez la latence de traitement des files d'attente en produisant une métrique CloudWatch chaque fois que vous retirez un message d'une file d'attente en comparant l'heure actuelle à l'horodatage du message.
- Lorsque des défaillances empêchent le bon traitement des messages ou que des pics de trafic concernent des volumes qui ne peuvent pas être traités conformément aux contrats de niveau de service, mettez de côté le trafic ancien ou excédentaire vers une file d'attente de débordement. Cela permet de traiter en priorité les nouvelles tâches et les tâches plus anciennes lorsque la capacité est disponible. Cette technique est une approximation du traitement LIFO et permet un traitement normal du système pour toutes les nouvelles tâches.
- Utilisez des lettres mortes ou réadaptez des files d'attente afin de déplacer les messages qui ne peuvent pas être traités hors du backlog vers un emplacement pouvant faire l'objet de recherches et de résolutions ultérieures.
- Réessayez ou, si cela est acceptable, supprimez les anciens messages en comparant l'heure actuelle à l'horodatage du message et en supprimant les messages qui ne sont plus pertinents pour le client demandeur.

Ressources

Bonnes pratiques associées:

- REL04-BP02 Implémenter des dépendances faiblement couplées
- REL05-BP02 Limiter les demandes
- REL05-BP03 Contrôler et limiter les appels de nouvelle tentative
- REL06-BP02 Définir et calculer des métriques (agrégation)
- REL06-BP07 Surveiller la traçabilité complète des demandes via votre système

Documents connexes :

- Éviter les backlogs insurmontables dans les files d'attente
- Interruption immédiate
- Comment puis-je empêcher un backlog de messages croissant dans ma file d'attente Amazon SQS ?
- Elastic Load Balancing : changement de zone
- Amazon Application Recovery Controller : contrôle du routage pour le basculement du trafic

Exemples connexes :

- Modèles d'intégration d'entreprise : canal des lettres mortes

Vidéos connexes :

- AWS re:Invent 2022 - Operating highly available Multi-AZ applications

Outils associés :

- Amazon SQS
- Amazon MQ
- AWS IoT Core
- Amazon CloudWatch

REL05-BP05 Définir les délais d'expiration des clients

Définissez les délais d’expiration de manière appropriée pour les connexions et les demandes, vérifiez-les systématiquement et ne vous fiez pas aux valeurs par défaut, car elles ne tiennent pas compte des spécificités de la charge de travail.

Résultat souhaité : les délais d’expiration du client doivent prendre en compte le coût pour le client, le serveur et la charge de travail associés à l’attente des demandes dont le traitement prend un temps anormal. Dans la mesure où il est impossible de connaître la cause exacte d’un délai d’attente, les clients doivent utiliser leur connaissance des services pour établir des attentes relatives aux causes probables et aux délais d’expiration appropriés.

Le délai d’expiration des connexions client dépend des valeurs configurées. Après avoir dépassé le délai imparti, les clients décident de revenir en arrière et de réessayer ou d’ouvrir un coupe-circuit. Ces modèles évitent d’émettre des demandes susceptibles d’exacerber un problème d’erreur sous-jacent.

Anti-modèles courants :

- Ne pas connaître les délais d’expiration du système ou les délais d’expiration par défaut.
- Ne pas connaître le délai normal d’exécution des demandes.
- Ne pas connaître les raisons pour lesquelles les demandes peuvent prendre un temps anormalement long à traiter, ni les coûts pour le client, le service ou les performances de la charge de travail associés à l’attente de ces traitements.
- Ne pas connaître la probabilité qu’un réseau défaillant entraîne l’échec d’une requête uniquement lorsque le délai d’expiration est atteint, ainsi que les coûts pour les performances du client et de la charge de travail si l’on n’adopte pas un délai d’expiration plus court.
- Ne pas tester les scénarios de délai d’expiration à la fois pour les connexions et les demandes.
- Définir des délais d’expiration trop élevés, ce qui peut entraîner de longs temps d’attente et augmenter l’utilisation des ressources.
- Définir des délais d’attente trop bas, ce qui entraîne des défaillances artificielles.
- Oublier les modèles pour gérer les erreurs de temporisation des appels distants, par exemple les disjoncteurs et les nouvelles tentatives.
- Ne pas envisager de surveiller les taux d’erreur des appels de service, les objectifs de niveau de service en matière de latence et les valeurs aberrantes en matière de latence. Ces métriques peuvent fournir des informations sur les délais d’attente agressifs ou permissifs.

Avantages du respect de cette bonne pratique : les délais d'expiration des appels distants sont configurés et les systèmes sont conçus pour gérer les délais d'expiration de façon appropriée afin de préserver les ressources lorsque les appels distants répondent de manière anormalement lente et que les erreurs de délai d'expiration sont gérées de façon appropriée par les clients du service.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : élevé

Directives d'implémentation

Définissez un délai d'expiration de la connexion et un délai d'expiration de la demande pour tout appel de dépendance de service et, plus généralement, pour tout appel entre processus. De nombreux cadres proposent des capacités de délai d'expiration intégrées, mais soyez prudent, car certains ont des valeurs par défaut infinies ou supérieures à ce qui est acceptable pour vos objectifs de service. Une valeur trop élevée réduit l'utilité du délai d'attente, car les ressources continuent d'être consommées pendant que le client attend l'expiration du délai. Une valeur trop faible peut générer un trafic accru sur le back-end et une latence accrue en raison du nombre excessif de demandes réessayées. Dans certains cas, cela peut entraîner des interruptions complètes, car toutes les demandes font l'objet d'une nouvelle tentative.

Tenez compte des points suivants lorsque vous déterminez des stratégies de délai d'expiration :

- Le traitement des demandes peut prendre plus de temps que d'habitude en raison de leur contenu, de défaillances d'un service cible ou d'une panne de partition réseau.
- Les demandes dont le contenu est anormalement coûteux peuvent consommer des ressources inutiles du serveur et du client. Dans ce cas, le fait d'avoir un délai d'expiration pour ces demandes et de ne pas réessayer peut préserver les ressources. Les services doivent également se protéger contre les contenus anormalement coûteux avec des limites et des délais d'expiration côté serveur.
- Les demandes qui prennent anormalement longtemps en raison d'une défaillance du service peuvent être interrompues et réessayées. Il convient de tenir compte des coûts de service liés à la demande et à la nouvelle tentative, mais si la cause est une déficience localisée, une nouvelle tentative ne sera probablement pas coûteuse et réduira la consommation de ressources du client. Le délai d'expiration peut également libérer des ressources du serveur en fonction de la nature de la déficience.
- Les demandes dont l'exécution prend beaucoup de temps parce que la demande ou la réponse n'a pas été envoyée par le réseau peuvent être interrompues et réessayées. La demande ou la réponse n'ayant pas été envoyée, il en aurait résulté un échec indépendamment de la durée du délai imparti. Dans ce cas, l'expiration du délai ne libérera pas les ressources du serveur, mais des ressources client et cela améliorera les performances de la charge de travail.

Tirez parti de modèles de conception bien établis, tels que les nouvelles tentatives et les disjoncteurs, pour gérer les délais d'attente avec élégance et prendre en charge les approches rapides. AWS SDKset AWS CLIpermettent de configurer les délais d'expiration des connexions et des demandes ainsi que les nouvelles tentatives avec un retard et une instabilité exponentiels. AWS Lambdales fonctions prennent en charge la configuration des délais d'attente, et avec AWS Step Functions, vous pouvez créer des disjoncteurs low code qui tirent parti des intégrations prédéfinies avec les services et. AWS SDKs AWS App Mesh Envoy fournit des fonctionnalités de délai d’expiration et de disjoncteur.

Étapes d’implémentation

- Configurez les délais d’expiration pour les appels de service à distance et profitez des fonctionnalités de délai d’expiration spécifique à la langue intégrées ou des bibliothèques de délai d’expiration open source.
- Lorsque votre charge de travail passe des appels avec un AWS SDK, consultez la documentation pour connaître la configuration du délai d'expiration spécifique à la langue.
  - Python
  - PHP
  - .NET
  - Ruby
  - Java
  - Go
  - Node.js
  - C++
- Lorsque vous utilisez des AWS CLI commandes AWS SDKs or dans votre charge de travail, configurez les valeurs de délai d'expiration par défaut en définissant les AWS valeurs par défaut pour `connectTimeoutInMillis` et. `tlsNegotiationTimeoutInMillis`
- Appliquez des options de ligne de commande `cli-connect-timeout` et `cli-read-timeout` contrôlez des AWS CLI commandes ponctuelles aux AWS services.
- Surveillez les appels de service à distance pour détecter les délais d’expiration et définissez des alarmes en cas d’erreurs persistantes afin de pouvoir gérer de manière proactive les scénarios d’erreur.
- Mettez en œuvre CloudWatch des mesures et CloudWatch une détection des anomalies sur les taux d'erreur des appels, les objectifs de niveau de service en matière de latence et les valeurs

aberrantes de latence afin de mieux comprendre la gestion des délais d'attente trop agressifs ou trop permissifs.

- Configurez les délais d’expiration des fonctions Lambda.
- APILes clients Gateway doivent implémenter leurs propres tentatives lors de la gestion des délais d'expiration. APIGateway prend en charge un délai d'intégration de 50 millisecondes à 29 secondes pour les intégrations en aval et ne réessaie pas lorsque les demandes d'intégration expirent.
- Implémentez le modèle de disjoncteur pour éviter de passer des appels à distance lorsque le délai d’expiration est écoulé. Ouvrez le circuit pour éviter les échecs d’appels et fermez-le lorsque les appels répondent normalement.
- Pour les charges de travail basées sur des conteneurs, consultez les fonctionnalités d’App Mesh Envoy pour tirer parti des délais d’attente et des disjoncteurs intégrés.
- AWS Step Functions À utiliser pour créer des disjoncteurs à faible code pour les appels de service à distance, en particulier lorsque vous faites appel à des intégrations Step Functions AWS natives SDKs et compatibles afin de simplifier votre charge de travail.

Ressources

Bonnes pratiques associées :

- REL05-BP03 Contrôler et limiter les appels de nouvelle tentative
- REL05-BP04 Procéder à une interruption immédiate et limiter les files d’attente
- REL06-BP07 Surveiller la traçabilité complète des demandes via votre système

Documents connexes :

- AWS SDK: tentatives et délais d'expiration
- Amazon Builders’ Library : délais d’attente, nouvelles tentatives et backoff avec instabilité
- Quotas Amazon API Gateway et remarques importantes
- AWS Command Line Interface : options de ligne de commande
- AWS SDK for Java 2.x: Configurer les API délais
- AWS Botocore utilisant l'objet de configuration et la référence de configuration
- AWS SDK pour .NET : nouvelles tentatives et délais d’expiration

- AWS Lambda : configuration des options de fonction Lambda

Exemples connexes :

- Utilisation du schéma du disjoncteur avec AWS Step Functions Amazon DynamoDB
- Martin Fowler : CircuitBreaker

Outils associés :

- AWS SDKs
- AWS Lambda
- Amazon SQS
- AWS Step Functions
- AWS Command Line Interface

REL05-BP06 Rendre les systèmes apatrides dans la mesure du possible

Les systèmes ne doivent pas exiger d'état ou doivent décharger un état de telle sorte qu'entre les différentes demandes client, il n'y ait pas de dépendance vis-à-vis des données stockées localement sur disque et en mémoire. Cela permet le remplacement à volonté des serveurs sans impact sur la disponibilité.

Lorsque des utilisateurs ou des services interagissent avec une application, ils exécutent souvent une série d'interactions qui forment une session. Une session correspond aux données uniques des utilisateurs qui persistent entre les requêtes pendant l'utilisation de l'application. Une application sans état n'a pas besoin de connaître les interactions précédentes et ne stocke pas d'informations de session.

Une fois conçu pour être apatride, vous pouvez ensuite utiliser des services de calcul sans serveur, tels que AWS Lambda ou. AWS Fargate

Outre le remplacement des serveurs, les applications apatrides présentent un autre avantage : elles peuvent évoluer horizontalement, car toutes les ressources informatiques disponibles (telles que les EC2 instances et les AWS Lambda fonctions) peuvent répondre à toutes les demandes.

Avantages du respect de cette bonne pratique : les systèmes conçus pour être sans état sont plus adaptables à la mise à l'échelle horizontale, ce qui permet d'ajouter ou de supprimer des capacités

en fonction des fluctuations du trafic et de la demande. Ils sont également intrinsèquement résilients aux défaillances et offrent flexibilité et agilité dans le cadre du développement d'applications.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : moyen

Directives d'implémentation

Rendre vos applications sans état. Les applications sans état permettent une mise à l'échelle horizontale et tolèrent la défaillance d'un nœud individuel. Analysez et comprenez les composants de votre application qui conservent leur état au sein de l'architecture. Cela vous permet d'évaluer l'impact potentiel de la transition vers une conception sans état. Une architecture sans état découple les données utilisateur et décharge les données de session. Cela permet de mettre à l'échelle chaque composant indépendamment pour répondre à des demandes variables de charge de travail et optimiser l'utilisation des ressources.

Étapes d'implémentation

- Identifiez et comprenez les composants avec état dans votre application.
- Découplez les données en séparant et en gérant les données utilisateur de la logique principale de l'application.
  - Amazon Cognito peut découpler les données utilisateur du code de l'application en utilisant des fonctionnalités telles que les groupes d'identités, les groupes d'utilisateurs et Amazon Cognito Sync.
  - Vous pouvez utiliser AWS Secrets Manager pour découpler les données utilisateur en stockant les secrets dans un emplacement sécurisé et centralisé. Cela signifie que le code de l'application n'a pas besoin de stocker de secrets, ce qui le rend plus sécurisé.
  - Envisagez d'utiliser Amazon S3 pour stocker des données volumineuses non structurées, telles que des images et des documents. Votre application peut récupérer ces données en cas de besoin, évitant ainsi d'avoir à les stocker en mémoire.
  - Utilisez Amazon DynamoDB pour stocker des informations telles que les profils utilisateur. Votre application peut interroger ces données en temps quasi réel.
- Déchargez les données de session dans une base de données, un cache ou des fichiers externes.
  - Amazon ElastiCache, Amazon DynamoDB, Amazon Elastic File System (Amazon) et EFS Amazon MemoryDB sont des exemples de services que vous pouvez utiliser pour AWS décharger des données de session.
- Concevez une architecture sans état après avoir identifié les données d'état et d'utilisateur qui doivent être conservées avec la solution de stockage de votre choix.

Ressources

Bonnes pratiques associées :

- REL11-BP03 Automatiser la guérison sur toutes les couches

Documents connexes :

- L'Amazon Builders' Library : éviter le basculement dans les systèmes distribués
- L'Amazon Builders' Library : éviter les retards de file d'attente insurmontables
- L'Amazon Builders' Library : défis et stratégies de mise en cache
- Bonnes pratiques pour Stateless Web Tier sur AWS

REL05-BP07 Mettre en œuvre des leviers de secours

Les leviers d'urgence sont des processus rapides qui peuvent réduire l'impact sur la disponibilité de votre charge de travail.

Les leviers d'urgence fonctionnent en désactivant, en limitant ou en modifiant le comportement des composants ou des dépendances à l'aide de mécanismes connus et testés. Ils permettent d'atténuer les perturbations de la charge de travail causées par l'épuisement des ressources dû à une augmentation inattendue de la demande et de réduire l'impact des défaillances des composants non stratégiques de votre charge de travail.

Résultat souhaité : en mettant en œuvre des leviers d'urgence, vous pouvez établir des processus dont le fonctionnement a été vérifié pour maintenir la disponibilité des composants essentiels de votre charge de travail. La charge de travail devrait se dégrader de manière appropriée et continuer à remplir ses fonctions stratégiques durant l'activation d'un levier d'urgence. Pour plus de détails sur la dégradation progressive, voir REL05-BP01 Implémenter la dégradation progressive pour transformer les dépendances strictes applicables en dépendances souples.

Anti-modèles courants :

- La défaillance des dépendances non stratégiques a un impact sur la disponibilité de votre charge de travail principale.
- Le comportement des composants stratégiques n'est pas testé ou vérifié lors d'une défaillance d'un composant non stratégique.

- Aucun critère clair et déterministe n'a été défini pour l'activation ou la désactivation d'un levier d'urgence.

Avantages du respect de cette bonne pratique : la mise en œuvre de leviers d'urgence peut améliorer la disponibilité des composants critiques de votre charge de travail en fournissant à vos résolveurs des processus établis pour répondre aux pics de demande inattendus ou aux défaillances liées à des dépendances non critiques.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : moyen

Directives d'implémentation

- Identifier les composants stratégiques de votre charge de travail.
- Concevoir et construire les composants stratégiques de votre charge de travail de manière à ce qu'ils résistent aux défaillances des composants non stratégiques.
- Effectuer des tests pour valider le comportement de vos composants stratégiques en cas de défaillance des composants non stratégiques.
- Définir et surveiller des métriques ou des déclencheurs pertinents pour lancer des procédures de levier d'urgence.
- Définir les procédures (manuelles ou automatisées) qui comprennent le levier d'urgence.

Étapes d'implémentation

- Identifier les composants stratégiques de votre charge de travail.
  - Chaque composant technique de votre charge de travail doit être associé à la fonction commerciale correspondante et classé comme stratégique ou non stratégique. Pour des exemples de fonctionnalités critiques et non critiques d'Amazon, consultez Any Day Can Be Prime Day: How Amazon.com Search Uses Chaos Engineering to Handle Over 84K Requests Per Second.
  - Il s'agit d'une décision à la fois technique et commerciale, qui varie en fonction de l'organisation et de la charge de travail.
- Concevoir et construire les composants stratégiques de votre charge de travail de manière à ce qu'ils résistent aux défaillances des composants non stratégiques.
  - Lors de l'analyse des dépendances, tenez compte de tous les modes de défaillance potentiels et vérifiez que vos mécanismes de levier d'urgence fournissent les fonctionnalités stratégiques aux composants en aval.

- Effectuer des tests pour valider le comportement de vos composants stratégiques pendant l'activation de vos leviers d'urgence.
  - Éviter les comportements bimodaux. Pour plus de détails, voir REL11-BP05 Utiliser la stabilité statique pour empêcher le comportement bimodal.
- Définir et surveiller des métriques pertinentes pour lancer des procédures de levier d'urgence.
  - La recherche des bonnes métriques à surveiller dépend de votre charge de travail. Parmi les métriques, citons la latence ou le nombre de demandes infructueuses à une dépendance.
- Définir les procédures (manuelles ou automatisées) qui comprennent le levier d'urgence.
  - Cela peut inclure des mécanismes tels que le délestage, les demandes de limitation ou la mise en œuvre d'une dégradation appropriée.

Ressources

Bonnes pratiques associées :

- REL05-BP01 Implémenter une dégradation progressive pour transformer les dépendances matérielles applicables en dépendances souples
- REL05-BP02 Demandes d'accélérateur
- REL11-BP05 Utiliser la stabilité statique pour empêcher le comportement bimodal

Documents connexes :

- Automatiser les déploiements sécurisés et sans intervention
- Any Day Can Be Prime Day: How Amazon.com Search Uses Chaos Engineering to Handle Over 84K Requests Per Second

Vidéos connexes :

- AWS re:Invent 2020 : fiabilité, cohérence et confiance grâce à l'immuabilité

Gestion des modifications

Les modifications apportées à votre charge de travail ou à son environnement doivent être anticipées et prises en compte pour assurer un fonctionnement fiable de la charge de travail. Les modifications incluent celles imposées à votre charge de travail telles que les pics de demande, ainsi que celles de l'intérieur comme les déploiements de fonctions et les correctifs de sécurité.

Les sections suivantes expliquent les bonnes pratiques en matière de gestion des modifications :

Rubriques

Surveiller les ressources de charge de travail

Les journaux et les métriques sont des outils puissants qui permettent de mieux comprendre l'état de santé de votre charge de travail. Vous pouvez configurer votre charge de travail pour qu'elle surveille les journaux et les métriques et envoie des notifications lorsque des seuils sont franchis ou que des événements importants se produisent. La surveillance permet à votre charge de travail de reconnaître quand des seuils de faibles performances sont franchis ou quand des défaillances se produisent, afin d'y répondre par une récupération automatique.

La surveillance est essentielle pour s'assurer que vous respectez vos exigences en matière de disponibilité. Votre surveillance doit détecter efficacement les pannes. La pire des pannes est « silencieuse ». La fonctionnalité devient inopérante, mais il est impossible de détecter la panne, sinon indirectement. Vos clients sont informés avant vous. La fonction d'alerte en cas de problèmes est l'une des principales raisons de votre surveillance. Vos alertes doivent être séparées autant que possible de vos systèmes. Si votre interruption de service supprime votre capacité d'alerte, cela rallongera votre période d'interruption.

Chez AWS, nous instrumentons nos applications à plusieurs niveaux. Nous enregistrons la latence, les taux d'erreurs et la disponibilité pour chaque requête, pour toutes les dépendances et pour les opérations clés du processus. Nous enregistrons également des métriques sur le bon fonctionnement. Cela nous permet de voir les problèmes imminents avant qu'ils se produisent. Nous ne prenons pas seulement en compte la latence moyenne. Nous nous concentrons davantage sur

la latence hors norme, comme les 99,9e et 99,99e centiles. En effet, si une requête sur 1 000 ou 10 000 est lente, cela reste une mauvaise expérience. Ainsi, si votre moyenne est acceptable, mais qu'une requête sur 100 entraîne une latence extrême, un problème peut survenir lorsque votre trafic augmente.

La surveillance au sein d'AWS comporte quatre phases distinctes :

1. Génération : surveillance de tous les composants de la charge de travail
2. Agrégation : définition et calcul des métriques
3. Traitement et alarmes en temps réel : envoi de notifications et automatisation des réponses
4. Stockage et analyse

Bonnes pratiques

- REL06-BP01 Surveiller tous les composants de la charge de travail (génération)
- REL06-BP02 Définir et calculer des métriques (agrégation)
- REL06-BP03 Envoyer des notifications (traitement et alarmes en temps réel)
- REL06-BP04 Automatiser les réponses (traitement et alarmes en temps réel)
- REL06-BP05 Analyser les journaux
- REL06-BP06 Passer régulièrement en revue la portée et les métriques de surveillance
- REL06-BP07 Surveiller la traçabilité complète des demandes via votre système

REL06-BP01 Surveiller tous les composants de la charge de travail (génération)

Surveillez les composants de la charge de travail avec Amazon CloudWatch ou des outils tiers. Surveillez les services AWS avec le tableau de bord AWS Health.

Tous les composants de votre charge de travail doivent être surveillés, y compris le côté utilisateur, la logique métier et les niveaux de stockage. Au besoin, définissez des métriques clés, décrivez leur procédure d'extraction des journaux, puis spécifiez des seuils d'invocation pour les événements d'alarme correspondants. Assurez-vous que les métriques sont pertinentes pour les indicateurs clés de performance (KPI) de votre charge de travail, et utilisez des métriques et des journaux pour identifier les signes avant-coureurs de la dégradation du service. Par exemple, une métrique liée aux résultats commerciaux, telle que le nombre de commandes traitées avec succès par minute, peut indiquer des problèmes de charge de travail plus rapidement qu'une métrique technique, telle que

l'utilisation du processeur. Utilisez le tableau de bord AWS Health pour obtenir une vue personnalisée des performances et de la disponibilité des services AWS sous-jacents à vos ressources AWS.

La surveillance dans le cloud offre de nouvelles opportunités. La plupart des fournisseurs de cloud ont développé des hooks personnalisables et peuvent fournir des informations pour vous aider à surveiller plusieurs couches de votre charge de travail. Des services AWS comme Amazon CloudWatch appliquent des algorithmes statistiques et de machine learning pour analyser en continu les métriques des systèmes et des applications, déterminer les points de référence normaux et détecter les anomalies avec une intervention minimale de l'utilisateur. Les algorithmes de détection des anomalies tiennent compte de la saisonnalité et des changements de tendance des métriques.

AWS met à disposition une multitude d'informations de surveillance et de journalisation qui peuvent être utilisées pour définir des métriques spécifiques à la charge de travail et des processus de changement de la demande, et pour adopter des techniques de machine learning, quelle que soit l'expertise en ML.

En outre, surveillez l'ensemble de vos points de terminaison externes afin de vous assurer qu'ils sont indépendants de votre implémentation de base. Cette surveillance active peut être effectuée avec des transactions synthétiques (parfois appelées Canary utilisateurs à ne pas confondre avec les déploiements Canary) qui exécutent régulièrement un certain nombre de tâches courantes effectuées par les clients de la charge de travail. Maintenez ces tâches de courte durée et veillez à ne pas surcharger votre charge de travail pendant les tests. Amazon CloudWatch Synthetics vous permet de créer des scripts Canary synthétiques pour surveiller vos points de terminaison et vos API. Vous pouvez également combiner les nœuds de clients synthétiques Canary avec la console AWS X-Ray pour identifier les scripts Canary synthétiques qui rencontrent des erreurs, des pannes ou des taux de limitation au cours de la période sélectionnée.

Résultat escompté :

Collectez et utilisez des métriques critiques de tous les composants de la charge de travail pour garantir la fiabilité de la charge de travail et une expérience utilisateur optimale. Détecter qu'une charge de travail n'atteint pas les résultats vous permet de déclarer rapidement un sinistre et de vous remettre d'un incident.

Anti-modèles courants :

- Surveillance limitée aux interfaces externes de votre charge de travail.
- Ne pas générer de métriques spécifiques à la charge de travail et s'appuyer uniquement sur les métriques qui vous sont fournies par les services AWS utilisés par votre charge de travail.

- Utiliser uniquement des métriques techniques dans votre charge de travail et ne surveiller aucune métrique liée aux KPI non techniques auxquels la charge de travail contribue.
- S'appuyer sur le trafic de production et de simples surveillances de l'état pour surveiller et évaluer l'état de la charge de travail.

Avantages du respect de cette bonne pratique : la surveillance à tous les niveaux de votre charge de travail vous permet d'anticiper et de résoudre plus rapidement les problèmes dans les composants qui constituent la charge de travail.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : élevé

Directives d'implémentation

1. Activez la journalisation lorsqu'elle est disponible. Les données de surveillance doivent être obtenues à partir de tous les composants des charges de travail. Activez la journalisation supplémentaire, telle que les journaux d'accès S3, et autorisez votre charge de travail à consigner des données qui lui sont spécifiques. Collectez des métriques pour les moyennes du processeur, des E/S réseau et des E/S du disque auprès de services tels qu'Amazon ECS, Amazon EKS, Amazon EC2, Elastic Load Balancing, AWS Auto Scaling et Amazon EMR. Consultez la section Services AWS qui publient des métriques CloudWatch pour obtenir la liste des services AWS qui publient des métriques sur CloudWatch.
2. Passez en revue toutes les métriques par défaut et explorez toutes les lacunes de collecte de données. Chaque service génère des métriques par défaut. La collecte des métriques par défaut vous permet de mieux comprendre les dépendances entre les composants de charge de travail et sur la manière dont la fiabilité et les performances des composants affectent la charge de travail. Vous pouvez également créer et publier vos propres métriques dans CloudWatch à l'aide de la AWS CLI ou d'une API.
3. Évaluez toutes les métriques pour décider celles pour lesquelles envoyer des alertes pour chaque service AWS de votre charge de travail. Vous pouvez choisir de sélectionner un sous-ensemble de métriques qui ont un impact majeur sur la fiabilité de la charge de travail. En vous concentrant sur les métriques et les seuils critiques, vous pouvez affiner le nombre d'alertes et réduire le nombre de faux positifs.
4. Définissez des alertes et le processus de récupération de votre charge de travail après l'invocation de l'alerte. La définition d'alertes vous permet de notifier, de faire remonter et de suivre rapidement les étapes nécessaires pour vous remettre d'un incident et atteindre votre objectif de délai de

reprise (RTO) prescrit. Vous pouvez utiliser des alarmes Amazon CloudWatch pour invoquer des flux de travail automatisés et lancer des procédures de récupération en fonction de seuils définis.

5. Explorez l'utilisation de transactions synthétiques pour collecter des données pertinentes sur l'état des charges de travail. La surveillance synthétique suit les mêmes routes et effectue les mêmes actions qu'un client, ce qui vous permet de vérifier en permanence l'expérience client même lorsque vous n'avez aucun trafic client sur vos charges de travail. En utilisant les transactions synthétiques, vous pouvez découvrir les problèmes avant vos clients.

Ressources

Bonnes pratiques associées:

- REL11-BP03 Automatiser la réparation sur toutes les couches

Documents connexes :

- Premiers pas avec le tableau de bord AWS Health : état de votre compte
- Services AWS qui publient des métriques CloudWatch
- Journaux d'accès de votre Network Load Balancer
- Journaux d'accès pour votre Application Load Balancer
- Accès aux journaux Amazon CloudWatch Logs pour AWS Lambda
- Journalisation des accès au serveur Amazon S
- Activation des journaux d'accès pour votre Classic Load Balancer
- Exporter les données du journal vers Amazon S3
- Installer l'agent CloudWatch sur une instance Amazon EC2
- Publication des métriques personnalisées
- Utilisation des tableaux de bord Amazon CloudWatch
- Utilisation des métriques Amazon CloudWatch
- Utilisation de scripts Canary (Amazon CloudWatch Synthetics)
- Qu'est-ce qu'Amazon CloudWatch Logs ?

Guides de l'utilisateur :

- Création d'un journal de suivi
- Surveillance des métriques de mémoire et de disque pour les instances Linux Amazon EC2

- Utilisation de CloudWatch Logs avec des instances de conteneurs
- Journaux de flux VPC
- Qu'est-ce qu'Amazon DevOps Guru ?
- Qu'est-ce qu AWS X-Ray?

Blogs connexes :

- Débogage avec Amazon CloudWatch Synthetics et AWS X-Ray

Exemples connexes :

- Amazon Builders' Library : Instrumentation des systèmes distribués au profit de la visibilité opérationnelle
- Atelier sur l'observabilité

REL06-BP02 Définir et calculer des métriques (agrégation)

Collectez les métriques et les journaux des composants de votre charge de travail, et calculez des métriques agrégées pertinentes à partir de ces derniers. Ces métriques permettent une observabilité large et approfondie de votre charge de travail et peuvent améliorer de manière significative votre posture de résilience.

L'observabilité ne se limite pas à collecter des métriques à partir des composants de votre charge de travail et à être en mesure de les visualiser et d'émettre des alertes à leur sujet. Elle permet de bénéficier d'une compréhension globale du comportement de votre charge de travail. Ces informations comportementales proviennent de tous les composants de vos charges de travail, notamment des services cloud dont elles dépendent, des journaux bien conçus et des métriques. Ces données vous permettent de surveiller le comportement de votre charge de travail dans son ensemble et de comprendre de manière détaillée l'interaction de chaque composant avec chaque unité de travail.

Résultat escompté :

- Vous collectez les journaux des composants de votre charge de travail et des dépendances des services AWS, et vous les publiez dans un emplacement central où ils peuvent être facilement consultés et traités.

- Vos journaux contiennent des horodatages fidèles et précis.
- Vos journaux contiennent des informations pertinentes sur le contexte du traitement, telles qu'un identifiant de suivi, un identifiant d'utilisateur ou de compte et une adresse IP distante.
- Vous créez des métriques agrégées à partir de vos journaux qui représentent le comportement de votre charge de travail d'un point de vue général.
- Vous pouvez interroger vos journaux agrégés pour obtenir des informations approfondies et pertinentes sur votre charge de travail et identifier les problèmes réels et potentiels.

Anti-modèles courants :

- Vous ne collectez pas de métriques ou de journaux pertinents à partir des instances de calcul sur lesquelles vos charges de travail s'exécutent ou des services cloud qu'elles utilisent.
- Vous négligez la collecte des journaux et métriques liés à vos indicateurs de performances clés (KPI) métier.
- Vous analysez la télémétrie liée à la charge de travail de manière isolée, sans agrégation ni corrélation.
- Vous laissez les métriques et les journaux expirer trop rapidement, ce qui entrave l'analyse des tendances et l'identification des problèmes récurrents.

Avantages du respect de ces bonnes pratiques : vous pouvez détecter davantage d'anomalies et corréler les événements et les métriques entre les différents composants de votre charge de travail. Vous pouvez créer des informations exploitables à partir des composants de votre charge de travail en vous basant sur les informations contenues dans les journaux, qui ne sont souvent pas disponibles dans les métriques seules. Vous pouvez déterminer plus rapidement les causes des défaillances en interrogeant vos journaux à grande échelle.

Niveau d'exposition au risque si ces bonnes pratiques ne sont pas respectées : élevé

Directives d'implémentation

Identifiez les sources des données de télémétrie pertinentes pour vos charges de travail et leurs composants. Ces données proviennent non seulement des composants qui publient des métriques, tels que votre système d'exploitation (OS) et les environnements d'exécution d'applications tels que Java, mais également des journaux des applications et des services cloud. Par exemple, les serveurs Web enregistrent généralement chaque demande avec des informations détaillées telles que l'horodatage, la latence de traitement, l'ID utilisateur, l'adresse IP distante, le chemin et la chaîne

de requête. Le niveau de détail de ces journaux vous permet d'effectuer des requêtes détaillées et de générer des métriques qui n'auraient peut-être pas été disponibles autrement.

Collectez les métriques et les journaux à l'aide d'outils et de processus appropriés. Les journaux générés par les applications exécutées sur une instance Amazon EC2 peuvent être collectés par un agent tel que l'agent Amazon CloudWatch et publiés dans un service de stockage central tel qu'Amazon CloudWatch Logs. Les services de calcul gérés par AWS tels qu'AWS Lambda et Amazon Elastic Container Service publient automatiquement les journaux dans CloudWatch Logs pour vous. Activez la collecte de journaux pour les services de stockage et de traitement AWS utilisés par vos charges de travail, tels qu'Amazon CloudFront, Amazon S3, Elastic Load Balancing et Amazon API Gateway.

Enrichissez vos données de télémétrie avec des dimensions qui peuvent vous aider à mieux identifier les modèles comportementaux et à isoler les problèmes corrélés à des groupes de composants associés. Une fois ces dimensions ajoutées, vous pouvez observer le comportement des composants de manière plus détaillée, détecter les défaillances corrélées et prendre les mesures correctives appropriées. La zone de disponibilité, l'ID d'instance EC2 et l'ID de pod ou de tâche de conteneur sont des exemples de dimensions utiles.

Une fois que vous avez collecté les métriques et les journaux, vous pouvez rédiger des requêtes et générer des métriques agrégées à partir de ces éléments pour fournir des informations exploitables utiles sur les comportements normaux et anormaux. Par exemple, vous pouvez utiliser Amazon CloudWatch Logs Insights pour dériver des métriques personnalisées des journaux de vos applications, Amazon CloudWatch Metrics Insights pour interroger vos métriques à grande échelle, Amazon CloudWatch Container Insights pour collecter, agréger et résumer les métriques et les journaux de vos applications et microservices conteneurisés, ou Amazon CloudWatch Lambda Insights si vous utilisez des fonctions AWS Lambda. Pour créer une métrique de taux d'erreur agrégé, vous pouvez incrémenter un compteur chaque fois qu'une réponse ou un message d'erreur est détecté dans les journaux de vos composants ou calculer la valeur agrégée d'une métrique de taux d'erreur existante. Vous pouvez utiliser ces données pour générer des histogrammes illustrant le comportement de queue, par exemple les demandes ou les processus les moins performants. Vous pouvez également analyser ces données en temps réel pour détecter des modèles anormaux à l'aide de solutions telles que la détection des anomalies de CloudWatch Logs. Il est possible de placer ces informations exploitables dans des tableaux de bord afin de les organiser en fonction de vos besoins et préférences.

L'interrogation des journaux peut vous aider à comprendre comment des demandes spécifiques ont été traitées par les composants de votre charge de travail et à révéler les modèles de demandes

ou tout autre contexte ayant un impact sur la résilience de votre charge de travail. Il peut être utile d'étudier et de préparer des requêtes à l'avance, en fonction de votre connaissance des comportements de vos applications et des autres composants, afin de pouvoir les utiliser plus facilement en cas de besoin. Par exemple, CloudWatch Logs Insights vous permet de rechercher et d'analyser de façon interactive les données de vos journaux dans CloudWatch Logs. Vous pouvez également utiliser Amazon Athena pour interroger les journaux provenant de plusieurs sources, y compris de nombreux services AWS, à l'échelle du pétaoctet.

Lorsque vous définissez une politique de conservation des journaux, tenez compte de la valeur des journaux d'historique. Les journaux d'historique peuvent aider à identifier les modèles d'utilisation et comportementaux à long terme, les régressions et les améliorations des performances de votre charge de travail. Les journaux définitivement supprimés ne peuvent pas être analysés ultérieurement. Toutefois, la valeur des journaux d'historique tend à diminuer sur de longues périodes. Choisissez une politique capable d'équilibrer vos besoins de manière appropriée et conforme à toutes les exigences légales ou contractuelles auxquelles vous pourriez être soumis.

Étapes d'implémentation

1. Choisissez des mécanismes de collecte, de stockage, d'analyse et d'affichage de vos données d'observabilité.
2. Installez et configurez des collecteurs de métriques et de journaux sur les composants appropriés de votre charge de travail (par exemple, sur les instances Amazon EC2 et dans les conteneurs sidecar). Configurez ces collecteurs pour qu'ils redémarrent automatiquement s'ils s'arrêtent de façon inattendue. Activez la mise en mémoire tampon du disque ou de la mémoire pour les collecteurs afin que les échecs de publication temporaires n'aient pas d'impact sur vos applications ou n'entraînent aucune perte de données.
3. Activez la journalisation sur les services AWS que vous utilisez dans le cadre de vos charges de travail et transférez ces journaux au service de stockage que vous avez sélectionné si nécessaire. Reportez-vous au guide d'utilisateur ou au manuel du développeur des services respectifs pour obtenir des instructions détaillées.
4. Définissez les métriques opérationnelles pour vos charges de travail en fonction de vos données de télémétrie. Elles peuvent être basées sur des métriques directes émises par les composants de votre charge de travail, qui peuvent inclure des métriques liées aux KPI métier, ou sur les résultats de calculs agrégés tels que des sommes, des taux, des centiles ou des histogrammes. Calculez ces métriques à l'aide de votre analyseur de journaux et placez-les dans des tableaux de bord, de façon appropriée.

5. Préparez des requêtes de journaux appropriées pour analyser les composants de la charge de travail, les demandes ou le comportement des transactions selon les besoins.
6. Définissez et activez une politique de conservation des journaux pour les journaux de vos composants. Supprimez régulièrement les journaux lorsqu'ils sont plus anciens que la politique ne l'autorise.

Ressources

Bonnes pratiques associées:

- REL06-BP01 Surveiller tous les composants de la charge de travail (génération)
- REL06-BP03 Envoyer des notifications (traitement et alarmes en temps réel)
- REL06-BP04 Automatiser les réponses (traitement et alarmes en temps réel)
- REL06-BP05 Analyser les journaux
- REL06-BP06 Passer régulièrement en revue la portée et les métriques de surveillance
- REL06-BP07 Surveiller le suivi de bout en bout des demandes via votre système

Documentation connexe :

- Fonctionnement d'Amazon CloudWatch
- Amazon Managed Prometheus
- Amazon Managed Grafana
- Analyse des données de journaux avec CloudWatch Logs Insights
- Amazon CloudWatch Lambda Insights
- Amazon CloudWatch Container Insights
- Interrogation de vos métriques avec CloudWatch Metrics Insights
- AWS Distro pour Open Telemetry
- Exemples de requêtes pour Amazon CloudWatch Logs Insights
- Débogage avec Amazon CloudWatch Synthetics et AWS X-Ray
- Recherche et filtrage des données de journaux
- Envoi des journaux directement à Amazon S3
- Amazon Builders' Library : Instrumentation des systèmes distribués au profit de la visibilité opérationnelle

Ateliers connexes:

- Un atelier sur l'observabilité

Outils associés :

- AWS Distro for OpenTelemetry (GitHub)

REL06-BP03 Envoyer des notifications (traitement et alarmes en temps réel)

Lorsque les organisations détectent des problèmes potentiels, elles envoient des notifications et des alertes en temps réel au personnel et aux systèmes appropriés afin de résoudre rapidement et efficacement ces problèmes.

Résultat escompté : il est possible d'obtenir des réponses rapides aux événements opérationnels en configurant des alarmes pertinentes basées sur les métriques de service et d'application. Lorsque les seuils d'alarme sont dépassés, le personnel et les systèmes appropriés sont avertis afin de résoudre les problèmes sous-jacents.

Anti-modèles courants :

- Vous configurez les alarmes avec un seuil trop élevé, ce qui fait échouer l'envoi des notifications vitales.
- Vous configurez les alarmes avec un seuil trop bas, ce qui empêche la prise en compte des alertes importantes à cause du bruit généré par un trop grand nombre de notifications.
- Vous ne mettez pas à jour les alarmes et leur seuil en cas de changement d'utilisation.
- Pour les alarmes qu'il est préférable de traiter par des actions automatisées, vous envoyez la notification au personnel au lieu de générer l'action automatisée, ce qui entraîne l'envoi d'un trop grand nombre de notifications.

Avantages du respect de cette bonne pratique : en envoyant des notifications et des alertes en temps réel au personnel et aux systèmes appropriés, vous pouvez détecter rapidement les problèmes et réagir rapidement face aux incidents opérationnels.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : élevé

Directives d'implémentation

Les charges de travail doivent être équipées d'un système de traitement et d'avertissement en temps réel afin d'améliorer la détectabilité des problèmes susceptibles d'affecter la disponibilité de l'application et de déclencher une réponse automatique. Les organisations peuvent procéder au traitement et à l'avertissement en temps réel en créant des alertes avec des métriques définies afin de recevoir des notifications chaque fois que des événements importants se produisent ou qu'une métrique dépasse un seuil.

Amazon CloudWatch vous permet de créer des alarmes métriques et composites à l'aide d'alarmes CloudWatch basées sur un seuil statique, la détection d'anomalies et d'autres critères. Pour plus de détails sur les types d'alarmes que vous pouvez configurer à l'aide de CloudWatch, consultez la section sur les alarmes de la documentation de CloudWatch.

Vous pouvez construire des vues personnalisées des métriques et des alertes de vos ressources AWS pour vos équipes en utilisant les tableaux de bord CloudWatch. Les pages d'accueil personnalisables de la console CloudWatch vous permettent de surveiller vos ressources dans une vue unique sur plusieurs régions.

Les alarmes peuvent effectuer une ou plusieurs actions, comme l'envoi d'une notification à un sujet Amazon SNS, l'exécution d'une action Amazon EC2 ou d'une action Amazon EC2 Auto Scaling, ou la création d'un OpsItem ou d'un incident dans AWS Systems Manager.

Amazon CloudWatch utilise Amazon SNS pour envoyer des notifications lorsque l'alarme change d'état, ce qui permet de transmettre les messages des diffuseurs de publication (producteurs) aux abonnés (consommateurs). Pour plus de détails sur la configuration des notifications Amazon SNS, consultez Configuration d'Amazon SNS.

CloudWatch envoie des événements à EventBridge chaque fois qu'une alerte CloudWatch est créée, mise à jour, supprimée ou change d'état. Vous pouvez utiliser EventBridge avec ces événements pour créer des règles qui exécutent des actions, comme vous avertir chaque fois que l'état d'une alarme change ou déclencher automatiquement des événements sur votre compte à l'aide de l'automatisation Systems Manager.

Restez informé avec AWS Health. AWS Health est la source d'informations faisant autorité sur l'état de vos ressources AWS Cloud. Utilisez AWS Health pour être informé de tout événement de service confirmé afin que vous puissiez rapidement prendre des mesures pour atténuer tout impact. Créez des notifications d'événements AWS Health spécialement adaptées aux e-mails et aux canaux de discussion via Notifications des utilisateurs AWS et intégrez-les de manière programmatique à

vos outils de surveillance et d'alerte via Amazon EventBridge. Si vous utilisez AWS Organizations, regroupez les événements AWS Health sur l'ensemble des comptes.

Quand devriez-vous utiliser EventBridge ou Amazon SNS?

EventBridge et Amazon SNS peuvent être utilisés pour développer des applications pilotées par des événements. Votre choix dépendra de vos besoins spécifiques.

Amazon EventBridge est recommandé lorsque vous souhaitez créer une application qui réagit aux événements de vos propres applications, des applications SaaS et des services AWS. EventBridge est le seul service basé sur des événements qui s'intègre directement à des partenaires SaaS tiers. EventBridge ingère également automatiquement les événements provenant de plus de 200 services AWS sans que les développeurs n'aient à créer de ressources dans leur compte.

EventBridge utilise une structure définie basée sur JSON pour les événements et vous aide à créer des règles qui s'appliquent à l'ensemble du corps de l'événement afin de sélectionner les événements à transférer vers une cible. EventBridge prend actuellement en charge plus de 20 services AWS en tant que cibles, notamment AWS Lambda, Amazon SQS, Amazon SNS, Amazon Kinesis Data Streams et Amazon Data Firehose.

Amazon SNS est recommandé pour les applications qui nécessitent un niveau de diffusion élevé (des milliers, voire des millions de points de terminaison). Il est courant d'observer que les clients utilisent Amazon SNS comme cible pour leur règle afin de filtrer les événements dont ils ont besoin et de les diffuser sur plusieurs points de terminaison.

Les messages ne sont pas structurés et peuvent être de n'importe quel format. Amazon SNS prend en charge le transfert de messages vers six types de cibles différents, notamment Lambda, Amazon SQS, les points de terminaison HTTP/S, les SMS, le push mobile et le courrier électronique. Amazon SNS possède une latence typique inférieure à 30 millisecondes. Un large éventail de services AWS envoie des messages Amazon SNS en configurant le service dans cet objectif (plus de 30, y compris Amazon EC2, Amazon S3 et Amazon RDS).

Étapes d'implémentation

1. Créez une alarme à l'aide des alarmes Amazon CloudWatch.
    a. Une alarme de métrique surveille une seule métrique CloudWatch ou une expression qui dépend de métriques CloudWatch. L'alarme déclenche une ou plusieurs actions en fonction de la valeur de la métrique ou de l'expression par rapport à un seuil sur un certain nombre d'intervalles de temps. L'action peut être l'envoi d'une notification à un sujet Amazon SNS,

l'exécution d'une action Amazon EC2 ou d'une action Amazon EC2 Auto Scaling ou la création d'un OpsItem ou d'un incident dans AWS Systems Manager.

b. Une alarme composite est une expression de règle qui prend en compte les conditions d'alarme des autres alarmes que vous avez créées. L'alarme composite ne passe en état d'alarme que si toutes les conditions de la règle sont satisfaites. Les alarmes spécifiées dans l'expression de règle d'une alarme composite peuvent inclure des alarmes de métrique et des alarmes composites supplémentaires. Les alertes composites peuvent envoyer des notifications Amazon SNS lorsqu'elles changent d'état et peuvent créer des OpsItems de Systems Manager ou des incidents lorsqu'elles passent à l'état d'alarme, mais ne peuvent pas effectuer d'actions Amazon EC2 ou Auto Scaling.

2. Configurer les notifications Amazon SNS. Lorsque vous créez une alarme CloudWatch, vous pouvez inclure une rubrique Amazon SNS pour envoyer une notification lorsque l'alarme change d'état.
3. Créez des règles dans EventBridge qui correspondent aux alarmes CloudWatch spécifiées. Chaque règle prend en charge plusieurs cibles, y compris des fonctions Lambda. Par exemple, vous pouvez définir une alarme qui se déclenche lorsque l'espace disque disponible est insuffisant, ce qui déclenche une fonction Lambda par le biais d'une règle EventBridge permettant de libérer de l'espace. Pour plus de détails sur les cibles EventBridge, consultez la section Cibles EventBridge.

Ressources

Bonnes pratiques Well-Architected connexes:

- REL06-BP01 Surveiller tous les composants de la charge de travail (génération)
- REL06-BP02 Définir et calculer des métriques (agrégation)
- REL12-BP01 Utiliser des playbooks pour enquêter sur les causes des défaillances

Documents connexes:

- Amazon CloudWatch
- Aperçu des journaux CloudWatch
- Utilisation d'alarmes Amazon CloudWatch
- Utilisation des tableaux de bord Amazon CloudWatch
- Utilisation des métriques Amazon CloudWatch

- Configuration des notifications Amazon SNS
- Détection des anomalies CloudWatch
- Protection des données CloudWatch Logs
- Amazon EventBridge
- Amazon Simple Notification Service

Vidéos connexes:

- reinvent 2022 observability videos
- AWS re:Invent 2022 - Observability best practices at Amazon

Exemples connexes:

- Un atelier sur l'observabilité
- Amazon EventBridge to AWS Lambda with feedback control by Amazon CloudWatch Alarms

REL06-BP04 Automatiser les réponses (traitement et alarmes en temps réel)

Utilisez l'automatisation pour agir en cas de détection d'événement, par exemple, pour remplacer les composants défectueux.

Un traitement automatique en temps réel des alarmes est mis en œuvre afin que les systèmes puissent prendre rapidement des mesures correctives et tenter d'éviter les pannes ou une dégradation du service lorsque les alarmes se déclenchent. Les réponses automatisées aux alarmes peuvent inclure le remplacement des composants défaillants, l'ajustement de la capacité de calcul, la redirection du trafic vers des hôtes, des zones de disponibilité ou d'autres régions en bonne santé, et la notification des opérateurs.

Résultat escompté : les alarmes en temps réel sont identifiées et le traitement automatique des alarmes est configuré pour invoquer les actions appropriées prises afin de maintenir les objectifs de niveau de service et les contrats de niveau de service (SLA). L'automatisation peut aller de l'autoréparation de composants individuels au basculement complet du site.

Anti-modèles courants :

- Pas d'inventaire ou de catalogue clair des principales alarmes en temps réel.
- Aucune réponse automatique aux alarmes critiques (par exemple, lorsque la capacité de calcul est presque épuisée, une mise à l'échelle automatique se produit).
- Réponses aux alarmes contradictoires.
- Pas de procédure opérationnelle standard (SOP) que les opérateurs doivent suivre lorsqu'ils reçoivent des notifications d'alerte.
- Pas de surveillance des modifications de configuration, alors que des changements de configuration non détectés peuvent entraîner des temps d'arrêt pour les charges de travail.
- Pas de stratégie pour annuler les modifications de configuration involontaires.

Avantages du respect de cette bonne pratique : l'automatisation du traitement des alarmes peut améliorer la résilience du système. Le système prend automatiquement des mesures correctives, réduisant ainsi les activités manuelles qui nécessitent des interventions humaines sujettes aux erreurs. L'exécution de la charge de travail permet d'atteindre les objectifs de disponibilité et de réduire les interruptions de service.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : moyen

Directives d'implémentation

Pour gérer efficacement les alertes et automatiser leur réponse, classez les alertes en fonction de leur criticité et de leur impact, documentez les procédures de réponse et planifiez les réponses avant de classer les tâches.

Identifiez les tâches nécessitant des actions spécifiques (souvent détaillées dans les runbooks) et examinez tous les runbooks et playbooks pour déterminer les tâches qui peuvent être automatisées. Si les actions peuvent être définies, alors elles sont souvent automatisables. Si elles ne le sont pas, documentez les étapes manuelles dans une SOP et formez les opérateurs à ces étapes. Remettez continuellement en question les processus manuels pour trouver des opportunités d'automatisation où vous pouvez établir et maintenir un plan d'automatisation des réponses aux alertes.

Étapes d'implémentation

1. Créez un inventaire des alarmes : pour obtenir une liste de toutes les alarmes, vous pouvez utiliser la AWS CLI utilisant la commande Amazon CloudWatch `describe-alarms`. Selon le nombre d'alarmes que vous avez configurées, vous devrez peut-être utiliser la pagination pour récupérer un sous-ensemble d'alarmes pour chaque appel, ou bien vous pouvez utiliser l'AWS SDK pour obtenir les alarmes à l'aide d'un appel d'API.

2. Documentez toutes les actions d'alarme : mettez à jour un runbook avec toutes les alarmes et leurs actions, qu'elles soient manuelles ou automatisées. AWS Systems Manager fournit des runbooks prédéfinis. Pour plus d'informations sur les runbooks, consultez Travailler avec des runbooks. Pour plus de détails sur la façon d'afficher le contenu du runbook, consultez Afficher le contenu du runbook.
3. Configurez et gérez les actions d'alarme : pour toutes les alarmes nécessitant une action, spécifiez l'action automatisée à l'aide de CloudWatch SDK. Par exemple, vous pouvez modifier automatiquement l'état de vos instances Amazon EC2 basées sur une alarme CloudWatch en créant des actions sur l'alarme et en les activant ou désactivant.

   Vous pouvez également utiliser Amazon EventBridge pour automatiser vos événements système, tels que des problèmes de disponibilité d'application ou des modifications de ressources. Vous pouvez créer des règles pour indiquer quels événements vous intéressent et les actions à effectuer quand un événement correspond à une règle. Les actions qui peuvent être initiées automatiquement incluent l'appel d'une fonction AWS Lambda, l'invocation d'Amazon EC2Run Command, le transfert de l'événement à Amazon Kinesis Data Streams et la visualisation d'Automate Amazon EC2 à l'aide d'EventBridge.
4. Procédures opérationnelles standard (SOP) : en fonction des composants de votre application, AWS Resilience Hub recommande plusieurs modèles de SOP. Vous pouvez utiliser ces SOP pour documenter tous les processus qu'un opérateur doit suivre en cas d'alerte. Vous pouvez également construire une SOP basée sur les recommandations du Resilience Hub, où vous avez besoin d'une application Resilience Hub associée à une politique de résilience, ainsi que d'une évaluation historique de la résilience par rapport à cette application. Les recommandations de SOP sont générées par l'évaluation de la résilience.

   Resilience Hub travaille avec Systems Manager pour automatiser les étapes de vos SOP en fournissant un certain nombre de documents SSM que vous pouvez utiliser comme base pour ces SOP. Par exemple, Resilience Hub peut recommander une procédure SOP pour ajouter de l'espace disque sur la base d'un document d'automatisation SSM existant.
5. Effectuer des action automatisées à l'aide d'Amazon DevOps Guru : vous pouvez utiliser Amazon DevOps Guru pour surveiller automatiquement les ressources d'applications pour détecter les comportements anormaux et fournir des recommandations ciblées afin d'accélérer l'identification des problèmes et de réduire les temps de remédiation. Avec DevOps Guru, vous pouvez surveiller les flux de données opérationnelles en temps quasi réel à partir de plusieurs sources, notamment les métriques Amazon CloudWatch, AWS Config, AWS CloudFormation et AWS X-Ray. Vous pouvez également utiliser DevOps Guru pour créer automatiquement des OpsiTems dans OpsCenter et envoyer des événements à EventBridge pour une automatisation supplémentaire.

Ressources

Bonnes pratiques associées:

- REL06-BP01 Surveiller tous les composants de la charge de travail (génération)
- REL06-BP02 Définir et calculer des métriques (agrégation)
- REL06-BP03 Envoyer des notifications (traitement et alarmes en temps réel)
- REL08-BP01 Utiliser des runbooks pour les activités standard telles que le déploiement

Documents connexes :

- Automatisation AWS Systems Manager
- Création d'une règle EventBridge qui se déclenche en cas d'événement provenant d'une ressource AWS
- Un atelier sur l'observabilité
- Amazon Builders' Library : Instrumentation des systèmes distribués au profit de la visibilité opérationnelle
- Qu'est-ce qu'Amazon DevOps Guru ?
- Utilisation des documents d'automatisation (playbooks)

Vidéos connexes :

- AWS re:Invent 2022 - Observability best practices at Amazon
- AWS re:Invent 2020: Automate anything with AWS Systems Manager
- Introduction à AWS Resilience Hub
- Créez des systèmes de tickets personnalisés pour les notifications Amazon DevOps Guru
- Activez l'agrégation d'informations sur plusieurs comptes avec Amazon DevOps Guru

Exemples connexes:

- Atelier Amazon CloudWatch et Systems Manager

REL06-BP05 Analyser les journaux

Collectez les fichiers journaux et les historiques de métriques, puis analysez-les pour obtenir des informations plus générales sur les tendances et la charge de travail.

Amazon CloudWatch Logs Insights prend en charge un langage de requête simple et puissant que vous pouvez utiliser pour analyser les données des journaux. Amazon CloudWatch Logs prend également en charge les abonnements qui permettent aux données de circuler en toute transparence vers Amazon S3 où vous pouvez utiliser Amazon Athena pour interroger les données. Il prend également en charge les requêtes dans une grande variétés de formats. Pour plus d'informations, consultez Formats de données et SerDes pris en charge dans le Guide de l'utilisateur Amazon Athena. Pour analyser des ensembles de fichiers journaux volumineux, vous pouvez exécuter un cluster Amazon EMR pour exécuter des analyses à l'échelle du pétaoctet.

Il existe divers outils fournis par les partenaires AWS et les tiers qui permettent l'agrégation, le traitement, le stockage et l'analyse. Parmi ces outils figurent New Relic, Splunk, Loggly, Logstash, CloudHealth et Nagios. Cependant, la génération en dehors du système et des journaux d'applications est propre à chaque fournisseur de cloud, et généralement, spécifique à chaque service.

Un partie souvent négligée de la surveillance des processus concerne la gestion des données. Vous devez déterminer les exigences de rétention des données de surveillance, puis appliquer des stratégies de cycle de vie en conséquence. Amazon S3 prend en charge la gestion du cycle de vie au niveau du compartiment S3. Cette gestion du cycle de vie peut être appliquée différemment à d'autres chemins dans le compartiment. Vers la fin du cycle de vie, vous pouvez transférer des données dans Amazon Glacier pour un stockage à long terme, puis les laisser expirer à la fin de la période de conservation. La classe de stockage S3 Intelligent-Tiering est conçue pour optimiser les coûts en transférant automatiquement les données vers le niveau d'accès le plus économique, sans impact sur les performances ni surcharge opérationnelle.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : moyen

Directives d'implémentation

- CloudWatch Logs Insights vous permet de rechercher et d'analyser de façon interactive les données de vos journaux dans Amazon CloudWatch Logs.
  - Analyse des données de journaux avec CloudWatch Logs Insights
  - Exemples de requêtes pour Amazon CloudWatch Logs Insights

- Utilisez Amazon CloudWatch Logs pour envoyer des journaux vers Amazon S3 où vous pouvez les utiliser ou utiliser Amazon Athena pour interroger les données.
  - Comment analyser les journaux d'accès au serveur Amazon S3 à l'aide d'Athena ?
    - Créez une stratégie de cycle de vie S3 pour votre compartiment de journaux d'accès au serveur. Configurez la stratégie de cycle de vie pour supprimer périodiquement les fichiers journaux. Cela permet de réduire la quantité de données analysées par Athena pour chaque requête.
      - Comment créer la stratégie de cycle de vie d'un compartiment S3 ?

Ressources

Documents connexes :

- Exemples de requêtes pour Amazon CloudWatch Logs Insights
- Analyse des données de journaux avec CloudWatch Logs Insights
- Débogage avec Amazon CloudWatch Synthetics et AWS X-Ray
- Comment créer la stratégie de cycle de vie d'un compartiment S3 ?
- Comment analyser les journaux d'accès au serveur Amazon S3 à l'aide d'Athena ?
- Un atelier sur l'observabilité
- Amazon Builders' Library : Instrumentation des systèmes distribués au profit de la visibilité opérationnelle

REL06-BP06 Passer régulièrement en revue la portée et les métriques de surveillance

Passez fréquemment en revue la manière dont la surveillance de la charge de travail est mise en œuvre et mettez-la à jour au fur et à mesure que votre charge de travail et son architecture évoluent. Des audits réguliers de votre surveillance permettent de réduire le risque de négliger ou d'omettre des indicateurs de panne et contribuent à aider votre charge de travail à atteindre ses objectifs de disponibilité.

Un suivi efficace s'appuie sur des métriques métier clés, qui évoluent en fonction des priorités de votre entreprise. Votre processus d'examen de la surveillance doit mettre l'accent sur les indicateurs de niveau de service (SLI) et incorporer des informations exploitables provenant de votre infrastructure, de vos applications, de vos clients et de vos utilisateurs.

Résultat escompté : vous disposez d'une stratégie de surveillance efficace qui est régulièrement revue et mise à jour, ainsi qu'après tout événement ou changement important. Vous vérifiez que les indicateurs d'intégrité clés des applications restent pertinents au fur et à mesure de l'évolution de votre charge de travail et de vos exigences professionnelles.

Anti-modèles courants :

- Vous collectez uniquement les métriques par défaut.
- Vous configurez une stratégie de surveillance, mais vous ne la passez jamais en revue.
- Vous ne remettez pas en question la surveillance lorsque des modifications majeures sont déployées.
- Vous vous fiez à des métriques obsolètes pour déterminer l'état de la charge de travail.
- Vos équipes d'exploitation sont submergées d'alertes faussement positives en raison de métriques et de seuils obsolètes.
- Vous ne bénéficiez pas de l'observabilité des composants d'application qui ne sont pas surveillés.
- Vous vous concentrez uniquement sur des métriques techniques de bas niveau et excluez les métriques métier de votre surveillance.

Avantages liés au respect de cette bonne pratique : lorsque vous passez régulièrement en revue votre surveillance, vous pouvez anticiper les problèmes potentiels et vérifier que vous êtes capable de les détecter. Cela vous permet également de découvrir des zones d'ombre que vous auriez pu manquer lors d'examens antérieurs, ce qui améliore encore votre capacité à détecter les problèmes.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : moyen

Directives d'implémentation

Passez en revue les métriques et la portée de la surveillance au cours de votre processus d'examen de l'état de préparation opérationnelle (ORR). Effectuez des examens périodiques de l'état de préparation opérationnelle selon un calendrier cohérent afin d'évaluer s'il existe des écarts entre votre charge de travail actuelle et la surveillance que vous avez configurée. Établissez une fréquence régulière d'examen des performances opérationnelles et de partage des connaissances afin d'améliorer votre capacité à obtenir de meilleures performances de la part de vos équipes d'exploitation. Confirmez ou non que les seuils d'alerte existants sont toujours adéquats et vérifiez les situations dans lesquelles les équipes d'exploitation reçoivent des alertes faussement positives ou ne surveillent pas les aspects de l'application qui devraient être surveillés.

Le cadre d'analyse de résilience fournit des conseils utiles qui peuvent vous aider à naviguer dans le processus. L'objectif de ce cadre est d'identifier les modes de défaillance potentiels et les contrôles préventifs et correctifs que vous pouvez utiliser pour atténuer leur impact. Ces connaissances peuvent vous aider à identifier les métriques et les événements appropriés à surveiller pour émettre des alertes.

Étapes d'implémentation

1. Planifiez et effectuez des vérifications régulières des tableaux de bord de charge de travail. Vous pouvez avoir des cadences différentes selon la profondeur à laquelle vous inspectez.
2. Inspectez les tendances dans les métriques. Comparez les valeurs des métriques aux valeurs historiques pour voir si des tendances peuvent indiquer que quelque chose doit faire l'objet d'une enquête. Cela peut être une augmentation de la latence, une diminution de la fonction principale de l'entreprise ou une augmentation des réponses aux échecs.
3. Recherchez des valeurs aberrantes et des anomalies dans vos métriques, qui peuvent être masquées par des moyennes ou des médianes. Observez les maximales et les minimales sur une période donnée et étudiez les causes des observations qui figurent loin des normales attendues. Au fur et à mesure que vous éliminez ces causes, vous pouvez resserrer les limites des métriques attendues en réponse à l'amélioration de la cohérence des performances de votre charge de travail.
4. Recherchez des changements importants de comportement. Un changement immédiat de quantité ou de direction d'une métrique peut indiquer une modification de l'application ou des facteurs externes, dont le suivi peut nécessiter l'ajout de métriques supplémentaires.
5. Vérifiez si la stratégie de surveillance actuelle reste pertinente pour l'application. Sur la base d'une analyse des incidents précédents (ou du cadre d'analyse de résilience), déterminez si d'autres aspects de l'application devraient être incorporés dans la portée de la surveillance.
6. Passez en revue vos métriques de surveillance des utilisateurs réels (RUM) pour déterminer s'il existe des lacunes dans la couverture des fonctionnalités de l'application.
7. Passez en revue votre processus de gestion des modifications. Mettez à jour vos procédures, si nécessaire, pour inclure une étape d'analyse de surveillance à effectuer avant d'approuver une modification.
8. Mettez en œuvre un examen de surveillance dans le cadre de votre examen de l'état de préparation opérationnelle et de vos processus de correction des erreurs.

Ressources

Bonnes pratiques associées

- REL06-BP01 Surveiller tous les composants de la charge de travail (génération)
- REL06-BP02 Définir et calculer des métriques (agrégation)
- REL06-BP07 Surveiller le suivi de bout en bout des demandes via votre système
- REL12-BP02 Effectuer une analyse post-incident
- REL12-BP06 Organiser régulièrement des tests de simulation de panne

Documents connexes :

- Pourquoi mettre en place la correction des erreurs (COE)
- Utilisation des tableaux de bord Amazon CloudWatch
- Création de tableaux de bord pour une visibilité opérationnelle
- Modèles de résilience multi-AZ avancés – Défaillances grises
- Exemples de requêtes pour Amazon CloudWatch Logs Insights
- Débogage avec Amazon CloudWatch Synthetics et AWS X-Ray
- Un atelier sur l'observabilité
- Amazon Builders' Library : Instrumentation des systèmes distribués au profit de la visibilité opérationnelle
- Utilisation des tableaux de bord Amazon CloudWatch
- AWS Bonnes pratiques en matière d'observabilité
- Cadre d'analyse de résilience
- Cadre d'analyse de résilience – Observabilité
- Examen de l'état de préparation opérationnelle – ORR

REL06-BP07 Surveiller la traçabilité complète des demandes via votre système

Suivez les demandes au fur et à mesure qu'elles sont traitées dans les composants du service afin que les équipes produits puissent plus facilement analyser et résoudre les problèmes et améliorer les performances.

Résultat escompté : les charges de travail dotées d'un suivi complet de tous les composants sont faciles à déboguer, ce qui améliore le temps moyen de résolution (MTTR) des erreurs et la latence en simplifiant la découverte des causes principales. Le traçage de bout en bout réduit le temps nécessaire à la découverte des composants concernés et à l'analyse détaillée des causes profondes des erreurs ou de la latence.

Anti-modèles courants :

- Le traçage est utilisé pour certains composants, mais pas pour tous. Par exemple, sans traçage pour AWS Lambda, les équipes risquent de ne pas comprendre clairement la latence provoquée par les démarrages à froid dans le cadre d'une charge de travail irrégulière.
- Les canarys synthétiques ou la surveillance des utilisateurs réels (RUM) ne sont pas configurés avec le traçage. Sans canarys ni RUM, la télémétrie des interactions avec le client est omise de l'analyse des traces, ce qui donne un profil de performance incomplet.
- Les charges de travail hybrides incluent à la fois des outils de suivi natifs du cloud et des outils tiers, mais aucune mesure n'a été prise pour intégrer pleinement une solution de traçage unique. En fonction de la solution de traçage sélectionnée, les kits SDK de traçage natifs du cloud doivent être utilisés pour instrumenter des composants qui ne sont pas natifs du cloud ou des outils tiers doivent être configurés pour ingérer la télémétrie de suivi native du cloud.

Avantages liés au respect de cette bonne pratique : lorsque les équipes de développement sont alertées de problèmes, elles peuvent obtenir une image complète des interactions entre les composants du système, y compris la corrélation composant par composant avec la journalisation, les performances et les défaillances. Dans la mesure où le traçage permet d'identifier visuellement les causes profondes, vous passez moins de temps à les étudier. Les équipes qui comprennent en détail les interactions entre les composants prennent de meilleures décisions plus rapidement lors de la résolution des problèmes. L'analyse des traces des systèmes permet d'améliorer la prise de décisions, par exemple quand il convient de recourir à la reprise après sinistre (DR) ou de choisir le meilleur endroit pour mettre en œuvre des stratégies d'autoréparation, ce qui permet d'améliorer la satisfaction des clients envers vos services.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : moyen

Directives d'implémentation

Les équipes qui exploitent des applications distribuées peuvent utiliser des outils de traçage pour établir un identifiant de corrélation, collecter des traces de demandes et créer des cartes de service pour les composants connectés. Tous les composants de l'application doivent être inclus dans les

traces des demandes, notamment les clients de service, les passerelles d'intergiciels et les bus d'événements, les composants de calcul et le stockage, y compris les magasins de clés-valeurs et les bases de données. Incluez des canarys synthétiques et une surveillance des utilisateurs réels dans votre configuration de traçage de bout en bout pour mesurer les interactions avec les clients distants et la latence afin d'évaluer avec précision les performances de vos systèmes par rapport à vos contrats et objectifs de niveau de service.

Vous pouvez utiliser AWS X-Ray et les services d'instrumentation Amazon CloudWatch Application Monitoring pour fournir une vue complète des demandes au fur et à mesure qu'elles transitent par votre application. X-Ray collecte la télémétrie des applications et vous permet de la visualiser et de la filtrer en fonction des charges utiles, des fonctions, des traces, des services, des API, et peut être activée pour les composants du système sans code ou à faible code. La surveillance des applications CloudWatch inclut ServiceLens pour intégrer vos traces aux métriques, aux journaux et aux alarmes. La surveillance des applications CloudWatch inclut également des outils synthétiques pour surveiller vos points de terminaison et vos API, ainsi que la surveillance des utilisateurs réels pour instrumenter vos clients d'applications Web.

Étapes d'implémentation

- Utilisez AWS X-Ray sur tous les services natifs pris en charge tels qu'Amazon S3, AWS Lambda et Amazon API Gateway. Ces services AWS permettent l'utilisation de X-Ray grâce à des options de configuration utilisant l'infrastructure sous forme de code, de kits AWS SDK ou de la AWS Management Console.
- Applications instrumentales AWS Distro pour Open Telemetry et X-Ray ou agents de collecte tiers.
- Consultez le Guide du développeur AWS X-Ray pour une implémentation spécifique au langage de programmation. Ces sections de la documentation expliquent comment instrumenter les requêtes HTTP, les requêtes SQL et d'autres processus spécifiques à votre langage de programmation d'application.
- Utilisez le suivi X-Ray pour les canarys Amazon CloudWatch Synthetic et Amazon CloudWatch RUM afin d'analyser le chemin des demandes depuis votre client utilisateur final via votre infrastructure AWS en aval.
- Configurez les métriques et les alarmes CloudWatch en fonction de l'intégrité des ressources et de la télémétrie canary afin que les équipes soient rapidement alertées des problèmes, puis qu'elles puissent analyser en profondeur les traces et les cartographies de services avec ServiceLens.
- Activez l'intégration de X-Ray pour les outils de suivi tiers tels que Datadog, New Relic ou Dynatrace si vous utilisez des outils tiers pour votre solution de suivi principale.

Ressources

Bonnes pratiques associées:

- REL06-BP01 Surveiller tous les composants de la charge de travail (génération)
- REL11-BP01 Surveiller tous les composants de la charge de travail pour détecter les défaillances

Documents connexes:

- Qu'est-ce qu AWS X-Ray?
- Amazon CloudWatch : surveillance des applications
- Débogage avec Amazon CloudWatch Synthetics et AWS X-Ray
- Amazon Builders' Library : Instrumentation des systèmes distribués au profit de la visibilité opérationnelle
- Intégration de AWS X-Ray avec d'autres services AWS
- AWS Distro for OpenTelemetry et AWS X-Ray
- Amazon CloudWatch : utilisation de la surveillance synthétique
- Amazon CloudWatch : utilisation de CloudWatch RUM
- Configurer Amazon CloudWatch Synthetics Canary et Amazon CloudWatch Alarm
- Disponibilité et plus encore : comprendre et améliorer la résilience des systèmes distribués sur AWS

Exemples connexes :

- Un atelier sur l'observabilité

Vidéos connexes :

- AWS re:Invent 2022 - How to monitor applications across multiple accounts
- How to Monitor your AWS Applications

Outils associés :

- AWS X-Ray
- Amazon CloudWatch

- Amazon Route 53

Concevoir votre charge de travail de sorte qu'elle s'adapte aux changements de demande

Une charge de travail évolutive permet d'ajouter ou de supprimer automatiquement des ressources de manière à ce qu'elles correspondent à la demande actuelle à un moment donné.

Bonnes pratiques

- REL07-BP01 Utiliser l'automatisation lors de l'obtention des ressources ou de leur mise à l'échelle
- REL07-BP02 Obtenir des ressources après la détection d'un problème sur une charge de travail
- REL07-BP03 Obtenir des ressources après avoir réalisé qu'un plus grand nombre de ressources est nécessaire pour une charge de travail
- REL07-BP04 Testez votre charge de travail

REL07-BP01 Utiliser l'automatisation lors de l'obtention des ressources ou de leur mise à l'échelle

La définition programmatique, le provisionnement et la gestion de votre infrastructure et de vos ressources constituent la pierre angulaire de la fiabilité dans le cloud. L'automatisation vous aide à rationaliser le provisionnement des ressources, à faciliter des déploiements cohérents et sécurisés et à mettre à l'échelle les ressources sur l'ensemble de votre infrastructure.

Résultat escompté : vous gérez votre infrastructure en tant que code (IaC). Vous définissez et gérez votre code d'infrastructure dans des systèmes de contrôle de version (VCS). Vous déléguez le provisionnement des ressources AWS à des mécanismes automatisés et vous tirez parti de services gérés tels qu'Application Load Balancer (ALB), Network Load Balancer (NLB) et des groupes Auto Scaling. Vous provisionnez vos ressources à l'aide de pipelines d'intégration continue et livraison continue (CI/CD) afin que les modifications du code déclenchent automatiquement des mises à jour des ressources, y compris des mises à jour de vos configurations Auto Scaling.

Anti-modèles courants :

- Vous déployez des ressources manuellement via la ligne de commande ou dans la AWS Management Console (processus également appelé ClickOps).

- Vous associez étroitement vos ressources et composants d'application et vous créez ainsi des architectures rigides.
- Vous mettez en œuvre des politiques de mise à l'échelle rigides qui ne s'adaptent pas à l'évolution des exigences commerciales, des modèles de trafic ou des nouveaux types de ressources.
- Vous estimez manuellement la capacité pour répondre de façon anticipée à la demande.

Avantages liés au respect de cette bonne pratique : l'infrastructure en tant que code (IaC) permet de définir programmatiquement l'infrastructure. Vous pouvez ainsi gérer les modifications d'infrastructure en suivant le même cycle de développement logiciel que pour des modifications d'application, ce qui favorise la cohérence et la reproductibilité et réduit le risque de tâches manuelles susceptibles d'engendrer des erreurs. Vous pouvez rationaliser davantage le processus de provisionnement et de mise à jour des ressources en implémentant l'infrastructure en tant que code avec des pipelines de livraison automatisés. Vous pouvez déployer des mises à jour d'infrastructure de manière fiable et efficace sans nécessiter d'intervention manuelle. Cette agilité est particulièrement importante lorsque vous mettez à l'échelle les ressources pour répondre à des demandes fluctuantes.

Vous pouvez obtenir une mise à l'échelle dynamique et automatisée des ressources en conjonction avec l'infrastructure en tant que code et les pipelines de livraison. En surveillant les métriques clés et en appliquant des politiques de mise à l'échelle prédéfinies, Auto Scaling peut automatiquement provisionner ou déprovisionner les ressources selon les besoins, ce qui améliore les performances et la rentabilité. Cela réduit le risque d'erreurs manuelles ou de retards en réponse aux modifications des exigences en matière d'application ou de charge de travail.

La combinaison de l'infrastructure en tant que code, des pipelines de livraison automatisés et d'Auto Scaling aide les organisations à provisionner, mettre à jour et mettre à l'échelle leurs environnements en toute confiance. Cette automatisation est essentielle pour maintenir une infrastructure cloud réactive, résiliente et gérée efficacement.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : élevé

Directives d'implémentation

Pour configurer l'automatisation avec des pipelines CI/CD et une infrastructure en tant que code (IaC) pour votre architecture AWS, choisissez un système de contrôle de version tel que Git pour stocker vos modèles et votre configuration IaC. Ces modèles peuvent être écrits à l'aide d'outils tels qu'AWS CloudFormation. Pour commencer, définissez les composants de votre infrastructure (tels que les VPC AWS, les groupes Amazon EC2 Auto Scaling et les bases de données Amazon RDS) dans ces modèles.

Ensuite, intégrez ces modèles d'infrastructure en tant que code à un pipeline CI/CD pour automatiser le processus de déploiement. AWS CodePipeline fournit une solution AWS native transparente, ou vous pouvez utiliser d'autres solutions CI/CD tierces. Créez un pipeline qui s'active lorsque des modifications surviennent dans votre référentiel de contrôle de version. Configurez le pipeline de manière à inclure des étapes qui vérifient et valident vos modèles d'infrastructure en tant que code, déploient l'infrastructure dans un environnement intermédiaire, exécutent des tests automatisés et déploient finalement la solution en production. Incorporez des étapes d'approbation si nécessaire pour garder le contrôle sur les modifications. Ce pipeline automatisé accélère le déploiement, mais favorise également la cohérence et la fiabilité entre les environnements.

Configurez la mise à l'échelle automatique de ressources telles que les instances Amazon EC2, les tâches Amazon ECS et les réplicas de base de données dans votre infrastructure en tant que code afin d'assurer l'augmentation horizontale et la réduction horizontale automatiques selon les besoins. Cette approche améliore la disponibilité et les performances des applications et optimise les coûts en ajustant dynamiquement les ressources en fonction de la demande. Pour obtenir la liste des ressources prises en charge, consultez Amazon EC2 Auto Scaling et AWS Auto Scaling.

Étapes d'implémentation

1. Créez et utilisez un référentiel de code source pour stocker le code qui contrôle la configuration de votre infrastructure. Validez les modifications apportées à ce référentiel afin de refléter les modifications continues que vous souhaitez apporter.
2. Sélectionnez une solution d'infrastructure en tant que code, telle qu'AWS CloudFormation pour maintenir à jour votre infrastructure et détecter les incohérences (dérive) par rapport à l'état prévu.
3. Intégrez votre plateforme IaC à votre pipeline CI/CD pour automatiser les déploiements.
4. Déterminez et collectez les métriques appropriées pour la mise à l'échelle automatique des ressources.
5. Configurez la mise à l'échelle automatique des ressources à l'aide de politiques d'augmentation horizontale et de réduction horizontale pour vos composants de charge de travail. Envisagez d'utiliser une mise à l'échelle planifiée pour les modèles d'utilisation prévisibles.
6. Surveillez les déploiements pour détecter les défaillances et les régressions. Mettez en œuvre des mécanismes de restauration au sein de votre plateforme CI/CD pour annuler les modifications si nécessaire.

Ressources

Documents connexes :

- AWS Auto Scaling: Fonctionnement des plans de dimensionnement
- AWS Marketplace: produits utilisables avec autoscaling
- Gestion automatique de la capacité de débit avec l'autoscaling de DynamoDB
- Utiliser un équilibreur de charge avec un groupe Auto Scaling
- Qu'est-ce qu'AWS Global Accelerator ?
- Qu'est-ce qu'Amazon EC2 Auto Scaling?
- Qu'est-ce qu AWS Auto Scaling?
- Qu'est-ce qu'Amazon CloudFront?
- Qu'est-ce qu'Amazon Route 53?
- Qu'est-ce qu'Elastic Load Balancing?
- Qu'est-ce qu'un équilibreur de charge Network Load Balancer?
- Qu'est-ce qu'un équilibreur de charge Application Load Balancer?
- Intégration de Jenkins avec AWS CodeBuild et AWS CodeDeploy
- Création d'un pipeline à quatre étapes avec AWS CodePipeline

Vidéos connexes :

- Back to Basics : Déployer votre code sur Amazon EC2
- AWS Supports You | Démarrer votre solution d'infrastructure en tant que code en utilisant les modèles AWS CloudFormation
- Rationaliser votre processus de publication de logiciel en utilisant AWS CodePipeline
- Surveiller les ressources AWS à l'aide des tableaux de bord Amazon CloudWatch
- Créer des tableaux de bord CloudWatch inter-comptes et inter-régions | Amazon Web Services

REL07-BP02 Obtenir des ressources après la détection d'un problème sur une charge de travail

Si la disponibilité est affectée, mettez à l'échelle les ressources de manière réactive si nécessaire, afin de restaurer la disponibilité de la charge de travail.

Vous devez commencer par configurer les surveillances de l'état et les critères de ces vérifications pour indiquer quand la disponibilité est affectée par le manque de ressources. Informez ensuite le

personnel approprié qu'il doit mettre à l'échelle manuellement la ressource ou lancer l'automatisation pour procéder à une mise à l'échelle automatique.

La mise à l'échelle peut être ajustée manuellement en fonction de votre charge de travail. Par exemple, vous pouvez modifier le nombre d'instances EC2 dans un groupe Auto Scaling ou le débit d'une table DynamoDB via la AWS Management Console ou la AWS CLI. Cependant, l'automatisation doit être utilisée dans la mesure du possible (voir Utiliser l'automatisation lors de l'obtention ou de la mise à l'échelle des ressources).

Résultat escompté : les activités de mise à l'échelle (automatiquement ou manuellement) sont lancées pour rétablir la disponibilité en cas de détection d'une panne ou d'une dégradation de l'expérience client.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : moyen

Directives d'implémentation

Mettez en œuvre l'observabilité et la surveillance de tous les composants de votre charge de travail, afin de surveiller l'expérience client et de détecter les défaillances. Définissez les procédures, manuelles ou automatisées, de mise à l'échelle des ressources requises. Pour plus d'informations, consultez REL11-BP01 Surveiller tous les composants de la charge de travail afin de détecter les défaillances.

Étapes d'implémentation

- Définissez les procédures, manuelles ou automatisées, de mise à l'échelle des ressources requises.
  - Les procédures de mise à l'échelle dépendent de la conception des différents composants de votre charge de travail.
  - Les procédures de mise à l'échelle varient également en fonction de la technologie sous-jacente utilisée.
    - Les composants qui utilisent AWS Auto Scaling peuvent utiliser des plans de mise à l'échelle pour configurer un ensemble d'instructions pour la mise à l'échelle de vos ressources. Si vous utilisez AWS CloudFormation ou que vous ajoutez des balises à des ressources AWS, vous pouvez configurer des plans de mise à l'échelle pour différents ensembles de ressources, par application. Auto Scaling fournit des recommandations pour les stratégies de mise à l'échelle personnalisées pour chaque ressource. Une fois que vous avez créé votre plan de mise à l'échelle, Auto Scaling combine les méthodes de mise à l'échelle dynamique et prédictive pour

prendre en charge votre stratégie de mise à l'échelle. Pour plus de détails, consultez Comment fonctionnent les plans de mise à l'échelle.

- Amazon EC2 Auto Scaling permet de vous assurer que vous disposez du bon nombre d'instances Amazon EC2 disponibles pour gérer la charge de l'application. Vous créez des ensembles d'instances EC2, appelés groupes Auto Scaling. Vous pouvez spécifier le nombre minimal et maximal d'instances dans chaque groupe Auto Scaling et Amazon EC2 Auto Scaling s'assure que votre groupe n'est jamais au-dessus ou en dessous de ces limites. Pour plus d'informations, consultez Qu'est-ce qu'Amazon EC2 Auto Scaling ?
- L'autoscaling d'Amazon DynamoDB utilise le service d'autoscaling d'application pour ajuster de manière dynamique la capacité de débit approvisionné en votre nom, en réponse aux schémas de trafic réels. Cela permet à une table ou à un index secondaire global d'augmenter les capacités en lecture et écriture qui lui sont allouées afin de gérer les hausses soudaines de trafic sans limitation. Pour plus d'informations, voir Gestion automatique de la capacité de débit avec l'autoscaling de DynamoDB.

Ressources

Bonnes pratiques associées :

- REL07-BP01 Utiliser l'automatisation lors de l'obtention des ressources ou de leur mise à l'échelle
- REL11-BP01 Surveiller tous les composants de la charge de travail pour détecter les défaillances

Documents connexes :

- AWS Auto Scaling : Fonctionnement des plans de dimensionnement
- Gestion automatique de la capacité de débit avec l'autoscaling de DynamoDB
- Qu'est-ce qu'Amazon EC2 Auto Scaling ?

REL07-BP03 Obtenir des ressources après avoir réalisé qu'un plus grand nombre de ressources est nécessaire pour une charge de travail

L'une des fonctionnalités les plus précieuses du cloud computing est sa capacité à provisionner des ressources de manière dynamique.

Dans les environnements informatiques sur site traditionnels, vous devez identifier et provisionner une capacité suffisante à l'avance pour répondre à un pic de demande. Cela pose problème car cela

coûte cher et présente des risques pour la disponibilité si vous sous-estimez la capacité maximale requise par la charge de travail.

Dans le cloud, vous n'avez pas à le faire. Au lieu de cela, vous pouvez provisionner comme il se doit des capacités de calcul, de base de données et d'autres ressources pour répondre à la demande actuelle et prévue. Des solutions automatisées telles qu'Amazon EC2 Auto Scaling et Application Auto Scaling peuvent mettre en ligne des ressources pour vous sur la base de métriques que vous spécifiez. Cela peut faciliter le processus de mise à l'échelle et le rendre prévisible, et cela peut rendre votre charge de travail nettement plus fiable en garantissant que vous disposez de suffisamment de ressources à tout moment.

Résultat escompté : vous configurez la mise à l'échelle automatique des ressources de calcul et autres pour répondre à la demande. Vous prévoyez une marge de manœuvre suffisante dans vos politiques de mise à l'échelle pour permettre de répondre à des pics de trafic pendant que des ressources supplémentaires sont mises en ligne.

Anti-modèles courants :

- Vous provisionnez un nombre fixe de ressources évolutives.
- Vous choisissez une métrique de mise à l'échelle qui ne correspond pas à la demande réelle.
- Vous ne parvenez pas à prévoir une marge de manœuvre suffisante dans vos plans de mise à l'échelle pour faire face aux pics de demande.
- Vos politiques de mise à l'échelle tardent trop à augmenter la capacité, ce qui entraîne l'épuisement de la capacité et une dégradation du service lors de la mise en ligne de ressources supplémentaires.
- Vous ne parvenez pas à configurer correctement le nombre minimal et le nombre maximal de ressources, ce qui entraîne des échecs de mise à l'échelle.

Avantages liés au respect de cette bonne pratique : il est essentiel de disposer de suffisamment de ressources pour répondre à la demande actuelle afin de garantir une haute disponibilité de votre charge de travail et de respecter les objectifs de niveau de service (SLO) définis. La mise à l'échelle automatique vous permet de fournir la bonne quantité de ressources de calcul, de base de données et autres dont votre charge de travail a besoin afin de répondre à la demande actuelle et prévue. Vous n'avez pas besoin de déterminer la capacité maximale requise et d'allouer statiquement des ressources pour la fournir. Au lieu de cela, à mesure que la demande augmente, vous pouvez allouer davantage de ressources pour y répondre et, une fois que la demande est retombée, vous pouvez désactiver les ressources pour réduire les coûts.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : moyen

Directives d'implémentation

Tout d'abord, déterminez si le composant de charge de travail est adapté à la mise à l'échelle automatique. Ces composants sont appelés évolutifs horizontalement car ils fournissent les mêmes ressources et se comportent de manière identique. Parmi les composants évolutifs horizontalement, citons les instances EC2 configurées de la même manière, les tâches Amazon Elastic Container Service (ECS) et les pods exécutés sur Amazon Elastic Kubernetes Service (EKS). Ces ressources de calcul sont généralement situées derrière un équilibreur de charge et sont appelées réplicas.

Les autres ressources répliquées peuvent inclure des réplicas en lecture de base de données, des tables Amazon DynamoDB et des clusters Amazon ElastiCache (Redis OSS). Pour obtenir la liste complète des ressources prises en charge, consultez Services AWS que vous pouvez utiliser avec Application Auto Scaling.

Pour les architectures basées sur des conteneurs, il peut être nécessaire de procéder à une mise à l'échelle de deux manières différentes. Tout d'abord, vous devrez peut-être mettre à l'échelle les conteneurs qui fournissent des services évolutifs horizontalement. Ensuite, vous devrez peut-être mettre à l'échelle les ressources de calcul pour libérer de l'espace pour de nouveaux conteneurs. Il existe différents mécanismes de mise à l'échelle automatique pour chaque couche. Pour mettre à l'échelle les tâches ECS, vous pouvez utiliser Application Auto Scaling. Pour mettre à l'échelle les pods Kubernetes, vous pouvez utiliser Horizontal Pod Autoscaler (HPA) ou Kubernetes Event-driven Autoscaler (KEDA). Pour mettre à l'échelle les ressources de calcul, vous pouvez utiliser les fournisseurs de capacité pour ECS, ou pour Kubernetes, vous pouvez utiliser Karpenter ou Cluster Autoscaler.

Ensuite, sélectionnez de quelle façon vous voulez effectuer la mise à l'échelle automatique. Il existe trois options principales : la mise à l'échelle basée sur des métriques, la mise à l'échelle planifiée et la mise à l'échelle prédictive.

Mise à l'échelle basée sur des métriques

La mise à l'échelle basée sur des métriques provisionne les ressources en fonction de la valeur d'une ou de plusieurs métriques de mise à l'échelle. Une métrique de mise à l'échelle est une métrique qui correspond à la demande de votre charge de travail. Un bon moyen de déterminer les métriques de mise à l'échelle appropriées consiste à effectuer des tests de charge dans un environnement hors production. Pendant vos tests de charge, maintenez fixe le nombre de ressources évolutives et augmentez lentement la demande (par exemple, débit, simultanéité ou utilisateurs simulés). Recherchez ensuite des métriques qui augmentent (ou diminuent) à mesure que la demande

augmente, et inversement diminuent (ou augmentent) lorsque la demande diminue. Les métriques de mise à l'échelle typiques incluent l'utilisation du processeur, la profondeur de la file d'attente de travail (telle qu'une file d'attente Amazon SQS), le nombre d'utilisateurs actifs et le débit du réseau.

> **Note**
>
> AWS a observé qu'avec la plupart des applications, l'utilisation de la mémoire augmente à mesure que l'application monte en intensité, puis atteint une valeur stable. Lorsque la demande diminue, l'utilisation de la mémoire reste généralement élevée au lieu de diminuer en parallèle. Étant donné que l'utilisation de la mémoire ne correspond pas à la demande dans les deux cas de figure (à savoir en augmentant ni en diminuant avec la demande), réfléchissez bien avant de sélectionner cette métrique pour la mise à l'échelle automatique.

La mise à l'échelle basée sur des métriques est une opération latente. Plusieurs minutes peuvent être nécessaires pour que les métriques d'utilisation se propagent aux mécanismes de mise à l'échelle automatique, et ces mécanismes attendent généralement un signal clair d'augmentation de la demande avant de réagir. Ensuite, à mesure que l'outil de mise à l'échelle automatique crée de nouvelles ressources, la mise en service complète de ces dernières peut prendre plus de temps. Pour cette raison, il est important de ne pas définir vos cibles de métriques de mise à l'échelle trop proches de l'utilisation totale (par exemple, 90 % d'utilisation du processeur). Cela risque d'épuiser la capacité de ressources existante avant qu'une capacité supplémentaire puisse être mise en ligne. Les cibles typiques d'utilisation des ressources peuvent varier entre 50 et 70 % pour une disponibilité optimale, en fonction des modèles de demande et du temps nécessaire pour provisionner des ressources supplémentaires.

Mise à l'échelle planifiée

La mise à l'échelle planifiée provisionne ou supprime des ressources en fonction du calendrier ou de l'heure de la journée. Elle est fréquemment utilisée pour les charges de travail dont la demande est prévisible, telles que les pics d'utilisation pendant les heures ouvrables de semaine ou lors de soldes. Amazon EC2 Auto Scaling et Application Auto Scaling prennent tous deux en charge la mise à l'échelle planifiée. La fonctionnalité cron scaler de KEDA prend en charge la mise à l'échelle planifiée des pods Kubernetes.

Mise à l'échelle prédictive

La mise à l'échelle prédictive utilise le machine learning pour mettre à l'échelle automatiquement les ressources en fonction de la demande anticipée. La mise à l'échelle prédictive analyse la valeur

historique d'une métrique d'utilisation que vous fournissez et prédit en permanence sa valeur future. La valeur prédite est ensuite utilisée pour augmenter ou réduire verticalement la ressource. Amazon EC2 Auto Scaling peut effectuer une mise à l'échelle prédictive.

Étapes d'implémentation

1. Déterminez si le composant de charge de travail est adapté à la mise à l'échelle automatique.
2. Déterminez le type de mécanisme de mise à l'échelle le plus approprié pour la charge de travail : mise à l'échelle basée sur des métriques, mise à l'échelle planifiée ou mise à l'échelle prédictive.
3. Sélectionnez le mécanisme de mise à l'échelle automatique approprié pour le composant. Pour les instances Amazon EC2, utilisez Amazon EC2 Auto Scaling. Pour les autres services AWS, utilisez Application Auto Scaling. Pour les pods Kubernetes (tels que ceux exécutés dans un cluster Amazon EKS), pensez à Horizontal Pod Autoscaler (HPA) ou à Kubernetes Event-driven Autoscaling (KEDA). Pour les nœuds Kubernetes ou EKS, pensez à Karpenter et à Cluster Auto Scaler (CAS).
4. Pour une mise à l'échelle basée sur des métriques ou planifiée, effectuez des tests de charge afin de déterminer les métriques de mise à l'échelle et les valeurs cibles appropriées pour votre charge de travail. Pour une mise à l'échelle planifiée, déterminez le nombre de ressources nécessaires aux dates et heures que vous sélectionnez. Déterminez le nombre maximal de ressources nécessaires pour répondre aux pics de trafic attendus.
5. Configurez l'outil de mise à l'échelle en fonction des informations collectées ci-dessus. Pour plus d'informations, consultez la documentation du service de mise à l'échelle automatique. Vérifiez que les limites de mise à l'échelle maximale et minimale sont correctement configurées.
6. Vérifiez que la configuration de mise à l'échelle fonctionne comme prévu. Effectuez des tests de charge dans un environnement hors production, observez comment le système réagit et ajustez le cas échéant. Lorsque vous activez la mise à l'échelle automatique en production, configurez les alarmes appropriées pour être averti de tout comportement inattendu.

Ressources

Documents connexes :

- Qu'est-ce qu'Amazon EC2 Auto Scaling?
- Conseils prescriptifs AWS : tests de charge des applications
- AWS Marketplace: produits utilisables avec autoscaling
- Gestion automatique de la capacité de débit avec l'autoscaling de DynamoDB

- Mise à l’échelle prédictive pour EC2 alimentée par le machine learning
- Mise à l’échelle planifiée pour Amazon EC2 Auto Scaling
- Telling Stories About Little’s Law

REL07-BP04 Testez votre charge de travail

Adoptez une méthodologie de test de charge pour déterminer si la mise à l’échelle répond aux exigences de la charge de travail.

Il est important d’exécuter régulièrement des tests de charge. Les tests de charge doivent découvrir le point de rupture et tester les performances de votre charge de travail. AWS facilite la mise en place d'environnements de test temporaires qui modélisent l'échelle de votre charge de travail de production. Dans le Cloud, vous pouvez créer un environnement d’essai à l’échelle de la production et à la demande, exécuter les tests, puis désactiver les ressources. Puisque vous ne payez l’environnement de test que lorsqu’il s’exécute, vous pouvez simuler votre environnement réel pour une fraction du coût d’un test sur site.

Les tests de charge en production doivent également être intégrés aux tests de simulation de pannes, lors desquels le système de production est mis sous tension pendant les périodes où le client est moins utilisé et tout le personnel est disponible pour interpréter les résultats et résoudre les problèmes qui surviennent.

Anti-modèles courants :

- Exécution de tests de charge sur des déploiements qui n’ont pas la même configuration que votre production.
- Exécution d’un test de charge uniquement sur des éléments individuels de votre charge de travail, et non sur l’ensemble de la charge de travail.
- Exécution de tests de charge avec un sous-ensemble de demandes et non un ensemble représentatif de demandes réelles.
- Exécution de tests de charge avec un faible facteur de sécurité au-dessus de la charge prévue.

Avantages du respect de cette bonne pratique : vous savez quels composants de votre architecture échouent sous charge et vous pouvez identifier les métriques à surveiller qui indiquent suffisamment à temps que vous approchez de cette charge pour que vous résolviez le problème et empêchiez ainsi l’impact de cette défaillance.

Niveau d’exposition au risque si cette bonne pratique n’est pas respectée : moyen

Directives d’implémentation

- Exécutez des tests de charge pour identifier l’aspect de votre charge de travail qui indique que vous devez ajouter ou supprimer de la capacité. Les tests de charge doivent avoir un trafic représentatif similaire à ce que vous recevez en production. Augmentez la charge tout en surveillant les métriques que vous avez instrumentées pour déterminer quelle métrique indique quand vous devez ajouter ou supprimer des ressources.
  - Test de charge distribué sur AWS : simulez des milliers d'utilisateurs connectés
    - Identifiez le mélange de demandes. Comme vous pouvez avoir divers mélanges de demandes, vous devez examiner les différentes périodes lors de l’identification de la combinaison de trafic.
    - Implémentez un pilote de charge. Vous pouvez utiliser un code personnalisé, un logiciel open source ou un logiciel commercial pour implémenter un pilote de charge.
    - Effectuez un test de charge initial avec une faible capacité. Vous constatez des effets immédiats en entraînant une charge moindre, éventuellement aussi petite qu’une instance ou un conteneur.
    - Effectuez un test de charge par rapport à une capacité plus importante. Étant donné que les effets seront différents sur une charge distribuée, vous devez procéder à des essais dans un environnement aussi proche que possible de celui du produit.

Ressources

Documents connexes :

- Test de charge distribué sur AWS : simulez des milliers d'utilisateurs connectés
- Tests de charge des applications

Vidéos connexes :

- AWS Sommet ANZ 2023 : Accélérez en toute confiance grâce AWS aux tests de charge distribués

Implémentation de la modification

Des modifications contrôlées sont nécessaires pour déployer de nouvelles fonctionnalités et garantir que les charges de travail et l'environnement d'exploitation exécutent des logiciels connus et correctement corrigés. Si ces modifications ne sont pas maîtrisées, il devient difficile d'en prévoir les effets ou de résoudre les problèmes qui en découlent.

Modèles de déploiement supplémentaires pour minimiser les risques

Les indicateurs de fonction (également connus sous le nom de basculements de fonction) sont des options de configuration d'une application. Vous pouvez déployer le logiciel en désactivant une fonction afin que vos clients ne la voient pas. Vous pouvez ensuite activer ladite fonction, comme vous le feriez pour un déploiement Canary, ou définir le rythme des modifications sur 100 % pour voir l'effet. Si le déploiement rencontre des problèmes, vous pouvez simplement faire marche arrière pour la fonction sans avoir besoin de restauration.

Déploiement des zones isolées contre les pannes : l'une des principales règles qu'AWS a instaurées pour ses propres déploiements consiste à éviter de toucher en même temps plusieurs zones de disponibilité au sein d'une région. Cet aspect est essentiel pour garantir que les zones de disponibilité sont indépendantes dans le cadre de vos calculs de disponibilité. Nous vous recommandons de veiller aux mêmes considérations lors de vos déploiements.

Examens de disponibilité opérationnelle (ORR)

AWS trouve utile d'effectuer des examens permettant d'évaluer l'exhaustivité des tests, la capacité à surveiller et plus important, la capacité à auditer les performances des applications sur ses SLA et fournir des données en cas d'interruption ou d'autre anomalie opérationnelle. Un ORR formel est effectué avant le déploiement initial de la production. AWS reproduit régulièrement les ORR (une fois par an, ou avant les périodes de performances critiques) pour s'assurer qu'il n'y a pas eu de décalage par rapport aux attentes opérationnelles. Pour en savoir plus sur la préparation opérationnelle, consultez le pilier Excellence opérationnelle d'AWS Well-Architected Framework.

Bonnes pratiques

- REL08-BP01 Utiliser des runbooks pour les activités standard telles que le déploiement
- REL08-BP02 Intégrer les tests fonctionnels dans le cadre de votre déploiement
- REL08-BP03 Intégrer les tests de résilience dans le cadre de votre déploiement
- REL08-BP04 Effectuer le déploiement à l'aide d'une infrastructure immuable

- REL08-BP05 Déployer les modifications avec l’automatisation

REL08-BP01 Utiliser des runbooks pour les activités standard telles que le déploiement

Les runbooks sont les procédures prédéfinies destinées à parvenir à un résultat spécifique. Utilisez des runbooks pour effectuer des tâches manuelles ou automatiques standard. Il peut s’agir du déploiement d’une charge de travail, de l’application de correctifs à une charge de travail ou de la modification du DNS.

Par exemple, mettez en place des processus pour assurer la sécurité des restaurations pendant les déploiements. Pour garantir la fiabilité d’un service, il est essentiel de s’assurer que vous pouvez restaurer un déploiement sans interruption pour vos clients.

Concernant les procédures de runbook, commencez par un processus manuel efficace valide, mettez-le en œuvre dans le code et, le cas échéant, déclenchez son exécution automatique.

Même pour les charges de travail sophistiquées hautement automatisées, les runbooks restent utiles pour exécuter des tests de simulation de pannes ou répondre à des exigences rigoureuses en matière de rapports et d’audit.

Notez que les playbooks sont utilisés en réponse à des incidents spécifiques et que les runbooks le sont pour obtenir des résultats spécifiques. En règle générale, les runbooks sont destinés aux activités de routine, tandis que les playbooks sont utilisés pour répondre à des événements non réguliers.

Anti-modèles courants :

- Exécution de modifications imprévues de la configuration en production.
- Ignorer les étapes de votre plan afin d’accélérer le déploiement, ce qui entraîne un échec du déploiement.
- Effectuez des modifications sans tester l’annulation de la modification.

Avantages liés au respect de cette bonne pratique : une planification efficace des modifications augmente votre capacité à exécuter correctement la modification, car vous êtes conscient de tous les systèmes concernés. Vous gagnez en confiance si vous réussissez à valider des modifications que vous apportez aux environnements de test.

Niveau d’exposition au risque si cette bonne pratique n’est pas respectée : élevé

Directives d’implémentation

- Obtenez des réponses cohérentes et rapides à des événements bien compris en documentant les procédures dans des runbooks.
- Utilisez le principe de l’infrastructure en tant que code pour définir votre infrastructure. En ayant recours à AWS CloudFormation ou à un tiers de confiance pour définir votre infrastructure, vous pouvez utiliser le contrôle de version et suivre les modifications apportées à la version du logiciel.
  - Utilisez AWS CloudFormation ou un fournisseur tiers de confiance pour définir votre infrastructure.
    - Qu’est-ce qu AWS CloudFormation?
  - Créez des modèles qui sont singuliers et découplés, en utilisant de bons principes de conception de logiciels.
    - Déterminez les autorisations, les modèles et les responsables de l’implémentation
      - Contrôle de l’accès avec Gestion des identités et des accès AWS
    - Utilisez un système de gestion de code source hébergé basé sur une technologie populaire telle que Git pour stocker votre code source et votre configuration d’infrastructure en tant que code (IaC).

Ressources

Documents connexes:

- Partenaire APN : partenaires pouvant vous aider à créer des solutions de déploiement automatisées
- AWS Marketplace: produits pouvant être utilisés pour automatiser vos déploiements
- Qu’est-ce qu AWS CloudFormation?

Exemples connexes:

- Automatisation des opérations avec les playbooks et les runbooks

REL08-BP02 Intégrer les tests fonctionnels dans le cadre de votre déploiement

Utilisez des techniques telles que les tests unitaires et les tests d'intégration qui valident les fonctionnalités requises.

Un test unitaire est un processus consistant à tester la plus petite unité fonctionnelle du code afin de valider son comportement. Un test d'intégration vise à valider que chaque fonctionnalité de l'application respecte les exigences du logiciel. Alors que les tests unitaires visent à tester une partie d'une application de manière isolée, les tests d'intégration prennent en compte les effets secondaires (par exemple, l'effet de la modification des données lors d'une opération de mutation). Dans les deux cas, les tests doivent être intégrés dans un pipeline de déploiement et si les critères de réussite ne sont pas respectés, le pipeline est arrêté ou annulé. Ces tests sont exécutés dans un environnement de pré-production, qui est mis en place avant la production dans le pipeline.

Vous obtenez les meilleurs résultats lorsque ces tests sont exécutés automatiquement dans le cadre d'actions de génération et de déploiement. Par exemple, avec AWS CodePipeline, les développeurs valident les modifications apportées à un référentiel source dans lequel CodePipeline détecte automatiquement les modifications. L'application est générée et des tests unitaires sont exécutés. Une fois les tests unitaires réussis, le code ainsi généré est déployé sur les serveurs intermédiaires, à des fins de test. Depuis le serveur intermédiaire, CodePipeline exécute d'autres tests, tels que des tests d'intégration ou de chargement. Une fois ces tests terminés avec succès, CodePipeline déploie le code testé et approuvé sur les instances de production.

Résultat escompté : vous utilisez l'automatisation pour effectuer des tests unitaires et d'intégration afin de valider que votre code se comporte comme prévu. Ces tests sont intégrés au processus de déploiement, et un échec entraîne l'abandon du déploiement.

Anti-modèles courants :

- Vous ignorez ou contournez les échecs de test et les plans pendant le processus de déploiement afin de raccourcir le délai de déploiement.
- Vous effectuez manuellement des tests en dehors du pipeline de déploiement.
- Vous sautez des étapes de test dans l'automatisation par le biais de flux de travail d'urgence manuels.
- Vous exécutez des tests automatisés dans un environnement qui ne ressemble vraiment pas à l'environnement de production.

- Vous créez une suite de tests trop peu flexible et difficile à maintenir, à mettre à jour ou à mettre à l'échelle à mesure de l'évolution de l'application.

Avantages liés au respect de cette bonne pratique : les tests automatisés effectués au cours du processus de déploiement détectent les problèmes à un stade précoce, ce qui réduit le risque d'une mise en production avec des bogues ou des comportements inattendus. Les tests unitaires valident le fait que le code se comporte comme souhaité et que les contrats d'API sont respectés. Les tests d'intégration valident le fait que le système fonctionne conformément aux exigences spécifiées. Ces types de tests vérifient l'état de fonctionnement escompté de composants tels que les interfaces utilisateur, les API, les bases de données et le code source.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : élevé

Directives d'implémentation

Adoptez une approche de développement piloté par les tests (TDD) pour écrire des logiciels, dans laquelle vous développez des cas de test pour spécifier et valider votre code. Pour commencer, créez des cas de test pour chaque fonction. Si le test échoue, vous devez écrire un nouveau code pour réussir le test. Cette approche vous permet de valider le résultat escompté de chaque fonction. Exécutez des tests unitaires et validez leur réussite avant de valider le code dans un référentiel de code source.

Mettez en œuvre des tests unitaires et d'intégration dans le cadre des étapes de création, de test et de déploiement du pipeline CI/CD. Automatisez les tests et lancez automatiquement les tests chaque fois qu'une nouvelle version de l'application est prête à être déployée. Si les critères de réussite ne sont pas respectés, le pipeline est arrêté ou annulé.

Dans le cas d'une application Web ou mobile, effectuez des tests d'intégration automatisés sur plusieurs navigateurs de bureau ou sur des appareils réels. Cette approche est particulièrement utile pour valider la compatibilité et les fonctionnalités des applications mobiles sur un large éventail d'appareils.

Étapes d'implémentation

1. Rédigez des tests unitaires avant d'écrire du code fonctionnel (développement piloté par les tests ou TDD). Établissez des lignes directrices en matière de code afin que l'écriture et l'exécution de tests unitaires soient une exigence non fonctionnelle de codage.

2. Créez une suite de tests d'intégration automatisés qui couvrent les fonctionnalités testables identifiées. Ces tests doivent simuler les interactions avec les utilisateurs et valider les résultats attendus.
3. Créez l'environnement de test nécessaire pour exécuter les tests d'intégration. Cela peut inclure des environnements de préparation ou de pré-production qui imitent étroitement l'environnement de production.
4. Configurez vos étapes source, de construction, de test et de déploiement à l'aide de la console AWS CodePipeline ou de l'AWS Command Line Interface (CLI).
5. Déployez l'application une fois que le code a été créé et testé. AWS CodeDeploy peut la déployer dans vos environnements intermédiaire (tests) et de production. Ces environnements peuvent inclure des instances Amazon EC2, des fonctions AWS Lambda et des serveurs sur site. Le même mécanisme de déploiement doit être utilisé pour déployer l'application dans tous les environnements.
6. Surveillez la progression de votre pipeline et le statut de chaque étape. Utilisez des contrôles de qualité pour bloquer le pipeline en fonction du statut des tests. Vous pouvez aussi recevoir des notifications en cas d'échec ou d'achèvement d'une étape du pipeline.
7. Surveillez en permanence les résultats des tests et recherchez des modèles, des régressions ou des domaines nécessitant une plus grande attention. Utilisez ces informations pour améliorer la suite de tests, identifier les domaines de l'application nécessitant des tests plus approfondis et optimiser le processus de déploiement.

Ressources

Bonnes pratiques associées :

- REL07-BP04 Effectuer un test de charge de votre charge de travail
- REL08-BP03 Intégrer les tests de résilience dans le cadre de votre déploiement
- REL12-BP04 Tester la résilience à l'aide de l'ingénierie du chaos

Documents connexes :

- Conseils prescriptifs AWS : automatisation des tests
- Intégration et livraison continues
- Indicateurs pour les tests fonctionnels
- Surveillance des pipelines

- Utilisation d'AWS CodePipeline avec AWS CodeBuild pour tester le code et exécuter des générations
- AWS Device Farm

REL08-BP03 Intégrer les tests de résilience dans le cadre de votre déploiement

Intégrez des tests de résilience en introduisant consciemment des défaillances dans le système afin de mesurer sa fonctionnalité en cas de scénarios perturbateurs. Les tests de résilience sont différents des tests unitaires et fonctionnels qui sont généralement intégrés dans les cycles de déploiement, car ils se concentrent sur l'identification des défaillances imprévues de votre système. Bien qu'il soit prudent de commencer par l'intégration des tests de résilience en préproduction, fixez-vous comme objectif d'implémenter ces tests en production dans le cadre de vos journées de simulation.

Résultat escompté : les tests de résilience contribuent à renforcer la confiance dans la capacité du système à résister à la dégradation en cours de production. Les expériences identifient les points faibles susceptibles d'entraîner une défaillance, ce qui vous permet d'améliorer le système afin d'atténuer automatiquement et efficacement les défaillances et la dégradation.

Anti-modèles courants :

- Manque d'observabilité et de surveillance dans les processus de déploiement
- Dépendance à l'humain pour résoudre les défaillances du système
- Mécanismes d'analyse de mauvaise qualité
- Accent mis sur les problèmes connus d'un système et manque d'expérimentation pour identifier les problèmes inconnus
- Identification des défaillances, mais pas de solution
- Aucune documentation sur les résultats ni aucun runbook

Avantages de l'établissement de bonnes pratiques : les tests de résilience intégrés à vos déploiements permettent d'identifier les problèmes inconnus du système qui, autrement, passeraient inaperçus, ce qui peut entraîner des interruptions de production. L'identification de ces problèmes système inconnus vous permet de documenter les résultats, d'intégrer les tests dans votre processus CI/CD et de créer des runbooks, ce qui simplifie leur atténuation grâce à des mécanismes efficaces et reproductibles.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : moyen

Directives d'implémentation

Les formes de test de résilience les plus courantes pouvant être intégrées dans les déploiements de votre système sont la reprise après sinistre et l'ingénierie du chaos.

- Incluez des mises à jour de vos plans de reprise après sinistre et de vos procédures opérationnelles standard (SOP) lors de tout déploiement important.
- Intégrez des tests de fiabilité à vos pipelines de déploiement automatisés. Des services tels que AWS Resilience Hub peuvent être intégrés à votre pipeline CI/CD pour établir des évaluations continues de la résilience qui sont automatiquement évaluées dans le cadre de chaque déploiement.
- Définissez vos applications dans AWS Resilience Hub. Les évaluations de résilience génèrent des extraits de code qui vous aident à créer des procédures de restauration sous forme de documents AWS Systems Manager pour vos applications et fournissent une liste de moniteurs et d'alarmes Amazon CloudWatch recommandés.
- Une fois vos plans de reprise après sinistre et vos SOP mis à jour, effectuez des tests de reprise après sinistre pour vérifier leur efficacité. Les tests de reprise après sinistre contribuent à déterminer si vous pouvez restaurer votre système après un événement et revenir à un fonctionnement normal. Vous pouvez simuler différentes stratégies de reprise après sinistre et déterminer si votre planification est suffisante pour répondre à vos exigences de disponibilité. Les stratégies courantes de reprise après sinistre incluent la sauvegarde et la restauration, l'environnement en veille, la veille à froid, la veille à chaud, la veille permanente et la veille active/active. Elles diffèrent toutes en matière de coût et de complexité. Avant les tests de reprise après sinistre, nous vous recommandons de définir votre objectif de délai de reprise (RTO) et votre objectif de point de reprise (RPO) afin de simplifier le choix de la stratégie à simuler. AWS propose des outils de reprise après sinistre comme Reprise après sinistre AWS Elastic destinés à vous aider à démarrer votre planification et vos tests.
- Les expériences d'ingénierie du chaos introduisent des perturbations dans le système, telles que des pannes de réseau et des pannes de service. En simulant le système avec des pannes contrôlées, vous pouvez en découvrir les vulnérabilités tout en limitant les impacts des pannes injectées. Comme pour les autres stratégies, exécutez des simulations de défaillances contrôlées dans des environnements non liés à la production en utilisant des services tels que AWS Fault Injection Service pour gagner en confiance avant de les déployer en production.

Ressources

Documents connexes :

- Tester les défaillances à l'aide de tests de résilience pour renforcer la préparation à la reprise
- Évaluation continue de la résilience des applications avec AWS Resilience Hub et AWS CodePipeline
- Architecture de reprise après sinistre (DR) sur AWS, première partie : stratégies de reprise dans le cloud
- Vérifier la résilience de vos charges de travail à l'aide de Chaos Engineering
- Principes de l'ingénierie du chaos
- Atelier d'ingénierie du chaos

Vidéos connexes :

- AWS re:Invent 2020: Testing Resilience using Chaos Engineering
- Improve Application Resilience with AWS Fault Injection Service
- Prepare & Protect Your Applications From Disruption With AWS Resilience Hub

REL08-BP04 Effectuer le déploiement à l'aide d'une infrastructure immuable

Une infrastructure immuable est un modèle qui exige qu'aucune mise à jour, aucune application de correctifs de sécurité ni aucun changement de configuration ne se produise sur place sur les charges de travail de production. Lorsqu'un changement est nécessaire, l'architecture est intégrée à la nouvelle infrastructure et déployée en production.

Suivez une stratégie de déploiement d'infrastructure immuable pour améliorer la fiabilité, la cohérence et la reproductibilité de vos déploiements de charges de travail.

Résultat escompté : avec une infrastructure immuable, aucune modification sur place n'est autorisée pour exécuter les ressources de l'infrastructure dans le cadre d'une charge de travail. Lorsqu'une modification est nécessaire, un nouvel ensemble de ressources d'infrastructure contenant toutes les modifications nécessaires est déployé parallèlement à vos ressources existantes. Ce déploiement est validé automatiquement et, en cas de succès, le trafic est progressivement transféré vers ce nouvel ensemble de ressources.

Cette stratégie de déploiement s'applique notamment aux mises à jour logicielles, aux correctifs de sécurité, aux modifications de l'infrastructure, ainsi qu'aux mises à jour de la configuration et des applications.

Anti-modèles courants :

- Modifications sur place des ressources d'infrastructure en cours d'exécution.

Avantages liés au respect de cette bonne pratique :

- Cohérence accrue entre les environnements : comme il n'existe aucune différence dans les ressources d'infrastructure entre les environnements, la cohérence est améliorée et les tests simplifiés.
- Réduction des dérives de configuration : en remplaçant fréquemment les ressources d'infrastructure à partir d'une configuration de base connue et contrôlée par les versions, l'infrastructure est réinitialisée à un état connu, testé et fiable, ce qui évite les dérives de configuration.
- Déploiements atomiques fiables : les déploiements se terminent avec succès ou rien ne change, ce qui améliore la cohérence et la fiabilité du processus de déploiement.
- Déploiements simplifiés : les déploiements sont simplifiés, car ils n'ont pas besoin de prendre en charge les mises à niveau. Les mises à niveau sont simplement de nouveaux déploiements.
- Déploiements plus sûrs avec des processus de restauration et de récupération rapides : les déploiements sont plus sûrs, car la version de travail précédente n'est pas modifiée. Vous pouvez la restaurer si des erreurs sont détectées.
- Position de sécurité améliorée : en interdisant les modifications de l'infrastructure, les mécanismes d'accès à distance (tels que SSH) peuvent être désactivés. Vous pouvez ainsi réduire les vecteurs d'attaque tout en renforçant la sécurité de votre organisation.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : moyen

Directives d'implémentation

Automatisation

Lors de la définition d'une stratégie de déploiement d'infrastructure immuable, il est recommandé de recourir autant que possible à l'automatisation afin d'accroître la reproductibilité et de minimiser le

risque d'erreur humaine. Pour plus de détails, voir REL08-BP05 Déployer les modifications grâce à l'automatisation et Automatisation de déploiements sûrs et sans intervention directe.

Avec l'infrastructure en tant que code (IaC), les étapes de provisionnement, d'orchestration et de déploiement de l'infrastructure sont définies de manière programmatique, descriptive et déclarative et stockées dans un système de contrôle des sources. L'utilisation de l'infrastructure en tant que code simplifie l'automatisation du déploiement de l'infrastructure et contribue à garantir l'immuabilité de cette dernière.

Modèles de déploiement

Lorsqu'une modification de la charge de travail est requise, la stratégie de déploiement d'infrastructure immuable impose le déploiement d'un nouvel ensemble de ressources d'infrastructure comprenant toutes les modifications nécessaires. Il est important que ce nouvel ensemble de ressources suive un schéma de déploiement qui minimise l'impact sur les utilisateurs. Il existe deux stratégies principales pour ce type de déploiement :

Déploiement Canary : consiste à diriger un petit nombre de vos clients vers la nouvelle version, généralement exécutée sur une seule instance de service (la version Canary). Examinez ensuite en profondeur les modifications de comportement ou les erreurs générées. Vous pouvez supprimer le trafic du Canary si vous rencontrez des problèmes critiques et faire basculer les utilisateurs vers la version précédente. Si le déploiement est réussi, vous pouvez le continuer à la vitesse souhaitée, tout en surveillant les modifications (pour éviter les erreurs), jusqu'à ce qu'il soit terminé. AWS CodeDeploy peut être configuré avec une configuration de déploiement qui permettra un déploiement canary.

Déploiement bleu/vert : semblable au déploiement Canary si ce n'est qu'un parc complet de l'application est déployé en parallèle. Vos déploiements alternent entre deux piles (bleu et vert). Une fois encore, vous pouvez faire basculer le trafic vers la nouvelle version et revenir à l'ancienne si vous rencontrez des problèmes lors du déploiement. Généralement, tout le trafic est commuté en même temps. Vous pouvez toutefois également utiliser des fractions de votre trafic vers chaque version pour modifier l'adoption de la nouvelle version à l'aide des capacités de routage DNS pondéré d'Amazon Route 53. AWS CodeDeploy et AWS Elastic Beanstalk peuvent être configurés avec une configuration de déploiement qui permet un déploiement bleu/vert.

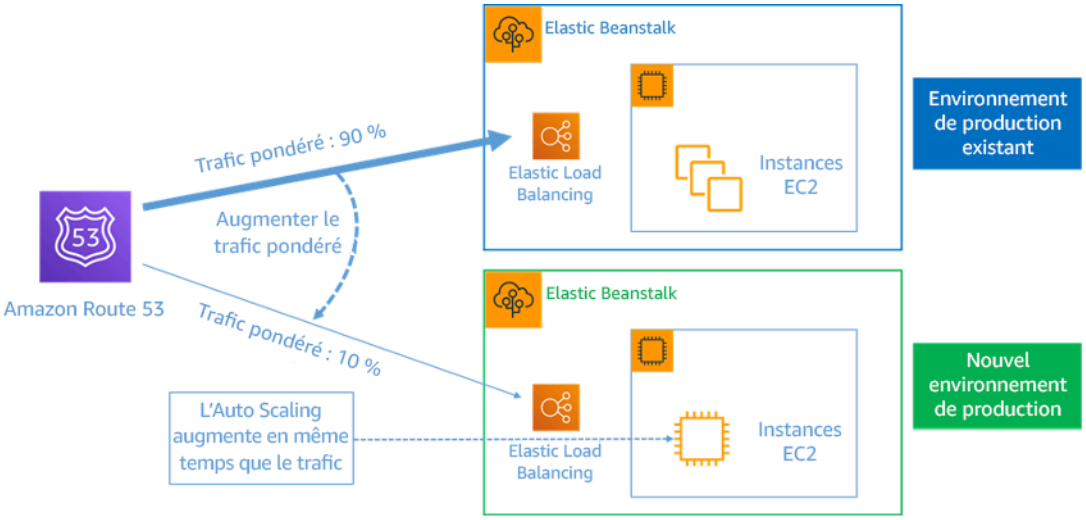


Figure 8 : Déploiement bleu/vert avec AWS Elastic Beanstalk et Amazon Route 53

Détection des écarts

L'écart est défini comme tout changement qui fait qu'une ressource d'infrastructure présente un état ou une configuration différent de ce qui est attendu. Toute modification de configuration non gérée va à l'encontre de la notion d'infrastructure immuable et doit être détectée et corrigée afin de garantir la mise en œuvre d'une infrastructure immuable.

Étapes d'implémentation

- Interdisez la modification sur place des ressources d'infrastructure en cours d'exécution.
  - Vous pouvez utiliser Gestion des identités et des accès AWS (IAM) pour spécifier qui ou quoi peut accéder aux services et aux ressources dans AWS, gérer de manière centralisée les autorisations détaillées et analyser les accès pour affiner les autorisations dans AWS.
- Automatisez le déploiement des ressources d'infrastructure pour améliorer la reproductibilité et minimiser le risque d'erreur humaine.
  - Comme décrit dans le livre blanc Introduction au DevOps sur AWS, l'automatisation est la pierre angulaire des services AWS et est prise en charge en interne dans tous les services, fonctionnalités et offres.
  - La préintégration de votre Amazon Machine Image (AMI) peut accélérer son lancement. EC2 Image Builder est un service AWS entièrement géré qui vous permet d'automatiser la création, la maintenance, la validation, le partage et le déploiement d'une AMI Linux ou Windows personnalisée, fiable et à jour.
  - Les services qui prennent en charge l'automatisation incluent :

    - AWS Elastic Beanstalk est un service permettant de déployer et de mettre à l'échelle rapidement des applications et des services web développés avec Java, .NET, PHP, Node.js, Python, Ruby, Go et Docker sur des serveurs courants, tels qu'Apache, NGINX, Passenger et IIS.
    - AWS Proton aide les équipes de plateforme à connecter et à coordonner les différents outils dont vos équipes de développement ont besoin pour le provisionnement de l'infrastructure, les déploiements de code, la surveillance et les mises à jour. AWS Proton permet une infrastructure automatisée sous forme de provisionnement de code et de déploiement d'applications sans serveur et basées sur des conteneurs.
- L'utilisation d'une infrastructure en tant que code facilite l'automatisation du déploiement de l'infrastructure et contribue à garantir l'immuabilité de l'infrastructure. AWS fournit des services de création, de déploiement et de maintenance programmatique, descriptive et déclarative de l'infrastructure.
    - AWS CloudFormation aide les développeurs à créer des ressources AWS de manière ordonnée et prévisible. Les ressources sont écrites dans des fichiers texte au format JSON ou YAML. Les modèles nécessitent une syntaxe et une structure spécifiques, qui dépendent des types de ressources créées et gérées. Vous créez vos ressources au format JSON ou YAML avec n'importe quel éditeur de code, vous les archivez dans un système de contrôle de version, puis CloudFormation crée les services spécifiés d'une manière sûre et reproductible.
    - AWS Serverless Application Model (AWS SAM) est un cadre open source que vous pouvez utiliser pour construire des applications sans serveur sur AWS. AWS SAM s'intègre à d'autres services AWS et constitue une extension de CloudFormation.
    - AWS Cloud Development Kit (AWS CDK) est un cadre de développement logiciel open source que vous pouvez utiliser pour modéliser et allouer vos ressources d'applications cloud à l'aide de langages de programmation familiers. Vous pouvez utiliser AWS CDK pour modéliser l'infrastructure d'applications avec TypeScript, Python, Java et .NET. AWS CDK utilise CloudFormation en arrière-plan pour provisionner les ressources de manière sécurisée et reproductible.
    - API de commande du Cloud AWS introduit un ensemble commun d'API CRUDL (Create, Read, Update, Delete, and List) pour aider les développeurs à gérer leur infrastructure cloud de manière simple et cohérente. Les API courantes de Cloud Control API permettent aux développeurs de gérer de manière uniforme le cycle de vie des services AWS et tiers.
- Mettez en œuvre des modèles de déploiement qui minimisent l'impact sur les utilisateurs.
    - Déploiements canary :

    - Configuration d'un déploiement de la version canary API Gateway
    - Créer un pipeline avec des déploiements Canary pour Amazon ECS à l'aide de AWS App Mesh
  - Déploiements bleu/vert : le livre blanc sur les déploiements bleu/vert sur AWS décrit des exemples de techniques pour mettre en œuvre des stratégies de déploiement bleu/vert.
- Détectez les écarts de configuration ou d'état. Pour plus de détails, consultez Détection de modifications non gérées de la configuration des piles et des ressources.

Ressources

Bonnes pratiques associées :

- REL08-BP05 Déployer les modifications avec l'automatisation

Documents connexes :

- Automatiser les déploiements sécurisés et sans intervention
- Tirer parti de AWS CloudFormation pour créer une infrastructure immuable chez Nubank
- Infrastructure en tant que code
- Implémentation d'une alarme pour détecter automatiquement l'écart dans les piles AWS CloudFormation

Vidéos connexes :

- AWS re:Invent 2020: Reliability, consistency, and confidence through immutability

REL08-BP05 Déployer les modifications avec l'automatisation

Les déploiements et l'application de correctifs sont automatisés pour éliminer l'impact négatif.

Les modifications apportées aux systèmes de production sont l'un des secteurs de risque les plus importants pour de nombreuses organisations. Nous considérons les déploiements comme un problème de premier ordre à résoudre, tout comme les problèmes opérationnels que le logiciel rencontre. Aujourd'hui, il convient d'appliquer l'automatisation dès que les opérations le permettent, y compris lors des tests et du déploiement de modifications, lors de l'ajout ou de la suppression de capacités et lors de la migration des données.

Résultat escompté : vous intégrez la sécurité des déploiements automatisés dans le processus de publication grâce à des tests de pré-production approfondis, à des annulations automatiques et à des déploiements de production échelonnés. Cette automatisation minimise l'impact potentiel de l'échec des déploiements sur la production, et les développeurs n'ont plus besoin de surveiller activement les déploiements jusqu'à la production.

Anti-modèles courants :

- Vous effectuez des modifications manuelles.
- Vous sautez des étapes de l'automatisation grâce à des flux de travail d'urgence manuels.
- Vous ne suivez pas les plans et processus établis au profit de délais accélérés.
- Vous effectuez des déploiements de suivi rapides sans prévoir de durée d'intégration.

Avantages du respect de cette bonne pratique : lorsque vous utilisez l'automatisation pour déployer toutes les modifications, vous éliminez le risque d'erreur humaine et vous offrez la possibilité de tester avant de modifier la production. L'exécution de ce processus avant la phase de production permet de vérifier que vos plans sont complets. En outre, la restauration automatique de votre processus de publication peut identifier les problèmes de production et ramener votre charge de travail à son état de fonctionnement antérieur.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : moyen

Directives d'implémentation

Automatisez votre pipeline de déploiement. Le déploiement des pipelines vous permet d'une part d'invoquer des tests automatisés et la détection des anomalies et, d'autre part, d'arrêter le pipeline à une certaine étape avant le déploiement en production ou de restaurer automatiquement l'environnement d'avant la modification. L'adoption de la culture de l'intégration continue et de la livraison/du déploiement continus (CI/CD) en fait partie intégrante. Lors de celle-ci, un commit ou une modification de code passe par différentes étapes automatisées, des étapes de construction et de test au déploiement dans les environnements de production.

Bien que les principes traditionnels suggèrent d'impliquer l'intervention humaine pour les procédures opérationnelles les plus complexes, nous vous conseillons d'automatiser ces mêmes procédures pour cette même raison.

Étapes d'implémentation

Vous pouvez automatiser les déploiements pour supprimer les opérations manuelles en procédant comme suit :

- Configurez un référentiel de code pour stocker votre code en toute sécurité : utilisez un système de gestion de code source hébergé basé sur une technologie populaire telle que Git pour stocker votre code source et votre configuration d'infrastructure en tant que code (IaC).
- Configurez un service d'intégration continue pour compiler votre code source, exécuter des tests et créer des artefacts de déploiement : pour configurer un projet de génération à cette fin, voir Commencer avec l'utilisation de la console par AWS CodeBuild.
- Configurez un service de déploiement qui automatise les déploiements d'applications et gère la complexité des mises à jour des applications sans recourir à des déploiements manuels sujets aux erreurs : AWS CodeDeploy automatise les déploiements de logiciels vers divers services informatiques, tels qu'Amazon EC2 AWS Fargate, AWS Lambda et vos serveurs sur site. Pour configurer ces étapes, consultez Premiers pas avec CodeDeploy.
- Configurez un service de livraison continue qui automatise vos pipelines de publication pour des mises à jour plus rapides et plus fiables des applications et de l'infrastructure : envisagez d'utiliser AWS CodePipeline pour vous aider à automatiser vos pipelines de publication. Pour plus de détails, consultez les didacticiels CodePipeline.

Ressources

Bonnes pratiques associées:

- OPS05-BP04 Utiliser des systèmes de gestion du développement et du déploiement
- OPS05-BP10 Automatiser complètement l'intégration et le déploiement
- OPS06-BP02 Déploiements de tests
- OPS06-BP04 Automatiser les tests et les restaurations

Documents connexes:

- Livraison continue de piles AWS CloudFormation imbriquées à l'aide de AWS CodePipeline
- Partenaire APN : partenaires pouvant vous aider à créer des solutions de déploiement automatisées
- AWS Marketplace: produits pouvant être utilisés pour automatiser vos déploiements

- Automatisez les messages de chat avec les webhooks.
- L'Amazon Builders' Library : Garantir la sécurité des restaurations pendant les déploiements
- L'Amazon Builders' Library : Aller plus vite avec la distribution continue
- Qu'est-ce que AWS CodePipeline?
- Qu'est-ce que CodeDeploy?
- AWS Gestionnaire de correctifs d' Systems Manager
- Qu'est-ce qu'Amazon SES?
- Qu'est-ce qu'Amazon Simple Notification Service?

Vidéos connexes:

- AWS Summit 2019: CI/CD on AWS

Gestion des défaillances

> Des pannes finiront toujours par arriver : des routeurs aux disques durs, des systèmes d'exploitation aux unités de mémoire corrompant des paquets TCP, des erreurs transitoires aux pannes permanentes. C'est inéluctable, que vous utilisiez du matériel de la plus haute qualité ou les composants les moins chers - Werner Vogels, Directeur technique – Amazon.com

Les pannes des composants matériels de bas niveau doivent être traitées au quotidien dans un centre de données sur site. En revanche, dans le cloud, vous devriez être à l'abri de la plupart de ces types de défaillances. Par exemple, les volumes Amazon EBS sont placés dans une zone de disponibilité spécifique où ils sont automatiquement répliqués pour vous protéger de la défaillance d'un seul composant. Tous les volumes EBS sont conçus pour offrir une disponibilité de 99,999 %. Les objets Amazon S3 sont stockés dans au moins trois zones de disponibilité offrant une durabilité des objets de 99,999999999 % sur une année donnée. Quel que soit votre fournisseur de cloud, des défaillances peuvent avoir un impact sur votre charge de travail. Vous devez donc prendre des mesures pour mettre en œuvre la résilience si voulez que votre charge de travail soit fiable.

Pour appliquer les bonnes pratiques présentées ici, vous devez vous assurer que les personnes qui conçoivent, implémentent et exécutent vos charges de travail connaissent les objectifs commerciaux et de fiabilité requis pour y parvenir. Elles doivent maîtriser ces exigences de fiabilité et être formées pour y répondre.

Les sections suivantes expliquent les bonnes pratiques de gestion des pannes afin de prévenir tout impact sur votre charge de travail.

Rubriques

- sauvegarder les données ;
- Utilisation de l'isolation des défaillances pour protéger votre charge de travail
- Conception d'une charge de travail qui résiste aux défaillances des composants
- Test de la fiabilité
- Planification de la reprise après sinistre

sauvegarder les données ;

Sauvegardez les données, les applications et la configuration pour répondre aux exigences relatives à la durée maximale d'interruption admissible (RTO) et aux objectifs de point de reprise (RPO).

Bonnes pratiques

- REL09-BP01 Identifier et sauvegarder toutes les données qui doivent être sauvegardées, ou reproduire les données à partir de sources
- REL09-BP02 Sécuriser et chiffrer les sauvegardes
- REL09-BP03 Effectuer automatiquement la sauvegarde des données
- REL09-BP04 Effectuer une récupération périodique des données pour vérifier l'intégrité et les processus de sauvegarde

REL09-BP01 Identifier et sauvegarder toutes les données qui doivent être sauvegardées, ou reproduire les données à partir de sources

Identifiez et utilisez les fonctionnalités de sauvegarde des services et ressources de données utilisés par votre charge de travail. La plupart des services offrent des fonctionnalités permettant de sauvegarder vos données de charge de travail.

Résultat escompté : les sources de données ont été identifiées et classées en fonction de leur ordre d'importance. Définissez ensuite une stratégie de récupération des données basée sur le RPO. Cette stratégie implique soit de sauvegarder ces sources de données, soit d'avoir la capacité de reproduire des données provenant d'autres sources. En cas de perte de données, la stratégie mise en place permet la récupération ou la reproduction des données dans les RPO et RTO définis.

Phase de maturité du cloud : fondamentale

Anti-modèles courants :

- Ne pas connaître toutes les sources de données pour la charge de travail ni leur ordre d'importance.
- Ne pas effectuer de sauvegardes des sources de données critiques.
- Sauvegarder uniquement certaines sources de données sans utiliser leur ordre d'importance comme critère.
- Aucun RPO défini, ou la fréquence de sauvegarde ne parvient pas à atteindre le RPO.

- Ne pas évaluer si une sauvegarde est nécessaire ou si les données peuvent être reproduites à partir d'autres sources.

Avantages liés au respect de cette bonne pratique : identifier les emplacements où les sauvegardes sont nécessaires et mettre en place un mécanisme pour créer des sauvegardes, ou être capable de reproduire les données à partir d'une source externe améliore la capacité de restauration et de récupération des données lors d'une panne.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : élevé

Directives d'implémentation

Tous les magasins de données AWS offrent des fonctionnalités de sauvegarde. Des services comme Amazon RDS et Amazon DynamoDB prennent également en charge la sauvegarde automatisée qui permet la reprise ponctuelle (PITR). Vous pouvez ainsi restaurer une sauvegarde remontant jusqu'à cinq minutes ou moins avant l'heure actuelle. De nombreux services AWS offrent la possibilité de copier des sauvegardes vers une autre Région AWS. AWS Backup est un outil qui vous permet de centraliser et d'automatiser la protection des données dans l'ensemble des services AWS. Reprise après sinistre AWS Elastic vous permet de copier les charges de travail complètes du serveur et de maintenir une protection continue des données sur site, entre les zones d'exploitation ou entre les régions, avec un objectif de point de reprise (RPO) mesuré en secondes.

Amazon S3 peut être utilisé comme destination de sauvegarde pour les sources de données autogérées et gérées par AWS. Les services AWS tels qu'Amazon EBS, Amazon RDS et Amazon DynamoDB ont des fonctionnalités intégrées permettant de créer des sauvegardes. Vous pouvez aussi utiliser des logiciels de sauvegarde tiers.

Les données sur site peuvent être sauvegardées dans le AWS Cloud avec AWS Storage Gateway ou AWS DataSync. Les compartiments Amazon S3 peuvent être utilisés pour stocker ces données sur AWS. Amazon S3 propose plusieurs niveaux de stockage tels qu'Amazon Glacier ou Amazon Glacier Deep Archive pour réduire les coûts du stockage de données.

Il se peut que vous puissiez répondre aux besoins de récupération de données en reproduisant les données à partir d'autres sources. Par exemple, les nœuds de réplication Amazon ElastiCache ou les répliques de lecture Amazon RDS peuvent être utilisés pour reproduire des données en cas de perte du nœud principal. Dans les cas où de telles sources peuvent être utilisées pour atteindre votre objectif de point de reprise (RPO) et votre objectif de délai de reprise (RTO), il se peut que vous n'ayez pas besoin d'une sauvegarde. Autre exemple, si vous travaillez avec Amazon EMR, il n'est

peut-être pas nécessaire de sauvegarder votre magasin de données HDFS, tant que vous pouvez reproduire les données dans Amazon EMR à partir d'Amazon S3.

Lors de la sélection d'une stratégie de sauvegarde, tenez compte du temps nécessaire pour récupérer les données. Le temps nécessaire pour récupérer les données dépend du type de sauvegarde (dans le cas d'une stratégie de sauvegarde) ou de la complexité du mécanisme de reproduction des données. Cette durée doit être conforme au RTO de la charge de travail.

Étapes d'implémentation

1. Identifiez toutes les sources de données pour la charge de travail. Les données peuvent être stockées sur un certain nombre de ressources telles que les bases de données, les volumes, les systèmes de fichiers, les systèmes de journalisation et le stockage d'objets. Reportez-vous à la section Ressources pour trouver des documents connexes sur les différents services AWS où les données sont stockées et sur la capacité de sauvegarde que ces services fournissent.
2. Classez les sources de données en fonction de leur ordre d'importance. Différents jeux de données ont différents niveaux d'importance pour une charge de travail, et donc différentes exigences en matière de résilience. Par exemple, certaines données peuvent être critiques et nécessiter un RPO proche de zéro, tandis que d'autres données peuvent être moins critiques et peuvent tolérer un RPO plus élevé et la perte de certaines données. De même, différents jeux de données peuvent également avoir des exigences de RTO différentes.
3. Utilisez AWS ou des services tiers pour créer des sauvegardes des données. AWS Backup est un service géré qui permet de créer des sauvegardes de différentes sources de données sur AWS. Reprise après sinistre AWS Elastic gère la réplication automatique des données en moins d'une seconde vers un Région AWS. La plupart des services AWS ont également des fonctionnalités natives permettant de créer des sauvegardes. AWS Marketplace inclut de nombreuses solutions qui offrent également ces fonctionnalités. Consultez Ressources ci-dessous pour découvrir comment créer des sauvegardes de données à partir de divers services AWS.
4. Pour les données non sauvegardées, définissez un mécanisme de reproduction des données. Vous pouvez choisir de ne pas sauvegarder les données qui peuvent être reproduites à partir d'autres sources pour diverses raisons. Il peut arriver qu'il soit moins coûteux de reproduire des données à partir de sources en cas de besoin plutôt que de créer une sauvegarde, car le stockage des sauvegardes peut impliquer un coût. Ou peut-être la restauration à partir d'une sauvegarde prend-elle plus de temps que la reproduction des données à partir des sources, ce qui entraîne une violation du RTO. Dans de telles situations, envisagez les avantages et inconvénients de chaque approche et définissez un processus clair sur la façon dont les données peuvent être reproduites à partir de ces sources lorsque la récupération des données est nécessaire. Par

exemple, si vous avez chargé des données depuis Amazon S3 vers un entrepôt de données (comme Amazon Redshift) ou un cluster MapReduce (comme Amazon EMR) pour les analyser, vous disposez d'un exemple de données reproductibles à partir d'autres sources. Tant que les résultats de ces analyses sont stockés quelque part ou reproductibles, vous ne perdrez pas données en cas de défaillance de l'entrepôt de données ou du cluster MapReduce. Parmi les autres exemples reproductibles à partir de sources, figurent les caches (comme Amazon ElastiCache) ou les réplicas en lecture RDS.

5. Spécifiez un rythme de sauvegarde des données. La création de sauvegardes de sources de données est un processus périodique, et la fréquence doit dépendre du RPO.

Niveau d'effort du plan d'implémentation : modéré

Ressources

Bonnes pratiques associées :

REL13-BP01 Définir les objectifs de reprise pour les temps d'arrêt et les pertes de données

REL13-BP02 Utiliser des stratégies de reprise définies pour répondre aux objectifs de reprise

Documents connexes :

- Qu'est-ce que AWS Backup?
- Qu'est-ce qu'AWS DataSync ?
- Qu'est-ce que la sauvegarde en volumes ?
- Partenaire APN : partenaires pouvant faciliter la sauvegarde
- AWS Marketplace : produits pouvant être utilisés pour la sauvegarde
- Instantanés Amazon EBS
- Sauvegarde d'Amazon EFS
- Sauvegarde d'Amazon FSx for Windows File Server
- Backup et restauration d'ElastiCache for Redis
- Création d'un instantané de cluster de bases de données dans Neptune
- Création d'un instantané de base de données
- Création d'une règle EventBridge qui se déclenche selon un calendrier
- Réplication entre Régions avec Amazon S3
- EFS-to-EFS AWS Backup

- Exporter les données du journal vers Amazon S3
- Gestion du cycle de vie des objets
- Sauvegarde et restauration à la demande pour DynamoDB
- Restauration à un instant dans le passé pour DynamoDB
- Travailler avec des instantanés d'index Amazon OpenSearch Service
- Présentation de Reprise après sinistre AWS Elastic

Vidéos connexes :

- AWS re:Invent 2021 - Backup, disaster recovery, and ransomware protection with AWS
- AWS Backup Demo: Cross-Account and Cross-Region Backup
- AWS re:Invent 2019: Deep dive on AWS Backup, ft. Rackspace (STG341)

REL09-BP02 Sécuriser et chiffrer les sauvegardes

Contrôlez et détectez l'accès aux sauvegardes à l'aide de l'authentification et de l'autorisation. Assurez la prévention et détectez si l'intégrité des données des sauvegardes est compromise à l'aide du chiffrement.

Mettez en œuvre des contrôles de sécurité pour empêcher tout accès non autorisé aux données de sauvegarde. Chiffrez les sauvegardes pour protéger la confidentialité et l'intégrité de vos données.

Anti-modèles courants :

- Avoir le même accès aux sauvegardes et à l'automatisation de la restauration que vous le faites pour les données.
- Ne pas chiffrer vos sauvegardes.
- Ne pas mettre en œuvre l'immuabilité pour la protection contre la suppression ou la falsification.
- Utiliser le même domaine de sécurité pour les systèmes de production et de sauvegarde.
- Ne pas valider l'intégrité de la sauvegarde par le biais de tests réguliers.

Avantages liés au respect de cette bonne pratique :

- La sécurisation de vos sauvegardes empêche la falsification des données. De même, le chiffrement des données empêche l'accès à ces données si elles sont accidentellement exposées.

- Protection améliorée contre les rançongiciels et autres cybermenaces ciblant l'infrastructure de sauvegarde.
- Réduction du temps de restauration suite à un cyberincident grâce à des processus de restauration validés.
- Amélioration des capacités de continuité des activités lors d'incidents de sécurité.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : élevé

Directives d'implémentation

Contrôlez et détectez l'accès aux sauvegardes à l'aide de l'authentification et de l'autorisation, telles qu'Gestion des identités et des accès AWS (IAM). Assurez la prévention et détectez si l'intégrité des données des sauvegardes est compromise à l'aide du chiffrement.

Amazon S3 prend en charge plusieurs méthodes de chiffrement de vos données au repos. Grâce au chiffrement côté serveur, Amazon S3 accepte vos objets sous forme de données non chiffrées, puis les chiffre lors de leur stockage. Avec le chiffrement côté client, votre application de charge de travail s'occupe du chiffrement des données avant leur transmission à Amazon S3. Ces deux méthodes vous permettent d'utiliser AWS Key Management Service (AWS KMS) pour créer et stocker la clé de données. Vous pouvez également fournir votre propre clé (vous en assumez alors la responsabilité). Avec AWS KMS, vous pouvez définir des stratégies via IAM pour déterminer qui peut ou non accéder à vos clés de données et à vos données déchiffrées.

Pour Amazon RDS, si vous avez choisi de chiffrer vos bases de données, vos sauvegardes sont également chiffrées. Les sauvegardes DynamoDB sont toujours chiffrées. En utilisant Reprise après sinistre AWS Elastic, toutes les données en transit et au repos sont chiffrées. Avec Elastic Disaster Recovery, les données au repos peuvent être chiffrées à l'aide de la clé Amazon EBS de chiffrement de volume par défaut ou d'une clé personnalisée gérée par le client.

Considérations relatives à la cyberrésilience

Pour améliorer la sécurité des sauvegardes contre les cybermenaces, envisagez de mettre en œuvre ces contrôles supplémentaires en plus du chiffrement :

- Mettez en œuvre l'immuabilité à l'aide du verrouillage du coffre-fort AWS Backup ou du verrouillage d'objet Amazon S3 pour empêcher la modification ou la suppression des données de sauvegarde pendant leur période de conservation, vous protégeant ainsi contre les rançongiciels et les suppressions malveillantes.

- Établissez une isolation logique entre les environnements de production et de sauvegarde grâce à un coffre-fort à isolation logique AWS Backup pour les systèmes critiques, créant ainsi une séparation qui empêche la compromission simultanée des deux environnements.
- Validez régulièrement l'intégrité des sauvegardes à l'aide de tests de restauration AWS Backup pour vérifier que les sauvegardes ne sont pas corrompues et peuvent être restaurées avec succès à la suite d'un cyberincident.
- Mettez en œuvre l'approbation multipartite pour les opérations de restauration critiques en utilisant l'approbation multipartite AWS Backup pour empêcher les tentatives de restauration non autorisées ou malveillantes en exigeant l'autorisation de plusieurs approbateurs désignés.

Étapes d'implémentation

1. Utilisez le chiffrement sur chacun de vos magasins de données. La sauvegarde est également chiffrée si vos données sources le sont.
   - Utilisez le chiffrement dans Amazon RDS. Vous pouvez configurer le chiffrement au repos à l'aide de AWS Key Management Service lorsque vous créez une instance RDS.
   - Chiffrez les volumes Amazon EBS. Vous pouvez configurer le chiffrement par défaut ou spécifier une clé unique lors de la création du volume.
   - Utilisez le chiffrement Amazon DynamoDB requis. DynamoDB chiffre toutes les données au repos. Vous pouvez utiliser une clé AWS KMS détenue par AWS ou une clé KMS gérée par AWS, en spécifiant une clé stockée dans votre compte.
   - Chiffrez vos données stockées dans Amazon EFS. Configurez le chiffrement lorsque vous créez votre système de fichiers.
   - Configurez le chiffrement dans les régions source et de destination. Vous pouvez configurer le chiffrement au repos dans Amazon S3 à l'aide de clés stockées dans KMS, mais les clés sont spécifiques à la région. Vous pouvez spécifier les clés de destination lorsque vous configurez la réplication.
   - Choisissez d'utiliser le chiffrement Amazon EBS par défaut ou personnalisé pour Elastic Disaster Recovery. Cette option permet de chiffrer les données au repos répliquées sur les disques du sous-réseau de la zone de transit et sur les disques répliqués.
2. Mettez en œuvre les autorisations de moindre privilège pour accéder à vos sauvegardes. Suivez les bonnes pratiques pour limiter l'accès aux sauvegardes, instantanés et réplicas conformément aux bonnes pratiques de sécurité.

3. Configurez l'immuabilité pour les sauvegardes critiques. Pour les données critiques, implémentez le verrouillage du coffre-fort AWS Backup ou le verrouillage d'objet S3 pour empêcher leur suppression ou leur modification pendant la période de conservation spécifiée. Pour plus de détails sur l'implémentation, consultez AWS Backup Vault Lock.
4. Créez une séparation logique pour les environnements de sauvegarde. Mettez en place un coffre-fort à isolation logique AWS Backup pour les systèmes critiques nécessitant une protection renforcée contre les cybermenaces. Pour obtenir des conseils de mise en œuvre, consultez Création de la cyber-résilience à l'aide d'un coffre-fort à isolation logique AWS Backup.
5. Mettez en œuvre des processus de validation des sauvegardes. Configurez les tests de restauration AWS Backup pour vérifier régulièrement que les sauvegardes ne sont pas corrompues et peuvent être restaurées avec succès à la suite d'un cyberincident. Pour plus d'informations, consultez Validation de l'état de préparation à la restauration à l'aide de tests de restauration AWS Backup.
6. Configurez l'approbation multipartite pour les opérations de restauration sensibles. Pour les systèmes critiques, mettez en œuvre une approbation multipartite AWS Backup afin d'exiger l'autorisation de plusieurs approbateurs désignés avant de procéder à la restauration. Pour plus de détails sur la mise en œuvre, consultez Amélioration de la résilience à la reprise grâce à la prise en charge de l'approbation multipartite par AWS Backup.

Ressources

Documents connexes:

- AWS Marketplace: produits pouvant être utilisés pour la sauvegarde
- Chiffrement Amazon EBS
- Amazon S3 : protection des données à l'aide du chiffrement
- Configuration supplémentaire de la réplication entre régions : réplication d'objets créés avec le chiffrement côté serveur (SSE) à l'aide de clés de chiffrement stockées dans AWS KMS
- Chiffrement de DynamoDB au repos
- Chiffrement des ressources Amazon RDS
- Chiffrement des données et métadonnées dans Amazon EFS
- Chiffrement pour les sauvegardes dans AWS
- Gestion des tables chiffrées
- Pilier Sécurité - AWS Well-Architected Framework

- Présentation de Reprise après sinistre AWS Elastic?
- FSISEC11 : Comment vous protégez-vous contre les rançongiciels ?
- Gestion des risques liés aux rançongiciels sur AWS à l'aide du NIST Cybersecurity Framework (CSF)
- Création de la cyber-résilience à l'aide d'un coffre-fort à isolation logique AWS Backup
- Validation de l'état de préparation à la restauration à l'aide de tests de restauration AWS Backup
- Amélioration de la résilience à la reprise grâce à la prise en charge de l'approbation multipartite par AWS Backup

Exemples connexes :

- Implémentation de la réplication entre régions bidirectionnelle (CRR) pour Amazon S3

REL09-BP03 Effectuer automatiquement la sauvegarde des données

Configurez les sauvegardes à effectuer automatiquement en fonction d'un calendrier périodique informé par l'objectif de point de reprise (RPO) ou par les modifications du jeu de données. Les jeux de données critiques dont le seuil de tolérance pour la perte de données est faible doivent être sauvegardés automatiquement et fréquemment, tandis que les données moins critiques où certaines données peuvent être perdues peuvent être sauvegardées moins fréquemment.

Résultat escompté : un processus automatisé qui crée des sauvegardes de sources de données à une cadence établie.

Anti-modèles courants :

- Exécution manuelle des sauvegardes.
- Utilisation de ressources qui ont une capacité de sauvegarde, mais sans inclure la sauvegarde dans votre automatisation.

Avantages du respect de cette bonne pratique : l'automatisation des sauvegardes garantit qu'elles sont effectuées régulièrement en fonction de votre RPO et vous alerte dans le cas contraire.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : moyen

Directives d'implémentation

AWS Backup peut être utilisé pour créer des sauvegardes de données automatisées de diverses sources de données AWS. Les instances Amazon RDS peuvent être sauvegardées presque en continu toutes les cinq minutes, et les objets Amazon S3 peuvent être sauvegardés presque en continu toutes les quinze minutes, ce qui permet une reprise ponctuelle (PITR) à un moment précis dans l'historique de sauvegarde. Pour les autres sources de données AWS, telles que les volumes Amazon EBS, les tables Amazon DynamoDB ou les systèmes de fichiers Amazon FSx, AWS Backup peut exécuter une sauvegarde automatique qui peut avoir lieu toutes les heures. Ces services offrent également des fonctionnalités de sauvegarde natives. Les services AWS proposant une sauvegarde automatisée avec restauration à un moment précis incluent Amazon DynamoDB, Amazon RDS et Amazon Keyspaces (pour Apache Cassandra), qui peuvent être restaurés à un moment précis dans l'historique des sauvegardes. La plupart des autres services de stockage de données AWS offrent la possibilité de planifier des sauvegardes périodiques, à une fréquence de sauvegarde pouvant atteindre toutes les heures.

Amazon RDS et Amazon DynamoDB autorisent une sauvegarde continue avec la restauration à un instant donné dans le passé. Une fois activée, la gestion des versions Amazon S3 est automatique. Vous pouvez utiliser Amazon Data Lifecycle Manager pour automatiser la création, la copie et la suppression d'instantanés Amazon EBS. Il peut également automatiser la création, la copie, l'abandon et le désenregistrement des Amazon Machine Images (AMI) basées sur Amazon EBS et de leurs instantanés Amazon EBS sous-jacents.

Reprise après sinistre AWS Elastic fournit une réplication continue au niveau des blocs depuis l'environnement source (sur site ou sur AWS) vers la région de reprise cible. Des instantanés Amazon EBS ponctuels sont automatiquement créés et gérés par le service.

AWS Backup fournit une solution de sauvegarde entièrement gérée basée sur des stratégies afin d'offrir une vue centralisée de l'automatisation et de l'historique de vos sauvegardes. Cette solution centralise et automatise la sauvegarde des données sur plusieurs services AWS dans le cloud et sur site à l'aide d'AWS Storage Gateway.

Outre la gestion des versions, Amazon S3 intègre la réplication. L'intégralité du compartiment S3 peut être automatiquement répliquée vers un autre compartiment de la même ou d'une autre Région AWS.

Étapes d'implémentation

1. Identifiez les sources de données qui sont actuellement sauvegardées manuellement. Pour en savoir plus, veuillez consulter REL09-BP01 Identifier et sauvegarder toutes les données qui doivent être sauvegardées, ou reproduire les données à partir de sources.
2. Déterminez le RPO pour la charge de travail. Pour en savoir plus, veuillez consulter REL13-BP01 Définir les objectifs de reprise pour les temps d'arrêt et les pertes de données.
3. Utilisez une solution de sauvegarde automatisée ou un service géré. AWS Backup est un service entièrement géré qui vous permet de facilement centraliser et automatiser la protection des données sur les services AWS dans le cloud et sur site. En utilisant les plans de sauvegarde dans AWS Backup, créez des règles qui définissent les ressources à sauvegarder, et la fréquence à laquelle ces sauvegardes doivent être créées. Cette fréquence doit être informée par le RPO établi à l'étape 2. Pour obtenir des conseils pratiques sur la façon de créer des sauvegardes automatisées à l'aide de AWS Backup, consultez la section Test de la sauvegarde et de la restauration de données. Les fonctionnalités de sauvegarde natives sont offertes par la plupart des services AWS qui stockent des données. Par exemple, RDS peut être exploité pour les sauvegardes automatisées avec une reprise ponctuelle (PITR).
4. Pour les sources de données non prises en charge par une solution de sauvegarde automatisée ou un service géré tel que des sources de données sur site ou des files d'attente de messages, envisagez d'utiliser une solution tierce de confiance pour créer des sauvegardes automatisées. Vous pouvez également créer une automatisation pour ce faire avec AWS CLI ou les kits SDK. Vous pouvez utiliser les fonctions AWS Lambda ou AWS Step Functions pour définir la logique impliquée dans la création d'une sauvegarde de données, et exploiter Amazon EventBridge pour l'invoquer à une fréquence basée sur votre RPO.

Niveau d'effort du plan d'implémentation : faible

Ressources

Documents connexes :

- Partenaire APN : partenaires pouvant faciliter la sauvegarde
- AWS Marketplace : produits pouvant être utilisés pour la sauvegarde
- Création d'une règle EventBridge qui se déclenche selon un calendrier
- Présentation de AWS Backup
- Présentation de AWS Step Functions
- Présentation de Reprise après sinistre AWS Elastic

Vidéos connexes :

- AWS re:Invent 2019: Deep dive on AWS Backup, ft. Rackspace (STG341)

REL09-BP04 Effectuer une récupération périodique des données pour vérifier l'intégrité et les processus de sauvegarde

Confirmez que l'implémentation de votre processus de sauvegarde répond à vos objectifs de délai de reprise (RTO) et à vos objectifs de point de reprise (RPO) en effectuant un test de reprise.

Résultat escompté : les données des sauvegardes sont périodiquement récupérées à l'aide de mécanismes bien définis pour garantir que la récupération est conforme à l'objectif de délai de reprise (RTO) établi pour la charge de travail. Vérifiez que la restauration à partir d'une sauvegarde aboutit à une ressource qui contient les données d'origine sans qu'aucune d'entre elles ne soit corrompue ou inaccessible, et avec une perte de données conforme à l'objectif de point de reprise (RPO).

Anti-modèles courants :

- Restauration d'une sauvegarde, mais sans interroger ou récupérer des données pour vérifier l'utilisation de la restauration.
- Supposer qu'une sauvegarde existe.
- Supposer que la sauvegarde d'un système est pleinement opérationnelle et que les données peuvent être récupérées à partir de celle-ci.
- Supposer que le temps de restauration ou de récupération des données à partir d'une sauvegarde est conforme au RTO de la charge de travail.
- Supposer que les données contenues dans la sauvegarde sont conformes au RPO de la charge de travail.
- Effectuez une restauration si nécessaire, sans utiliser de runbook ou en dehors d'une procédure automatisée établie.

Avantages du respect de cette bonne pratique : le test de la récupération des sauvegardes garantit que les données peuvent être restaurées en cas de besoin sans craindre qu'elles soient manquantes ou corrompues, que la restauration et la récupération sont possibles conformément au RTO de la charge de travail et que toute perte de données est conforme au RPO de la charge de travail.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : moyen

Directives d'implémentation

Tester la fonctionnalité de sauvegarde et de restauration permet de garantir que ces actions peuvent être effectuées pendant une panne. Restaurez périodiquement les sauvegardes vers un nouvel emplacement et exécutez des tests pour vérifier l'intégrité des données. Certains tests courants doivent être effectués pour vérifier que toutes les données sont disponibles, qu'elles ne sont pas corrompues, qu'elles sont accessibles et que toute perte de données ne dépasse pas le RPO de la charge de travail. Ces tests peuvent également contribuer à déterminer si les mécanismes de récupération sont suffisamment rapides pour s'adapter au RTO de la charge de travail.

Avec AWS, vous pouvez mettre en place un environnement de test et restaurer vos sauvegardes afin d'évaluer le RTO et le RPO, et exécuter des tests sur le contenu et l'intégrité des données.

De plus, Amazon RDS et Amazon DynamoDB permettent une reprise ponctuelle (PITR). Grâce à la sauvegarde continue, vous pouvez restaurer votre jeu de données à l'état dans lequel il était à une date et une heure spécifiées.

Si toutes les données sont disponibles, si elles ne sont pas corrompues, si elles sont accessibles et si toute perte de données est conforme au RPO de la charge de travail. Ces tests peuvent également contribuer à déterminer si les mécanismes de récupération sont suffisamment rapides pour s'adapter au RTO de la charge de travail.

Reprise après sinistre AWS Elastic offre des instantanés de récupération ponctuelle et continue des volumes Amazon EBS. Au fur et à mesure que les serveurs source sont répliqués, les états ponctuels sont chroniqués dans le temps en fonction de la politique configurée. Elastic Disaster Recovery vous aide à vérifier l'intégrité de ces instantanés en lançant des instances à des fins de test et d'analyse sans rediriger le trafic.

Étapes d'implémentation

1. Identifiez les sources de données qui sont actuellement sauvegardées et où ces sauvegardes sont stockées. Pour obtenir des conseils de mise en œuvre, consultez REL09-BP01 Identifier et sauvegarder toutes les données qui doivent être sauvegardées, ou reproduire les données à partir de sources.
2. Établissez des critères de validation des données pour chaque source de données. Différents types de données ont des propriétés différentes qui pourraient nécessiter des mécanismes de validation distincts. Réfléchissez à la manière dont ces données pourraient être validées avant de vous assurer que vous pouvez les utiliser en production. Certaines méthodes courantes

de validation des données consistent à utiliser des propriétés de données et de sauvegarde telles que le type de données, le format, la somme de contrôle, la taille ou une combinaison de ces propriétés avec une logique de validation personnalisée. Par exemple, il peut s'agir d'une comparaison des valeurs de somme de contrôle entre la ressource restaurée et la source de données au moment de la création de la sauvegarde.

3. Établissez un RTO et un RPO de sorte à restaurer les données en fonction de l'ordre d'importance des données. Pour obtenir des conseils de mise en œuvre, consultez REL13-BP01 Définir les objectifs de reprise pour les temps d'arrêt et les pertes de données.
4. Évaluez votre capacité de récupération. Passez en revue votre stratégie de sauvegarde et de restauration pour déterminer si elle peut répondre au RTO et au RPO, et ajustez la stratégie si nécessaire. À l'aide de AWS Resilience Hub, vous pouvez exécuter une évaluation de votre charge de travail. Celle-ci se concentre sur la configuration de votre application par rapport à la stratégie de résilience et signale si vos objectifs RTO et RPO peuvent être atteints.
5. Effectuez un test de restauration avec les processus établis utilisés en production pour la restauration des données. Ces processus dépendent de la façon dont la source de données d'origine a été sauvegardée, du format et de l'emplacement de stockage de la sauvegarde elle-même, ou ils varient selon que les données sont reproduites à partir d'autres sources. Par exemple, si vous utilisez un service géré tel que AWS Backup, cela peut être aussi simple que de restaurer la sauvegarde dans une nouvelle ressource. Si vous avez utilisé Reprise après sinistre AWS Elastic vous pouvez lancer une simulation de récupération.
6. Validez la récupération des données à partir de la ressource restaurée en fonction des critères que vous avez définis précédemment pour la validation des données. Les données restaurées et récupérées contiennent-elles l'enregistrement/l'élément le plus récent au moment de la sauvegarde ? Ces données sont-elles conformes au RPO de la charge de travail ?
7. Mesurez le temps nécessaire à la restauration et à la récupération et comparez-le à votre RTO établi. Ce processus est-il conforme au RTO de la charge de travail ? Par exemple, comparez les horodatages du début du processus de restauration et de la fin de la validation de la récupération pour calculer la durée de ce processus. Tous les appels d'API AWS sont horodatés, et ces informations sont disponibles dans AWS CloudTrail. Bien que ces informations puissent fournir des détails sur le début du processus de restauration, l'horodatage indiquant la fin de la validation doit être enregistré par votre logique de validation. Si vous utilisez un processus automatisé, des services tels qu'Amazon DynamoDB peuvent être utilisés pour stocker ces informations. En outre, de nombreux services AWS fournissent un historique des événements qui contient des informations horodatées indiquant quand certaines actions se sont produites. Dans AWS Backup, les actions de sauvegarde et de restauration sont appelées tâches, et ces tâches contiennent des

informations d'horodatage dans le cadre de leurs métadonnées qui peuvent être utilisées pour mesurer le temps nécessaire à la restauration et à la récupération.

8. Informez les parties prenantes si la validation des données échoue ou si le temps requis pour la restauration et la récupération dépasse le RTO établi pour la charge de travail. Lors de la mise en œuvre de l'automatisation à cette fin, comme dans cet atelier, des services tels qu'Amazon Simple Notification Service (Amazon SNS) peuvent être utilisés pour envoyer des notifications push telles que des e-mails ou des SMS aux parties prenantes. Ces messages peuvent également être publiés sur des applications de messagerie telles qu'Amazon Chime, Slack ou Microsoft Teams ou utilisés pour créer des tâches sous forme d'OpsiTems à l'aide d'AWS Systems Manager OpsCenter.
9. Automatisez ce processus pour qu'il s'exécute périodiquement. Par exemple, des services comme AWS Lambda ou une machine d'état dans AWS Step Functions peuvent être utilisés pour automatiser les processus de restauration et de récupération, et Amazon EventBridge peut être utilisé pour invoquer périodiquement ce flux de travail d'automatisation, comme indiqué dans le diagramme d'architecture ci-dessous. Découvrez comment automatiser la validation de récupération de données avec AWS Backup. De plus, cet atelier Well-Architected apporte une expérience pratique sur une façon d'automatiser plusieurs des étapes indiquées ici.

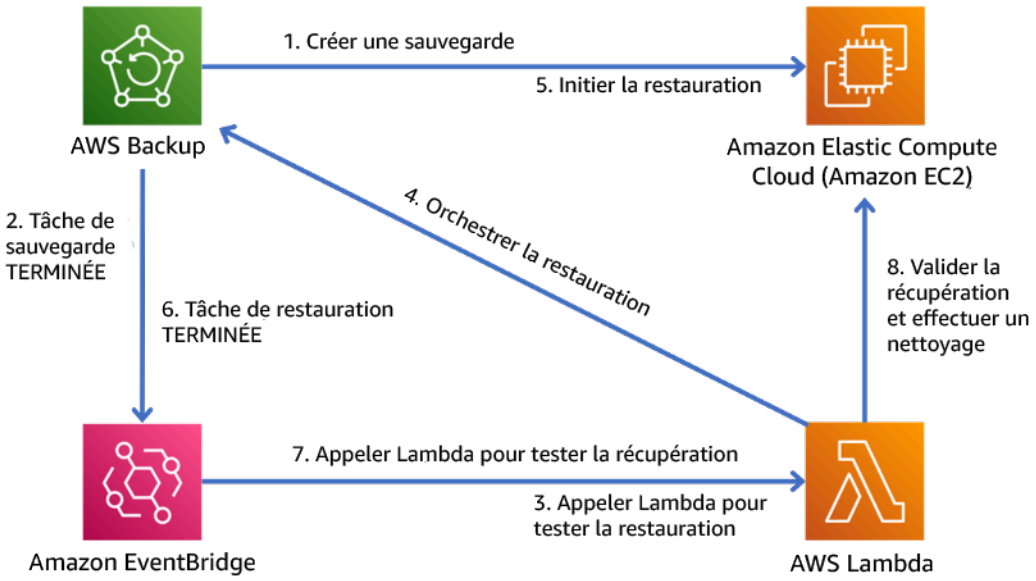


Figure 9. Un processus de sauvegarde et de restauration automatisé

Niveau d'effort pour le plan de mise en œuvre : modéré à élevé selon la complexité des critères de validation.

Ressources

Documents connexes :

- Automatiser la validation de la récupération des données avec AWS Backup
- Partenaire APN : partenaires pouvant faciliter la sauvegarde
- AWS Marketplace : produits pouvant être utilisés pour la sauvegarde
- Création d'une règle EventBridge qui se déclenche selon un calendrier
- Sauvegarde et restauration à la demande pour DynamoDB
- Qu'est-ce que AWS Backup?
- Qu'est-ce que AWS Step Functions?
- Présentation de Reprise après sinistre AWS Elastic
- Reprise après sinistre AWS Elastic

Utilisation de l'isolation des défaillances pour protéger votre charge de travail

L'isolation des défaillances limite l'impact de la défaillance d'un composant ou d'un système à une limite définie. Si l'isolation est correcte, les composants situés en dehors de cette limite ne sont pas affectés par la défaillance. L'exécution de votre charge de travail au-delà de plusieurs limites d'isolation des défaillances peut la rendre plus résistante aux défaillances.

Bonnes pratiques

- REL10-BP01 Déploiement de la charge de travail sur plusieurs emplacements
- REL10-BP02 Automatiser la récupération des composants limités à un seul emplacement
- REL10-BP03 Utiliser des architectures cloisonnées pour limiter la portée de l'impact

REL10-BP01 Déploiement de la charge de travail sur plusieurs emplacements

Distribuez les données et les ressources de charge de travail sur plusieurs zones de disponibilité ou, si nécessaire, entre Régions AWS.

L'un des principes fondamentaux de la conception de services dans AWS est d'éviter les points de défaillance uniques, y compris l'infrastructure physique sous-jacente. AWS fournit des ressources et des services de cloud computing à l'échelle mondiale, sur plusieurs sites géographiques appelés régions. Chaque région est physiquement et logiquement indépendante et se compose de trois zones de disponibilité (AZ) ou plus. Les zones de disponibilité sont géographiquement proches les unes des autres, mais sont physiquement séparées et isolées. La répartition de vos charges de travail entre les zones de disponibilité et les régions vous permet de réduire le risque de menaces telles que les incendies, les inondations, les catastrophes météorologiques, les tremblements de terre et les erreurs humaines.

Créez une stratégie de localisation pour assurer une haute disponibilité adaptée à vos charges de travail.

Résultat escompté : les charges de travail de production sont réparties entre plusieurs zones de disponibilité (AZ) ou régions afin de garantir la tolérance aux pannes et la haute disponibilité.

Anti-modèles courants :

- Votre charge de travail de production n'existe que dans une seule zone de disponibilité.
- Vous mettez en œuvre une architecture multirégionale alors qu'une architecture multi-AZ répondrait aux exigences.
- Vos déploiements ou vos données sont désynchronisés, ce qui entraîne une dérive de la configuration ou une sous-réplication des données.
- Vous ne tenez pas compte des dépendances entre les composants de l'application si les exigences en matière de résilience et de multi-localisation diffèrent entre ces composants.

Avantages liés au respect de cette bonne pratique :

- Votre charge de travail est plus résiliente face aux incidents, tels que les pannes d'alimentation, les défaillances de contrôle environnemental, les catastrophes naturelles, les pannes de service en amont ou les problèmes réseau affectant une zone de disponibilité ou une région entière.
- Vous pouvez accéder à un inventaire plus large d'instances Amazon EC2 et réduire le risque d'exceptions InsufficientCapacityExceptions (ICE) lors du lancement de types d'instances EC2 spécifiques.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : élevé

Directives d'implémentation

Déployez et gérez toutes les charges de travail de production dans au moins deux zones de disponibilité (AZ) d'une région.

Utilisation de plusieurs zones de disponibilité

Les zones de disponibilité sont des lieux d'hébergement de ressources qui sont physiquement séparés les uns des autres afin d'éviter les défaillances corrélées dues à des risques tels que les incendies, les inondations et les tornades. Chaque zone de disponibilité possède une infrastructure physique indépendante comprenant des connexions électriques, des sources d'alimentation de secours, des services mécaniques et une connectivité réseau. Cette disposition limite les défaillances d'un de ces composants à la seule zone de disponibilité affectée. Par exemple, si un incident à l'échelle de la zone de disponibilité rend les instances EC2 indisponibles dans la zone de disponibilité affectée, vos instances situées dans une autre zone de disponibilité restent disponibles.

Bien que physiquement séparées, les zones de disponibilité situées dans la même Région AWS sont suffisamment proches pour fournir une mise en réseau à haut débit et à faible latence (moins de dix millisecondes). Vous pouvez répliquer les données de manière synchrone entre les zones de disponibilité pour la plupart des charges de travail sans affecter de manière significative l'expérience utilisateur. Cela signifie que vous pouvez utiliser les zones de disponibilité d'une région dans une configuration active/active ou active/veille.

Tous les calculs associés à votre charge de travail doivent être répartis entre les différentes zones de disponibilité. Cela inclut les instances Amazon EC2, les tâches AWS Fargate et les fonctions AWS Lambda associées au VPC. Les services de calcul AWS, notamment EC2 Auto Scaling, Amazon Elastic Container Service (ECS) et Amazon Elastic Kubernetes Service (EKS), vous permettent de lancer et de gérer les calculs sur l'ensemble des zones de disponibilité. Configurez-les pour remplacer automatiquement les calculs selon les besoins dans une autre zone de disponibilité afin de maintenir la disponibilité. Pour diriger le trafic vers les zones de disponibilité disponibles, placez un équilibreur de charge devant vos ressources de calcul, tel qu'un Application Load Balancer ou un Network Load Balancer. Les équilibreurs de charge AWS peuvent rediriger le trafic vers les instances disponibles en cas d'altération de la zone de disponibilité.

Vous devez également répliquer les données pour votre charge de travail et les rendre disponibles dans plusieurs zones de disponibilité. Certains services de données AWS gérés, tels qu'Amazon S3, Amazon Elastic File Service (EFS), Amazon Aurora, Amazon DynamoDB, Amazon Simple Queue Service (SQS) et Amazon Kinesis Data Streams répliquent les données dans plusieurs zones de

disponibilité par défaut et résistent à l'altération de la zone de disponibilité. Avec d'autres services de données AWS gérés, tels qu'Amazon Relational Database Service (RDS), Amazon Redshift et Amazon ElastiCache, vous devez activer la réplication multi-AZ. Une fois l'option activée, ces services détectent automatiquement une altération de la zone de disponibilité, redirigent les demandes vers une zone de disponibilité disponible et répliquent à nouveau les données selon les besoins après reprise, sans intervention du client. Familiarisez-vous avec le guide de l'utilisateur de chaque service de données AWS géré que vous utilisez pour comprendre ses fonctionnalités, ses comportements et son fonctionnement multi-AZ.

Si vous utilisez un stockage autogéré, tel que les volumes Amazon Elastic Block Store (EBS) ou le stockage d'instances Amazon EC2, vous devez gérer vous-même la réplication multi-AZ.

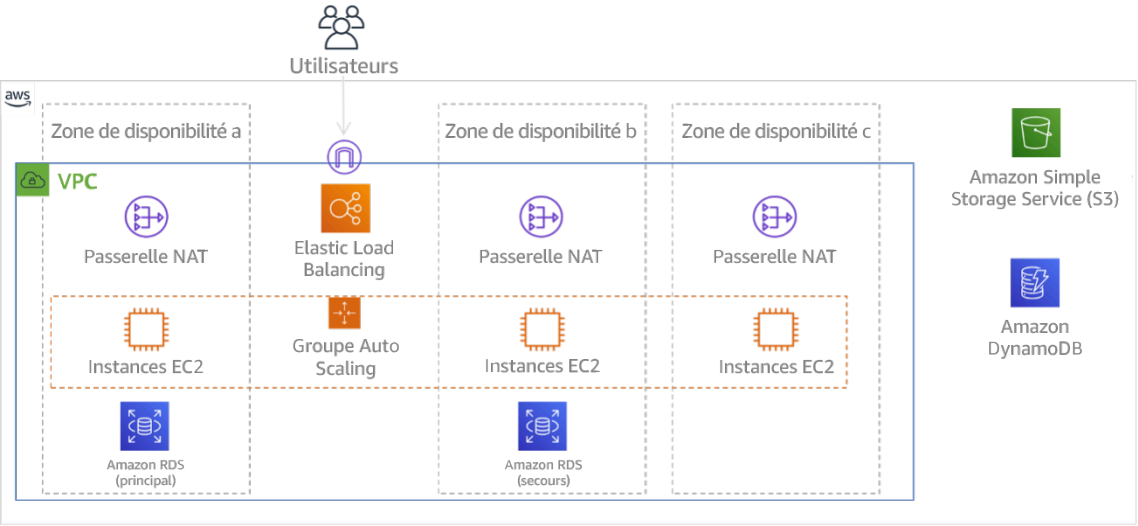


Figure 9 : Architecture multiniveau déployée sur trois zones de disponibilité. Notez qu'Amazon S3 et Amazon DynamoDB comportent toujours automatiquement plusieurs zones de disponibilités. L'ELB est également déployé dans les trois zones.

Utilisation de plusieurs Régions AWS

Si vos charges de travail nécessitent une résilience extrême (telles qu'une infrastructure critique, des applications liées à la santé ou des services répondant à des exigences strictes de disponibilité imposées par le client ou par le législateur), vous pouvez avoir besoin d'une disponibilité supérieure à ce qu'une Région AWS unique peut fournir. Dans ce cas, vous devez déployer et exploiter votre charge de travail sur au moins deux Régions AWS (en supposant que vos exigences en matière de résidence des données le permettent).

Les Régions AWS sont situées dans différentes régions géographiques du monde et sur plusieurs continents. Les Régions AWSprésentent une séparation physique et une isolation encore plus

importantes que les zones de disponibilité. À quelques exceptions près, les services AWS profitent de cette conception pour fonctionner de manière totalement indépendante entre les différentes régions (on parle alors de services régionaux). Un service Région AWSal est conçu de manière à ce qu'une défaillance n'ait pas d'impact sur le service dans une autre région.

Lorsque vous gérez votre charge de travail dans plusieurs régions, vous devez prendre en compte des exigences supplémentaires. Les ressources des différentes régions étant séparées et indépendantes les unes des autres, vous devez dupliquer les composants de votre charge de travail dans chaque région. Cela inclut l'infrastructure de base, telle que les VPC, en plus des services de calcul et de données.

REMARQUE : Lorsque vous envisagez une conception multirégionale, vérifiez que votre charge de travail est capable de s'exécuter dans une région unique. Si vous créez des dépendances entre des régions de manière à ce qu'un composant d'une région repose sur des services ou des composants d'une autre région, vous pouvez augmenter le risque de défaillance et affaiblir considérablement votre position en matière de fiabilité.

Pour faciliter les déploiements multirégionaux et maintenir la cohérence, AWS CloudFormation StackSets peut répliquer l'ensemble de votre infrastructure AWS entre plusieurs régions. AWS CloudFormation peut également détecter une dérive de configuration et vous informer lorsque vos ressources AWS d'une région ne sont pas synchronisées. De nombreux services AWS proposent la réplication multirégionale des ressources de charge de travail importantes. Par exemple, EC2 Image Builder peut publier vos images de machine EC2 (AMI) après chaque génération dans chaque région que vous utilisez. Amazon Elastic Container Registry (ECR) peut répliquer vos images de conteneur dans les régions que vous avez sélectionnées.

Vous devez également répliquer vos données dans chacune des régions que vous avez choisies. De nombreux services de données AWS gérés offrent une capacité de réplication interrégionale, notamment Amazon S3, Amazon DynamoDB, Amazon RDS, Amazon Aurora, Amazon Redshift, Amazon Elasticache, et Amazon EFS. Les tables globales Amazon DynamoDB acceptent les écritures dans toute région prise en charge et répliqueront les données entre toutes vos autres régions configurées. Pour les autres services, vous devez désigner une région principale pour les écritures, car les autres régions contiennent des réplicas en lecture seule. Consultez le guide de l'utilisateur et le manuel du développeur de chaque service de données AWS géré utilisé par votre charge de travail pour comprendre ses capacités et ses limites multirégionales. Accordez une attention particulière à l'endroit où les écritures doivent être dirigées, aux capacités et aux limites transactionnelles, à la manière dont la réplication est effectuée et à la manière de surveiller la synchronisation entre les régions.

AWS permet également d'acheminer de façon très flexible le trafic de demandes vers vos déploiements régionaux. Par exemple, vous pouvez configurer vos enregistrements DNS à l'aide d'Amazon Route 53 pour diriger le trafic vers la région disponible la plus proche de l'utilisateur. Vous pouvez également configurer vos enregistrements DNS dans une configuration active/en veille, dans laquelle vous désignez une région comme principale et ne vous rabattez sur un réplica régional que si la région principale devient défectueuse. Vous pouvez configurer la surveillance de l'état Route 53 pour détecter les points de terminaison défectueux et effectuer un basculement automatique. Vous pouvez également utiliser Amazon Application Recovery Controller (ARC) pour fournir un contrôle de routage hautement disponible permettant de réacheminer manuellement le trafic selon les besoins.

Même si vous choisissez de ne pas opérer dans plusieurs régions pour des raisons de haute disponibilité, considérez plusieurs régions dans le cadre de votre stratégie de reprise après sinistre (DR). Si possible, répliquez les composants et les données de l'infrastructure de votre charge de travail dans une configuration de secours à chaud ou d'environnement en veille dans une région secondaire. Dans cette conception, vous répliquez l'infrastructure de base de la région principale, telle que les VPC, les groupes Auto Scaling, les orchestrateurs de conteneurs et d'autres composants, mais vous configurez les composants de taille variable dans la région de secours (tels que le nombre d'instances EC2 et de réplicas de base de données) de manière à ce qu'ils aient une taille minimale exploitable. Vous organisez également une réplication continue des données de la région principale vers la région de secours. En cas d'incident, vous pouvez augmenter horizontalement ou accroître les ressources de la région de secours, puis la promouvoir en région principale.

Étapes d'implémentation

1. Travaillez avec les parties prenantes de l'entreprise et les experts en résidence des données pour déterminer les Régions AWS qui peuvent être utilisées pour héberger vos ressources et vos données.
2. Travaillez avec les parties prenantes techniques et commerciales pour évaluer votre charge de travail et déterminer si ses besoins de résilience peuvent être satisfaits par une approche multi-AZ (une seule Région AWS) ou s'ils nécessitent une approche multirégionale (si plusieurs régions sont autorisées). L'utilisation de plusieurs régions permet de bénéficier d'une plus grande disponibilité, mais peut entraîner une complexité et des coûts supplémentaires. Tenez compte des facteurs suivants dans votre évaluation :
    a. Objectifs commerciaux et exigences des clients : quelle est la durée d'indisponibilité autorisée en cas d'incident affectant la charge de travail dans une zone de disponibilité ou une région ?

Évaluez vos objectifs de point de récupération tels qu'ils sont présentés dans REL13-BP01 Définir les objectifs de reprise en termes de durée d'indisponibilité et de perte de données.

b. Exigences relatives à la reprise après sinistre (DR) : contre quel type de sinistre potentiel souhaitez-vous vous assurer ? Envisagez la possibilité d'une perte de données ou d'une indisponibilité à long terme à différents niveaux d'impact, d'une simple zone de disponibilité à une région entière. Si vous répliquez des données et des ressources entre des zones de disponibilité et qu'une seule zone de disponibilité connaît une défaillance prolongée, vous pouvez récupérer le service dans une autre zone de disponibilité. Si vous répliquez des données et des ressources entre plusieurs régions, vous pouvez récupérer le service dans une autre région.

3. Déployez vos ressources de calcul dans plusieurs zones de disponibilité.

a. Dans votre VPC, créez plusieurs sous-réseaux dans des zones de disponibilité différentes. Configurez chacune d'elles de manière à ce qu'elle soit suffisamment grande pour accueillir les ressources nécessaires pour répondre à la charge de travail, même en cas d'incident. Pour plus d'informations, consultez REL02-BP03 S'assurer que l'allocation des sous-réseaux IP tient compte de l'expansion et de la disponibilité.

b. Si vous utilisez des instances Amazon EC2, utilisez EC2 Auto Scaling pour gérer vos instances. Spécifiez les sous-réseaux que vous avez choisis à l'étape précédente lorsque vous créez vos groupes Auto Scaling.

c. Si vous utilisez le calcul AWS Fargate pour Amazon ECS ou Amazon EKS, sélectionnez les sous-réseaux que vous avez choisis à la première étape lors de la création d'un service ECS, lancez une tâche ECS ou créez un profil Fargate pour EKS.

d. Si vous utilisez des fonctions AWS Lambda qui doivent être exécutées dans votre VPC, sélectionnez les sous-réseaux que vous avez choisis à la première étape lors de la création de la fonction Lambda. Pour toutes les fonctions qui n'ont pas de configuration VPC, AWS Lambda gère automatiquement la disponibilité pour vous.

e. Placez des redirecteurs de trafic tels que des équilibreurs de charge devant vos ressources de calcul. Si l'équilibrage de charge entre zones est activé, les équilibreurs AWS Application Load Balancers et Network Load Balancers détectent quand des cibles telles que des instances et des conteneurs EC2 sont inaccessibles en raison d'une altération de la zone de disponibilité et redirigent le trafic vers des cibles situées dans des zones de disponibilité saines. Si vous désactivez l'équilibrage de charge entre zones, utilisez Amazon Application Recovery Controller (ARC) pour fournir une fonctionnalité de changement de zone. Si vous utilisez un équilibreur de charge tiers ou si vous avez implémenté vos propres équilibreurs de charge, configurez-les avec plusieurs front ends répartis dans différentes zones de disponibilité.

4. Répliquez les données de votre charge de travail sur plusieurs zones de disponibilité.
    a. Si vous utilisez un service de données AWS géré tel qu'Amazon RDS, Amazon ElastiCache ou Amazon FSx, étudiez son guide de l'utilisateur pour comprendre ses capacités de réplication de données et de résilience. Activez la réplication et le basculement entre zones de disponibilité si nécessaire.
    b. Si vous utilisez des services de stockage AWS gérés tels qu'Amazon S3, Amazon EFS et Amazon FSx, évitez d'utiliser des configurations mono-AZ ou à zone unique pour des données qui requièrent une durabilité élevée. Utilisez une configuration multi-AZ pour ces services. Consultez le guide de l'utilisateur du service correspondant pour déterminer si la réplication multi-AZ est activée par défaut ou si vous devez l'activer.
    c. Si vous exécutez une base de données, une file d'attente ou un autre service de stockage autogéré, organisez la réplication multi-AZ conformément aux instructions ou aux bonnes pratiques de l'application. Familiarisez-vous avec les procédures de basculement de votre application.
5. Configurez votre service DNS pour détecter une altération de la zone de disponibilité et rediriger le trafic vers une zone de disponibilité saine. Amazon Route 53, lorsqu'il est utilisé en combinaison avec des Elastic Load Balancers, peut le faire automatiquement. Route 53 peut également être configuré avec des enregistrements de basculement qui utilisent la surveillance de l'état pour répondre aux requêtes avec uniquement des adresses IP saines. Pour tous les enregistrements DNS utilisés pour le basculement, spécifiez une faible valeur de durée de vie (TTL) (par exemple, 60 secondes ou moins) afin d'éviter que la mise en cache des enregistrements n'entrave la reprise (les enregistrements d'alias Route 53 fournissent des durées de vie (TTL) appropriées pour vous).

Étapes supplémentaires lors de l'utilisation de plusieurs Régions AWS

1. Répliquez l'ensemble du code d'application et de système d'exploitation (OS) utilisé par votre charge de travail dans les régions que vous avez sélectionnées. Répliquez les images Amazon Machine Image (AMI) utilisées par vos instances EC2, si nécessaire, à l'aide de solutions telles qu'Amazon EC2 Image Builder. Répliquez les images de conteneur stockées dans des registres à l'aide de solutions telles que la réplication entre régions Amazon ECR. Activez la réplication régionale pour tous les compartiments Amazon S3 utilisés pour stocker les ressources d'application.
2. Déployez vos ressources de calcul et vos métadonnées de configuration (telles que les paramètres stockés dans AWS Systems Manager Parameter Store) dans plusieurs régions. Utilisez les mêmes procédures que celles décrites dans les étapes précédentes, mais répliquez

la configuration pour chaque région que vous utilisez pour votre charge de travail. Utilisez des solutions d'infrastructure en tant que code, telles qu'AWS CloudFormation pour reproduire uniformément les configurations entre les régions. Si vous utilisez une région secondaire dans une configuration d'environnement en veille pour la reprise après sinistre, vous pouvez réduire le nombre de vos ressources de calcul à une valeur minimale afin de réduire les coûts, avec une augmentation correspondante du temps de reprise.

3. Répliquez vos données de votre région principale vers vos régions secondaires.
   a. Les tables globales Amazon DynamoDB fournissent des réplicas globaux de vos données sur lesquels vous pouvez écrire depuis n'importe quelle région prise en charge. Avec d'autres services de données AWS gérés, tels qu'Amazon RDS, Amazon Aurora et Amazon Elasticache, vous désignez une région principale (lecture/écriture) et des régions de réplica (lecture seule). Consultez les guides de l'utilisateur et les manuels du développeur des services respectifs pour plus de détails sur la réplication régionale.
   b. Si vous exécutez une base de données autogérée, organisez la réplication multirégionale conformément aux instructions ou aux bonnes pratiques de l'application. Familiarisez-vous avec les procédures de basculement de votre application.
   c. Si votre charge de travail utilise AWS EventBridge, vous devrez peut-être transférer certains événements de votre région principale vers vos régions secondaires. Pour ce faire, spécifiez les bus d'événements dans vos régions secondaires comme cibles pour les événements correspondants dans votre région principale.
4. Déterminez si et dans quelle mesure vous souhaitez utiliser des clés de chiffrement identiques entre les régions. Une approche standard conciliant sécurité et facilité d'utilisation consiste à utiliser des clés régionales pour les données et l'authentification locales d'une région, et à utiliser des clés globales pour le chiffrement des données répliquées entre différentes régions. AWS Key Management Service (KMS) prend en charge les clés multirégionales pour répartir en toute sécurité et protéger les clés partagées entre les régions.
5. Envisagez d'utiliser AWS Global Accelerator pour améliorer la disponibilité de votre application en dirigeant le trafic vers les régions qui contiennent des points de terminaison sains.

Ressources

Bonnes pratiques associées :

- REL02-BP03 S'assurer que l'allocation des sous-réseaux IP tient compte de l'expansion et de la disponibilité

- REL11-BP05 Utiliser la stabilité statique pour éviter les comportements bimodaux
- REL13-BP01 Définir les objectifs de reprise en termes de durée d'indisponibilité et de perte de données

Documents connexes :

- Infrastructure mondiale AWS
- Livre blanc : Limites d'isolation des défaillances des services AWS
- Résilience dans Amazon EC2 Auto Scaling
- Amazon EC2 Auto Scaling : exemple : répartition des instances entre les zones de disponibilité
- Fonctionnement d'EC2 Image Builder
- Comment Amazon ECS place les tâches sur les instances de conteneur (y compris Fargate)
- Résilience dans AWS Lambda
- Amazon S3 : présentation de la réplication d'objets
- Réplication d'images privées sur Amazon ECR
- Tables globales : réplication multirégion avec DynamoDB
- Amazon Elasticache for Redis OSS : réplication entre Régions AWS à l'aide d'entrepôts de données globaux
- Résilience dans Amazon RDS
- Utilisation de bases de données globales Amazon Aurora
- Manuel du développeur AWS Global Accelerator
- Clés multi-régions dans AWS KMS
- Amazon Route 53 : configuration du basculement DNS
- Manuel du développeur Amazon Application Recovery Controller (ARC)
- Envoi et réception d'événements Amazon EventBridge entre régions Régions AWS
- Série de blog sur la création d'une application multirégion avec les services AWS
- Architecture de reprise après sinistre (DR) sur AWS, partie I : stratégies de reprise dans le cloud
- Architecture de reprise après sinistre sur AWS, partie III : Environnement en veille et secours à chaud

Vidéos connexes :

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications
- AWS re:Invent 2019: Innovation and operation of the AWS global network infrastructure

REL10-BP02 Automatiser la récupération des composants limités à un seul emplacement

Si les composants de la charge de travail ne peuvent s'exécuter que dans une seule zone de disponibilité ou un centre de données sur site, implémentez la capacité permettant d'effectuer une reconstruction complète de la charge de travail dans le cadre de vos objectifs de reprise définis.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : moyen

Directives d'implémentation

Si la bonne pratique de déploiement de la charge de travail sur plusieurs emplacements n'est pas possible en raison de contraintes technologiques, vous devez implémenter une autre solution de résilience. Vous devez automatiser la possibilité de recréer l'infrastructure nécessaire, de redéployer les applications et de recréer les données nécessaires pour ces situations.

Par exemple, Amazon EMR lance tous les nœuds d'un cluster donné dans la même zone de disponibilité, car l'exécution d'un cluster dans la même zone améliore les performances des flux de travail en fournissant un taux d'accès aux données plus élevé. Si ce composant est requis pour la résilience de la charge de travail, vous devez pouvoir redéployer le cluster et ses données. De même, pour Amazon EMR, vous devez assurer la redondance autrement qu'en utilisant plusieurs zones de disponibilité. Vous pouvez passer par plusieurs nœuds. Avec le système de fichiers EMR (EMRFS), les données EMR peuvent être conservées dans Amazon S3, et ainsi être répliquées sur plusieurs zones de disponibilité ou Régions AWS.

De même, pour Amazon Redshift, il met en service, par défaut, votre cluster dans une zone de disponibilité sélectionnée de façon aléatoire au sein de la Région AWS que vous sélectionnez. Tous les nœuds de cluster sont provisionnés dans la même zone.

Pour les charges de travail basées sur des serveurs avec état déployés dans un centre de données sur site, vous pouvez utiliser Reprise après sinistre AWS Elastic pour protéger vos charges de travail dans AWS. Si vous êtes déjà hébergé dans AWS, Elastic Disaster Recovery peut vous permettre de protéger votre charge de travail dans une autre zone de disponibilité ou région. Elastic Disaster Recovery utilise une réplication continue au niveau des blocs vers une zone de stockage légère afin de fournir une récupération rapide et fiable des applications sur site et dans le cloud.

Étapes d'implémentation

1. Implémentation de l'autorégénération Dans la mesure du possible, déployez vos instances ou vos conteneurs en utilisant la mise à l'échelle automatique. Si vous ne pouvez pas utiliser la mise à l'échelle automatique, utilisez la récupération automatique pour les instances EC2 ou mettez en place un mécanisme d'autoréparation basé sur Amazon EC2 ou des événements de cycle de vie de conteneur ECS.
   - Utilisez les groupes Amazon EC2 Auto Scaling pour les instances et les charges de travail de conteneur qui n'ont aucune exigence en matière d'adresse IP d'instance, d'adresse IP privée, d'adresse IP élastique et de métadonnées d'instance.
     - Les données utilisateur du modèle de lancement peuvent être utilisées pour mettre en place un mécanisme permettant la récupération automatique de la plupart des charges de travail.
   - Utilisez la récupération automatique des instances Amazon EC2 pour les charges de travail nécessitant une seule adresse d'ID d'instance, une adresse IP privée, une adresse IP élastique et les métadonnées d'instance.
     - La récupération automatique envoie des alertes de statut de récupération à une rubrique SNS lorsque la défaillance de l'instance est détectée.
   - Utilisez les événements du cycle de vie de l'instance Amazon EC2 ou les événements Amazon ECS pour automatiser l'autoréparation lorsque la mise à l'échelle automatique ou la récupération de votre instance EC2 ne peuvent pas être utilisées.
     - Utilisez les événements pour invoquer le mécanisme vous permettant de réparer votre composant selon la logique de processus dont vous avez besoin.
   - Protégez les charges de travail avec état limitées à un seul emplacement à l'aide de Reprise après sinistre AWS Elastic.

Ressources

Documents connexes :

- Événements Amazon ECS
- Hooks de cycle de vie Amazon EC2 Auto Scaling
- Récupération de votre instance.
- Mise à l'échelle automatique des services
- Qu'est-ce qu'Amazon EC2 Auto Scaling ?
- Reprise après sinistre AWS Elastic

REL10-BP03 Utiliser des architectures cloisonnées pour limiter la portée de l'impact

Mettez en œuvre des architectures de cloisonnement (également connues sous le nom d'architectures cellulaires) pour restreindre l'effet d'une panne au sein d'une charge de travail à un nombre limité de composants.

Résultat escompté : une architecture cellulaire utilise plusieurs instances isolées d'une charge de travail, chaque instance étant appelée cellule. Chaque cellule est indépendante, ne partage pas d'état avec les autres cellules et traite un sous-ensemble des demandes de la charge de travail globale. L'impact potentiel d'une défaillance, telle qu'une mauvaise mise à jour logicielle, sur une cellule individuelle et sur les demandes qu'elle traite est ainsi réduit. Si une charge de travail utilise 10 cellules pour traiter 100 demandes, lorsqu'une panne survient, 90 % des demandes globales ne sont pas affectées par la panne.

Anti-modèles courants :

- Permettre aux cellules de se développer sans limites.
- Appliquer des mises à jour ou des déploiements de code à toutes les cellules en même temps.
- Partage de l'état ou des composants entre les cellules (à l'exception de la couche routeur).
- Ajout d'une logique métier ou de routage complexe à la couche routeur.
- Ne pas minimiser les interactions entre les cellules.

Avantages liés au respect de cette bonne pratique : avec les architectures cellulaires, de nombreux types de défaillances courants sont contenus dans la cellule elle-même, ce qui permet une isolation supplémentaire des pannes. Ces limites de défaillances peuvent apporter de la résilience face à des types de défaillances difficiles à contenir, tels que des déploiements de code infructueux ou des demandes corrompues ou invoquant un mode de défaillance spécifique (également appelées demandes de pilules empoisonnées).

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : élevé

Directives d'implémentation

Sur un navire, les cloisons permettent de contenir une brèche dans la coque dans une seule section de la coque. Dans les systèmes complexes, ce modèle est souvent répliqué pour permettre d'isoler

des pannes. Les limites isolées pour les défaillances restreignent l'effet d'une panne au sein d'une charge de travail à un nombre limité de composants. Les composants situés en dehors du périmètre ne sont pas affectés par la défaillance. En utilisant plusieurs périmètres d'isolation des pannes, vous pouvez limiter l'impact sur votre charge de travail. Sur AWS, les clients peuvent utiliser plusieurs zones de disponibilité et régions pour isoler des pannes, mais le concept d'isolement des pannes peut également être étendu à l'architecture de votre charge de travail.

La charge de travail globale est divisée en cellules par une clé de partition. Celle-ci doit s'aligner sur la base de granularité du service, ou sur la manière naturelle dont la charge de travail d'un service peut être subdivisée avec un minimum d'interactions entre les cellules. Des exemples de clés de partition sont l'ID du client, l'ID de la ressource ou tout autre paramètre facilement accessible dans la plupart des appels d'API. Une couche de routage des cellules distribue les requêtes aux cellules individuelles en fonction de la clé de partition et présente un point de terminaison unique aux clients.

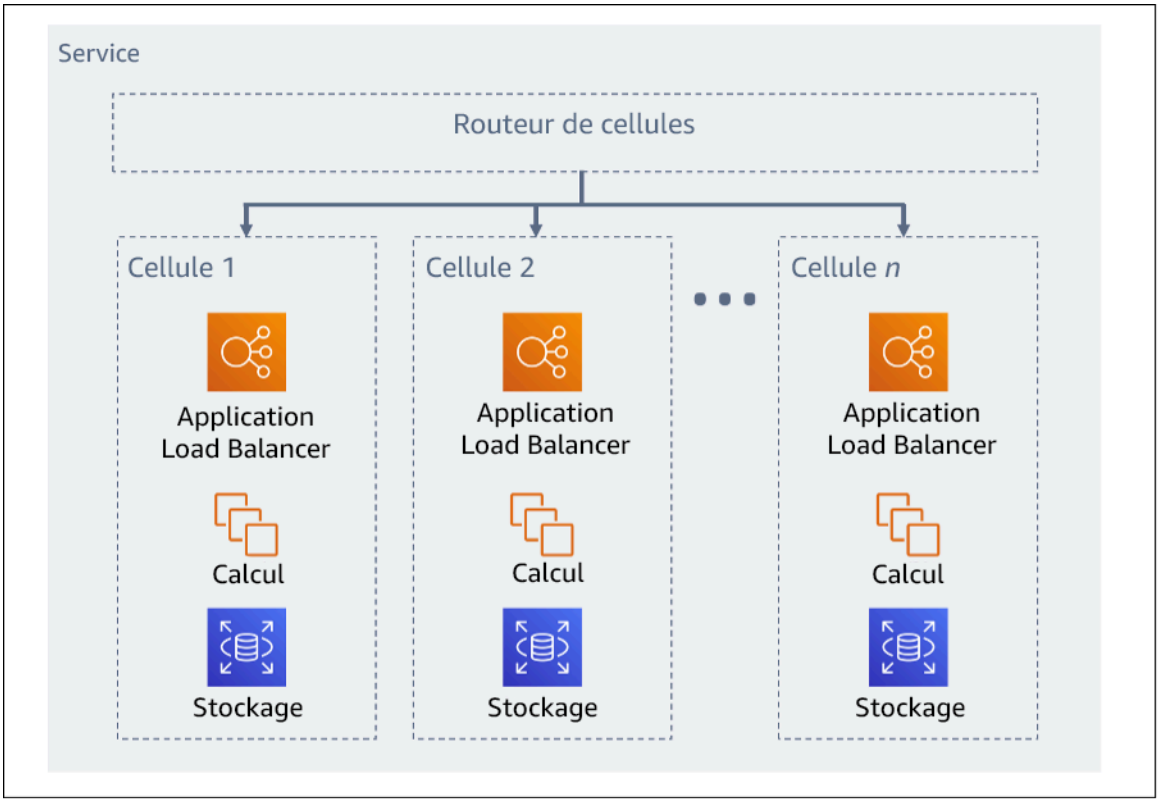


Figure 11 : architecture cellulaire

Étapes d'implémentation

Lors de la conception d'une architecture cellulaire, vous devez tenir compte de plusieurs éléments :

1. Clé de partition : une attention particulière doit être prise lors du choix de la clé de partition.

- Celle-ci doit s'aligner sur la base de granularité du service, ou sur la manière naturelle dont la charge de travail d'un service peut être subdivisée avec un minimum d'interactions entre les cellules. Exemples : `customer ID` ou `resource ID`.
- La clé de partition doit être disponible dans toutes les requêtes, soit directement, soit d'une manière qui pourrait être facilement déduite de façon déterministe par d'autres paramètres.

2. Mappage cellulaire persistant : les services en amont ne doivent interagir qu'avec une seule cellule pendant le cycle de vie de leurs ressources.
   - En fonction de la charge de travail, vous devrez peut-être concevoir une stratégie de migration de cellules pour faire migrer les données d'une cellule à l'autre. La migration d'une cellule peut s'avérer nécessaire si un utilisateur ou une ressource particulière de votre charge de travail devient trop importante et nécessite une cellule dédiée.
   - Les cellules ne doivent pas partager d'état ou de composants entre elles.
   - Par conséquent, les interactions entre cellules doivent être évitées ou réduites au minimum, car elles créent des dépendances entre les cellules et diminuent donc les bénéfices de l'isolement des défaillances.
3. Couche routeur : la couche routeur est un composant partagé entre les cellules et ne peut donc pas suivre la même stratégie de compartimentage qu'avec les cellules.
   - Nous recommandons de paramétrer la couche routeur pour distribuer les requêtes à des cellules individuelles à l'aide d'un algorithme de mappage de partition d'une manière efficace sur le plan des calculs. Par exemple, en combinant des fonctions de hachage cryptographiques et de l'arithmétique modulaire pour mapper les clés de partition aux cellules.
   - Pour éviter les impacts sur plusieurs cellules, la couche de routage doit rester le plus simple possible et être doté d'une capacité de mise à l'échelle horizontale optimale, ce qui nécessite d'éviter toute logique métier complexe au sein de cette couche. L'avantage supplémentaire est qu'il est facile de comprendre le comportement attendu à tout moment, ce qui permet de réaliser des tests approfondis. Comme l'explique Colm MacCárthaigh dans Fiabilité, travail constant et une bonne tasse de café, des conceptions simples et des modèles de travail constants produisent des systèmes fiables et réduisent la fragilité.
4. Taille des cellules : les cellules doivent avoir une taille maximale et ne doivent pas être autorisées à croître au-delà de cette taille.
   - La taille maximale doit être identifiée en effectuant des tests approfondis, jusqu'à ce que les points de rupture soient atteints et que des marges de fonctionnement sûres soient établies. Pour en savoir plus sur la mise en œuvre des pratiques de test, consultez REL07-BP04 Testez votre charge de travail

- La charge de travail globale doit se développer en ajoutant des cellules supplémentaires, ce qui lui permet de s'adapter à l'augmentation de la demande.

5. Stratégies multi-AZ ou multi-régions : plusieurs niveaux de résilience doivent être exploités pour se protéger contre différents domaines de défaillance.
   - Pour la résilience, vous devez adopter une approche qui repose sur des couches de défense. Une couche protège contre les perturbations de petite envergure et courantes en créant une architecture hautement disponible à l'aide de plusieurs AZ. Une autre couche de défense est destinée à protéger contre les événements rares tels que les catastrophes naturelles généralisées et les perturbations au niveau régional. Cette deuxième couche implique de concevoir l'architecture de votre application pour qu'elle s'étende sur plusieurs Régions AWS. La mise en œuvre d'une stratégie multi-région pour votre charge de travail permet de la protéger contre les catastrophes naturelles généralisées qui affectent une grande région géographique d'un pays, ou les défaillances techniques à l'échelle régionale. Sachez que la mise en œuvre d'une architecture multi-région peut être très complexe et n'est généralement pas requise pour la plupart des charges de travail. Pour en savoir plus, veuillez consulter REL10-BP01 Déploiement de la charge de travail sur plusieurs emplacements.
6. Déploiement du code : une stratégie de déploiement de code échelonnée doit être préférée au déploiement de modifications de code dans toutes les cellules en même temps.
   - Les risques de panne de plusieurs cellules sont ainsi réduits en raison d'un mauvais déploiement ou d'une erreur humaine. Pour en savoir plus, consulter Automatiser un déploiement sûr et sans intervention.

Ressources

Bonnes pratiques associées:

- REL07-BP04 Testez votre charge de travail
- REL10-BP01 Déploiement de la charge de travail sur plusieurs emplacements

Documents connexes :

- Fiabilité, travail constant et une bonne tasse de café
- AWS et compartimentage
- Isolation de la charge de travail à l'aide du partitionnement aléatoire
- Automatiser un déploiement sûr et sans intervention

Vidéos connexes :

- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small
- AWS re:Invent 2018: How AWS Minimizes the Blast Radius of Failures (ARC338)
- Shuffle-sharding: AWS re:Invent 2019: Introducing The Amazon Builders' Library (DOP328)
- AWS Summit ANZ 2021 - Everything fails, all the time: Designing for resilience

Conception d'une charge de travail qui résiste aux défaillances des composants

Les charges de travail exigeant une haute disponibilité et un faible temps moyen de récupération (MTTR) doivent être conçues pour être résilientes.

Bonnes pratiques

- REL11-BP01 Surveiller tous les composants de la charge de travail pour détecter les défaillances
- REL11-BP02 Basculer vers des ressources saines
- REL11-BP03 Automatiser la réparation sur toutes les couches
- REL11-BP04 S'appuyer sur le plan de données et non sur le plan de contrôle pendant la récupération
- REL11-BP05 Utiliser la stabilité statique pour éviter les comportements bimodaux
- REL11-BP06 Envoyer des notifications lorsque des événements affectent la disponibilité
- REL11-BP07 Concevoir votre produit pour atteindre les objectifs de disponibilité et les contrats de niveau de service (SLA)

REL11-BP01 Surveiller tous les composants de la charge de travail pour détecter les défaillances

Surveillez en continu l'état de votre charge de travail afin que vous et vos systèmes automatisés ayez connaissance des dégradations ou des défaillances dès qu'elles se produisent. Surveillez les indicateurs clés de performance (KPI) en fonction de la valeur commerciale.

Tous les mécanismes de récupération et de réparation doivent commencer par la capacité à détecter rapidement les problèmes. Les défaillances techniques doivent être détectées au préalable pour être

résolues. Cependant, la disponibilité repose sur la capacité de votre charge de travail à fournir une valeur commerciale. Il doit donc s'agir d'indicateurs clés de performance (KPI) de votre stratégie de détection et de correction.

Résultat escompté : les composants essentiels d'une charge de travail sont surveillés de manière indépendante afin de détecter les défaillances et de les signaler au moment et à l'emplacement où elles se produisent.

Anti-modèles courants :

- Aucune alarme n'a été configurée. Les pannes se produisent donc sans notification.
- Des alarmes existent, mais les seuils ne laissent pas assez de temps pour réagir.
- Les métriques ne sont pas collectées à une fréquence suffisante pour atteindre l'objectif de délai de reprise (RTO).
- Seules les interfaces de la charge de travail axées directement sur le client sont activement surveillées.
- Collecte uniquement des métriques techniques et non des métriques de fonction commerciale.
- Aucune métrique ne mesure l'expérience utilisateur de la charge de travail.
- Trop de contrôleurs sont créés.

Avantages liés au respect de cette bonne pratique : la surveillance appropriée à tous les niveaux vous permet de raccourcir le délai de reprise en réduisant le temps de détection.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : élévé

Directives d'implémentation

Identifiez toutes les charges de travail qui seront examinées à des fins de surveillance. Une fois que vous avez identifié tous les composants de la charge de travail à surveiller, déterminez l'intervalle de surveillance. Cet intervalle a un impact direct sur la rapidité avec laquelle la restauration peut être initiée en fonction du temps nécessaire pour détecter une panne. Le délai moyen de détection (MTTD) est le délai entre le moment où une panne survient et le moment où les opérations de réparation commencent. La liste des services doit être longue et complète.

La surveillance doit couvrir toutes les couches de la pile d'applications, y compris l'application, la plateforme, l'infrastructure et le réseau.

Votre stratégie de surveillance doit tenir compte de l'impact des défaillances grises. Pour en savoir plus sur les défaillances grises, consultez la section Défaillances grises dans le livre blanc Modèles de résilience Multi-AZ avancée.

Étapes d'implémentation

- Votre intervalle de surveillance dépend de la vitesse à laquelle vous devez effectuer la récupération. Votre délai de reprise dépend du temps nécessaire à la récupération. Vous devez donc déterminer la fréquence de collecte en tenant compte de cette durée et de votre objectif de délai de reprise (RTO).
- Configurez la surveillance détaillée des composants et des services gérés.
  - Déterminez la nécessité d'une surveillance détaillée pour les instances EC2 et Auto Scaling. La surveillance détaillée fournit des métriques à intervalle d'une minute, et la surveillance par défaut fournit des métriques à intervalle de cinq minutes.
  - Déterminez la nécessité de la surveillance améliorée pour RDS. La surveillance améliorée utilise un agent sur les instances RDS pour obtenir des informations utiles sur différents processus ou fils.
  - Déterminez les exigences de surveillance des composants sans serveur critiques pour Lambda, API Gateway, Amazon EKS, Amazon ECS et tous les types d'équilibreurs de charge.
  - Déterminez les exigences de surveillance des composants de stockage pour Amazon S3, Amazon FSx, Amazon EFS et Amazon EBS.
- Créez des métriques personnalisées pour mesurer les indicateurs clés de performance (KPI) métier. Les charges de travail mettent en œuvre des fonctions commerciales stratégiques, qui doivent être utilisées comme indicateurs clés de performance permettant d'identifier les problèmes indirects.
- Surveillez l'expérience utilisateur pour détecter les défaillances à l'aide de tests canary utilisateur. Les tests de transaction synthétiques (également appelés « tests canary », à ne pas confondre avec les déploiements canary) qui peuvent exécuter et simuler le comportement des clients font partie des processus de test les plus importants. Exécutez ces tests en permanence sur vos points de terminaison de charge de travail à partir de divers emplacements distants.
- Créez des métriques personnalisées qui suivent l'expérience utilisateur. Si vous pouvez analyser l'expérience du client, vous pouvez savoir à quel moment l'expérience du consommateur se dégrade.
- Définissez des alarmes pour détecter quand une partie de votre charge de travail ne fonctionne pas correctement et pour indiquer quand mettre à l'échelle automatiquement les ressources. Le

système peut afficher les alarmes sur des tableaux de bord, envoyer des alertes via Amazon SNS ou par e-mail et fonctionner avec Auto Scaling pour une mise à l'échelle à la hausse ou à la baisse des ressources de la charge de travail.

- Créez des tableaux de bord pour la visualisation de vos métriques. Les tableaux de bord peuvent être utilisés pour afficher visuellement des tendances, des valeurs aberrantes et d'autres indicateurs de problèmes potentiels ou pour fournir une indication des problèmes que vous pourriez vouloir examiner.
- Créez un système de suivi distribué pour vos services. La surveillance distribuée vous permet d'analyser les performances de votre application et de ses services sous-jacents, afin d'identifier et de dépanner la cause première des problèmes et des erreurs de performances.
- Créez des systèmes de surveillance (à l'aide de CloudWatch ou X-Ray), des tableaux de bord et collectez des données dans une région et un compte distincts.
- Restez informé des dégradations de service avec AWS Health. Créez des notifications d'événements AWS Health spécialement adaptées aux e-mails et aux canaux de discussion via Notifications des utilisateurs AWS et intégrez-les de manière programmatique à vos outils de surveillance et d'alerte via Amazon EventBridge.

Ressources

Bonnes pratiques associées:

- Définition de la disponibilité
- REL11-BP06 Envoi de notifications lorsque des événements affectent la disponibilité

Documents connexes:

- Amazon CloudWatch Synthetics vous permet de créer des tests canary utilisateur
- Activer ou désactiver la surveillance détaillée pour votre instance
- Surveillance améliorée
- Surveillance de vos groupes et instances Auto Scaling à l'aide d'Amazon CloudWatch
- Publication des métriques personnalisées
- Utilisation d'alarmes Amazon CloudWatch
- Utilisation des tableaux de bord CloudWatch
- Utilisation de tableaux de bord CloudWatch interrégionaux entre comptes

- Utilisation du suivi X-Ray interrégional entre comptes
- Compréhension de la disponibilité

Vidéos connexes:

- Mitigating gray failures

Exemples connexes :

- Un atelier sur l'observabilité : explorer X-Ray

Outils associés:

- CloudWatch
- CloudWatch X-Ray

REL11-BP02 Basculer vers des ressources saines

En cas de défaillance des ressources, les ressources saines doivent continuer à répondre aux requêtes. Pour les altérations liées à l'emplacement (par exemple, zone de disponibilité ou Région AWS), vérifiez que des systèmes sont en place pour basculer vers des ressources saines dans des emplacements intacts.

Lorsque vous concevez un service, répartissez la charge entre les ressources, les zones de disponibilité ou les régions. La défaillance d'une ressource individuelle ou l'altération peut ainsi être atténuée en déplaçant le trafic vers les ressources saines restantes. Réfléchissez à la manière dont les services sont découverts et acheminés en cas de défaillance.

Concevez vos services en tenant compte de la restauration après panne. Chez AWS, nous concevons les services afin de réduire le temps de restauration au minimum en cas de défaillance, ainsi que l'impact sur les données. Nos services utilisent principalement des magasins de données qui valident les requêtes uniquement lorsque les données sont stockées durablement sur plusieurs réplicas au sein d'une région. Ils sont élaborés de manière à utiliser l'isolation basée sur les cellules et à faire appel à l'isolement des pannes fourni par des zones de disponibilité. Nous utilisons largement l'automatisation dans nos procédures opérationnelles. Nous optimisons également notre fonction de remplacement et redémarrage afin de récupérer rapidement en cas d'interruptions.

Les modèles et les conceptions qui permettent le basculement varient pour chaque service de plateforme AWS. De nombreux services natifs gérés par AWS sont dotés de plusieurs zones de disponibilité (comme Lambda ou API Gateway). D'autres services AWS (comme EC2 et EKS) nécessitent des conceptions répondant à des bonnes pratiques spécifiques pour prendre en charge le basculement des ressources ou le stockage des données entre les zones de disponibilité.

La surveillance doit être configurée pour vérifier que la ressource de basculement est saine, suivre la progression du basculement des ressources et surveiller le rétablissement des processus métier.

Résultat escompté : les systèmes sont capables d'utiliser automatiquement ou manuellement de nouvelles ressources pour se remettre d'une dégradation.

Anti-modèles courants :

- La planification des défaillances ne fait pas partie de la phase de planification et de conception.
- Le RTO et le RPO ne sont pas établis.
- Surveillance insuffisante pour détecter les ressources défaillantes.
- Isolement approprié des domaines de défaillance.
- Le basculement multi-régional n'est pas pris en compte.
- La détection des défaillances est trop sensible ou trop agressive lors de la décision de basculer.
- Pas de test ni de validation de la conception du basculement.
- Automatisation de la réparation automatique sans notification indiquant que la réparation était nécessaire.
- Pas de période d'attente pour éviter un failback trop précoce.

Avantages liés au respect de cette bonne pratique : vous pouvez créer des systèmes plus résilients qui préservent leur fiabilité en cas de défaillance en se dégradant progressivement et en se rétablissant rapidement.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : élevé

Directives d'implémentation

Les services AWS tels que Elastic Load Balancing et Amazon EC2 Auto Scaling aident à répartir la charge entre les ressources et les zones de disponibilité. Par conséquent, la défaillance d'une ressource individuelle (telle qu'une instance EC2) ou l'altération d'une zone de disponibilité peut être atténuée en déplaçant le trafic vers les ressources saines restantes.

Pour les charges de travail multi-régionales, les conceptions sont plus complexes. Par exemple, les réplicas en lecture entre régions vous permettent de déployer vos données sur plusieurs Régions AWS. Cependant, le basculement est toujours nécessaire pour faire passer le réplica en lecture au niveau principal, puis pour rediriger le trafic vers le nouveau point de terminaison. Amazon Route 53, Amazon Application Recovery Controller (ARC), Amazon CloudFront et AWS Global Accelerator peuvent aider à router le trafic entre les Régions AWS.

Les services AWS, tels qu'Amazon S3, Lambda, API Gateway, Amazon SQS, Amazon SNS, Amazon SES, Amazon Pinpoint, Amazon ECR, AWS Certificate Manager, EventBridge ou Amazon DynamoDB, sont automatiquement déployés vers plusieurs zones de disponibilité par AWS. En cas de défaillance, ces services AWS acheminent automatiquement le trafic vers des emplacements sains. Les données sont stockées de manière redondante dans plusieurs zones de disponibilité et restent disponibles.

Pour Amazon RDS, Amazon Aurora, Amazon Redshift, Amazon EKS ou Amazon ECS, avoir plusieurs zones de disponibilité est une option de configuration. AWS peut diriger le trafic vers l'instance saine si le basculement est initié. Cette action de basculement peut être prise par AWS ou conformément aux exigences du client.

Pour les instances Amazon EC2, Amazon Redshift, les tâches Amazon ECS ou les pods Amazon EKS, vous choisissez les zones de disponibilité dans lesquelles effectuer le déploiement. Pour certaines conceptions, Elastic Load Balancing fournit la solution pour détecter les instances dans les zones défectueuses et acheminer le trafic vers les zones saines. Elastic Load Balancing peut même acheminer le trafic vers les composants de votre centre de données sur site.

Pour le basculement du trafic multi-régional, le réacheminement peut tirer parti d'Amazon Route 53, d'Amazon Application Recovery Controller, d'AWS Global Accelerator, de Route 53 Private DNS pour VPC ou de CloudFront pour fournir un moyen de définir des domaines Internet et d'attribuer des politiques de routage, y compris des surveillances de l'état, afin de router le trafic vers des régions saines. AWS Global Accelerator fournit des adresses IP statiques qui agissent comme point d'entrée fixe vers votre application, puis routent le trafic vers les points de terminaison des Régions AWS de votre choix, en utilisant le réseau mondial AWS plutôt qu'Internet pour de meilleures performances et une meilleure fiabilité.

Étapes d'implémentation

- Créez des modèles de basculement pour toutes les applications et tous les services appropriés. Isolez chaque composant de l'architecture et créez des conceptions de basculement respectant le RTO et le RPO pour chacun d'eux.

- Configurez les environnements de bas niveau (tels que le développement ou les tests) avec tous les services requis pour disposer d'un plan de basculement. Déployez les solutions en utilisant l'infrastructure en tant que code (IaC) pour garantir la reproductibilité.
- Configurez un site de reprise tel qu'une deuxième région pour implémenter et tester les modèles de basculement. Si nécessaire, les ressources pour les tests peuvent être configurées temporairement afin de limiter les coûts supplémentaires.
- Déterminez quels plans de basculement seront automatisés par AWS, ceux qui peuvent être automatisés par un processus DevOps et ceux qui peuvent être manuels. Documentez et mesurez le RTO et le RPO de chaque service.
- Créez un manuel de basculement et incluez toutes les étapes nécessaires au basculement de chaque ressource, application et service.
- Créez un manuel de failback et incluez toutes les étapes nécessaires (avec calendrier) pour chaque ressource, application et service.
- Créez un plan pour lancer et répéter le manuel. Utilisez des simulations et des tests de chaos pour tester les étapes du manuel et l'automatisation.
- Pour toute altération liée à l'emplacement (par exemple, zone de disponibilité ou Région AWS), vérifiez que des systèmes sont en place pour basculer vers des ressources saines dans des emplacements intacts. Vérifiez le quota, les niveaux de mise à l'échelle automatique et les ressources en cours d'exécution avant le test de basculement.

Ressources

Bonnes pratiques Well-Architected connexes:

- REL13 – Plan de reprise après sinistre
- REL10 – Utilisation de l'isolation des défaillances pour protéger votre charge de travail

Documents connexes:

- Définition des objectifs RTO et RPO
- Basculement à l'aide du routage pondéré Route 53
- Disaster Recovery with Amazon Application Recovery Controller
- EC2 avec mise à l'échelle automatique
- Déploiements EC2 – Multi-AZ

- Déploiements ECS – Multi-AZ
- Switch traffic using Amazon Application Recovery Controller
- Lambda avec Application Load Balancer et basculement
- Réplication et basculement ACM
- Réplication et basculement du magasin de paramètres
- Réplication entre régions ECR et basculement
- Configuration de la réplication entre régions du gestionnaire de secrets
- Activation de la réplication entre régions pour EFS et basculement
- Réplication entre régions EFS et basculement
- Basculement du réseau
- Basculement des points de terminaison S3 à l'aide du protocole MRAP
- Création d'une réplication entre régions pour S3
- Guidance for Cross Region Failover and Graceful Failback on AWS
- Basculement à l'aide d'un accélérateur mondial multi-régional
- Basculement avec DRS

Exemples connexes :

- Reprise après sinistre sur AWS
- Elastic Disaster Recovery sur AWS

REL11-BP03 Automatiser la réparation sur toutes les couches

Utilisez des capacités automatisées pour effectuer des actions correctives en cas de détection d'une défaillance. Les dégradations peuvent être automatiquement corrigées par le biais de mécanismes de service internes ou peuvent nécessiter le redémarrage ou la suppression des ressources par le biais d'actions correctives.

Pour les applications autogérées et la réparation interrégionale, les modèles de restauration et les processus de réparation automatisés peuvent être extraits des bonnes pratiques existantes.

Pouvoir redémarrer ou supprimer une ressource est important pour remédier aux défaillances. Une bonne pratique consiste à rendre les services sans état dans la mesure du possible. Cela évite toute perte de données ou de disponibilité au redémarrage des ressources. Dans le cloud, vous pouvez

(et devriez généralement) remplacer la totalité de la ressource (par exemple, une instance de calcul ou une fonction sans serveur) dans le cadre du redémarrage. Le redémarrage proprement dit est un moyen simple et fiable de récupération après une défaillance. De nombreux types de défaillances différents se produisent dans les charges de travail. Les défaillances peuvent se produire au niveau du matériel, des logiciels, des communications et des opérations.

Le redémarrage ou la nouvelle tentative s'appliquent également aux requêtes réseau. Appliquez la même approche de récupération à la fois pour un délai d'expiration réseau et une défaillance de la dépendance, si la dépendance renvoie une erreur. Comme ces deux événements ont un effet semblable sur le système, plutôt que d'essayer de traiter l'un ou l'autre comme un « cas particulier », appliquez une stratégie semblable de nouvelle tentative limitée avec un backoff exponentiel et une instabilité. La possibilité d'exécuter un redémarrage est un mécanisme de récupération présenté dans l'informatique orientée récupération et dans les architectures de cluster haute disponibilité.

Résultat escompté : des actions automatisées sont effectuées pour remédier à la détection d'une panne.

Anti-modèles courants :

- Allocation des ressources sans mise à l'échelle automatique.
- Déploiement d'applications une par une dans des instances ou des conteneurs.
- Déploiement d'applications qui ne peuvent pas être déployées dans plusieurs emplacements sans utiliser la récupération automatique.
- Réparation manuelle des applications impossible à réparer par la mise à l'échelle et la récupération automatiques.
- Aucune automatisation pour le basculement des bases de données.
- Absence de méthodes automatisées pour rediriger le trafic vers de nouveaux points de terminaison.
- Aucune réplication du stockage.

Avantages liés au respect de cette bonne pratique : la réparation automatisée contribue à réduire le temps moyen de récupération et à améliorer votre disponibilité.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : élevé

Directives d'implémentation

Les conceptions pour Amazon EKS ou les autres services Kubernetes doivent inclure à la fois un minimum et un maximum de réplicas ou d'ensembles dynamiques, ainsi que le dimensionnement minimal des clusters et des groupes de nœuds. Ces mécanismes mettent à disposition un minimum de ressources de traitement en permanence tout en corrigeant automatiquement les défaillances à l'aide du plan de contrôle Kubernetes.

Les modèles de conception accessibles via un équilibreur de charge utilisant des clusters de calcul doivent tirer parti des groupes Auto Scaling. Elastic Load Balancing (ELB) distribue automatiquement le trafic applicatif entrant sur plusieurs cibles et appareils virtuels dans une ou plusieurs zones de disponibilité (AZ).

La taille des conceptions basées sur le calcul en cluster qui n'utilisent pas l'équilibrage de charge doit être conçue pour la perte d'au moins un nœud. Cela permet au service de continuer à fonctionner avec une capacité potentiellement réduite pendant la restauration d'un nouveau nœud. Mongo, DynamoDB Accelerator, Amazon Redshift, Amazon EMR, Cassandra, Kafka, MSK-EC2, Couchbase, ELK et Amazon OpenSearch Service sont des exemples de services. Bon nombre de ces services peuvent être conçus avec des fonctionnalités supplémentaires de réparation automatique. Certaines technologies de cluster doivent générer une alerte en cas de perte d'un nœud, déclenchant un flux de travail automatique ou manuel pour recréer un nœud. Ce flux de travail peut être automatisé avec AWS Systems Manager pour résoudre rapidement les problèmes.

Amazon EventBridge peut être utilisé pour surveiller et filtrer les événements tels que les alarmes CloudWatch ou les changements d'état d'autres services AWS. En fonction des informations d'événement, il peut ensuite invoquer AWS Lambda, Systems Manager Automation ou d'autres cibles pour exécuter une logique de correction personnalisée sur votre charge de travail. Amazon EC2 Auto Scaling peut être configuré pour vérifier l'état de l'instance EC2. Si l'instance est dans un état autre que celui en cours d'exécution, ou si le statut du système est dégradé, Amazon EC2 Auto Scaling considère l'instance comme défectueuse et lance une instance de remplacement. Pour les remplacements à grande échelle (comme la perte d'une zone de disponibilité complète), il est préférable d'opter pour la stabilité statique pour une haute disponibilité.

Étapes d'implémentation

- Utilisez des groupes Auto Scaling pour déployer des niveaux dans une charge de travail. L'autoscaling peut effectuer une autoréparation sur les applications sans état et ajouter ou supprimer de la capacité.

- Pour les instances de calcul mentionnées précédemment, utilisez l'équilibrage de charge et choisissez le type d'équilibreur de charge approprié.
- Envisagez de réparer Amazon RDS. Avec les instances de secours, configurez le basculement automatique vers l'instance de secours. Pour les réplicas en lecture Amazon RDS, un flux de travail automatisé est nécessaire pour rendre un réplica en lecture principal.
- Implémentez la récupération automatique sur les instances EC2 dont les applications déployées ne peuvent pas être déployées dans plusieurs emplacements et qui peuvent tolérer le redémarrage en cas de défaillance. La récupération automatique peut être utilisée pour remplacer du matériel défaillant et redémarrer l'instance lorsque l'application ne peut pas être déployée sur plusieurs emplacements. Les métadonnées de l'instance et les adresses IP associées sont conservées, tout comme le sont les volumes EBS et les points de montage sur Amazon Elastic File System ou sur File Systems for Lustre et Windows. À l'aide de AWS OpsWorks, vous pouvez configurer la réparation automatique des instances EC2 au niveau de la couche.
- Implémentez la récupération automatique à l'aide d'AWS Step Functions et d'AWS Lambda lorsque vous ne pouvez pas utiliser la mise à l'échelle automatique ou la récupération automatique, ou lorsque la récupération automatique échoue. Lorsque vous ne pouvez pas utiliser la scalabilité automatique, que vous ne pouvez pas utiliser la récupération automatique ou que la récupération automatique échoue, vous pouvez automatiser la réparation à l'aide d'AWS Step Functions et d'AWS Lambda.
- Amazon EventBridge peut être utilisé pour surveiller et filtrer les événements tels que les alarmes CloudWatch ou les changements d'état d'autres services AWS. En fonction des informations d'événement, il peut ensuite déclencher AWS Lambda (ou d'autres cibles) pour exécuter une logique de correction personnalisée sur votre charge de travail.

Ressources

Bonnes pratiques associées:

- Définition de la disponibilité
- REL11-BP01 Surveiller tous les composants de la charge de travail pour détecter les défaillances

Documents connexes :

- Fonctionnement d'AWS Auto Scaling
- Récupération automatique Amazon EC2

- Amazon Elastic Block Store (Amazon EBS)
- Amazon Elastic File System (Amazon EFS)
- Qu’est-ce qu’Amazon FSx pour Lustre ?
- Qu’est-ce qu’Amazon FSx for Windows File Server ?
- AWS OpsWorks : Utilisation de la réparation automatique pour remplacer les instances en échec
- Qu'est-ce que AWS Step Functions?
- Qu'est-ce que AWS Lambda?
- Qu’est-ce qu’Amazon EventBridge?
- Utilisation d’alarmes Amazon CloudWatch
- Basculement d’Amazon RDS
- SSM - Systems Manager Automation
- Bonnes pratiques en matière d’architecture résiliente

Vidéos connexes :

- Automatically Provision and Scale OpenSearch Service
- Amazon RDS Failover Automatically

Exemples connexes :

- Atelier sur le basculement d’Amazon RDS

Outils associés:

- CloudWatch
- CloudWatch X-Ray

REL11-BP04 S’appuyer sur le plan de données et non sur le plan de contrôle pendant la récupération

Les plans de contrôle fournissent les API administratives utilisées pour créer, lire et décrire, mettre à jour, supprimer et répertorier (CRUDL) les ressources, tandis que les plans de données gèrent

le trafic quotidien des services. Lorsque vous mettez en œuvre des réponses de restauration ou d'atténuation en cas d'événements susceptibles d'avoir un impact sur la résilience, concentrez-vous sur l'utilisation d'un nombre minimal d'opérations du plan de contrôle pour récupérer, redimensionner, restaurer, réparer ou basculer le service. L'action du plan de données doit remplacer toute activité lors de ces événements de dégradation.

Par exemple, les actions suivantes font toutes partie du plan de contrôle : lancement d'une nouvelle instance de calcul, création d'un stockage par blocs et description des services de file d'attente. Lorsque vous lancez des instances de calcul, le plan de contrôle doit effectuer plusieurs tâches, telles que la recherche d'un hôte physique avec la capacité suffisante, l'allocation d'interfaces réseau, la préparation de volumes locaux de stockage par blocs, la génération d'informations d'identification et l'ajout de règles de sécurité. Les plans de contrôle relèvent souvent d'une orchestration complexe.

Résultat escompté : lorsqu'une ressource passe à un état altéré, le système peut être rétabli automatiquement ou manuellement en transférant le trafic des ressources altérées vers des ressources saines.

Anti-modèles courants :

- Nécessité de modifier les enregistrements DNS pour rediriger le trafic.
- Nécessité de réaliser des opérations de mise à l'échelle du plan de contrôle pour remplacer les composants endommagés en raison de ressources sous-provisionnées.
- Utilisation d'actions de plan de contrôle étendues, multiservices et multi-API pour remédier à toute catégorie d'altération.

Avantages du respect de cette bonne pratique : l'augmentation du taux de réussite de la correction automatisée contribue à réduire le temps moyen de récupération et à améliorer la disponibilité de la charge de travail.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : moyen : pour certains types de dégradations de service, les plans de contrôle sont affectés. La nécessité d'utiliser de manière intensive le plan de contrôle pour la correction peut augmenter le délai de reprise (RTO) et le temps moyen de récupération (MTTR).

Directives d'implémentation

Pour limiter les actions du plan de données, évaluez chaque service pour déterminer les actions nécessaires afin de restaurer le service.

Tirez parti d'Amazon Application Recovery Controller pour déplacer le trafic DNS. Ces fonctionnalités surveillent en permanence la capacité de votre application à se rétablir suite à des défaillances et vous permettent de contrôler la reprise de votre application dans plusieurs Régions AWS, plusieurs zones de disponibilité et sur site.

Les politiques de routage Route 53 utilisent le plan de contrôle. Ne vous fiez donc pas à celui-ci pour la récupération. Les plans de données Route 53 répondent aux requêtes DNS et effectuent et évaluent les surveillances de l'état. Ils sont distribués dans le monde entier et conçus pour un contrat de niveau de service (SLA) de disponibilité à 100 %.

Les API et consoles de gestion Route 53 dans lesquelles vous créez, mettez à jour et supprimez des ressources Route 53 s'exécutent sur des plans de contrôle conçus pour donner la priorité à la cohérence forte et à la durabilité dont vous avez besoin lors de la gestion du DNS. Pour ce faire, les plans de contrôle sont situés dans une seule région : USA Est (Virginie du Nord). Bien que les deux systèmes soient conçus pour être très fiables, les plans de contrôle ne sont pas inclus dans le SLA. Dans de rares cas, la conception résiliente du plan de données permet de maintenir la disponibilité alors que les plans de contrôle ne le font pas. Pour les mécanismes de reprise après sinistre et de basculement, utilisez les fonctions du plan de données pour assurer la meilleure fiabilité possible.

Concevez votre infrastructure informatique de manière à ce qu'elle soit statiquement stable afin d'éviter d'utiliser le plan de contrôle lors d'un incident. Par exemple, si vous utilisez des instances Amazon EC2, évitez de provisionner de nouvelles instances manuellement ou de demander aux groupes Auto Scaling d'ajouter des instances en réponse. Pour obtenir les niveaux de résilience les plus élevés, allouez une capacité suffisante dans le cluster utilisé pour le basculement. Si ce seuil de capacité doit être limité, définissez des limitations sur l'ensemble du système de bout en bout afin de restreindre en toute sécurité le trafic total atteignant l'ensemble limité de ressources.

Pour des services comme Amazon DynamoDB, Amazon API Gateway, les équilibreurs de charge et AWS Lambda sans serveur, leur utilisation permet de tirer parti du plan de données. Cependant, la création de fonctions, d'équilibreurs de charge, de passerelles d'API ou de tables DynamoDB est une action du plan de contrôle qui doit être terminée avant la dégradation afin de préparer un événement et de répéter les actions de basculement. Pour Amazon RDS, les actions du plan de données permettent d'accéder aux données.

Pour plus d'informations sur les plans de données, les plans de contrôle et sur comment AWS crée des services pour atteindre les objectifs de haute disponibilité, consultez Stabilité statique à l'aide de zones de disponibilité.

Comprendre quelles opérations relèvent du plan de données et quelles opérations relèvent du plan de contrôle.

Étapes d'implémentation

Pour chaque charge de travail qui doit être restaurée après un événement de dégradation, évaluez le runbook de basculement, la conception de la haute disponibilité, la conception de la réparation automatique ou le plan de restauration des ressources haute disponibilité. Identifiez chaque action qui pourrait être considérée comme une action du plan de contrôle.

Envisagez de remplacer l'action du plan de contrôle par une action de plan de données :

- Autoscaling (plan de contrôle) vers les ressources Amazon EC2 préalablement mises à l'échelle (plan de données)
- Mise à l'échelle d'instance Amazon EC2 (plan de contrôle) par rapport à la mise à l'échelle AWS Lambda (plan de données)
- Évaluez toutes les conceptions utilisant Kubernetes, ainsi que la nature des actions du plan de contrôle. L'ajout de pods est une action du plan de données dans Kubernetes. Les actions doivent se limiter à l'ajout de pods et non à l'ajout de nœuds. L'utilisation de nœuds surapprovisionnés est la méthode préférée pour limiter les actions du plan de contrôle

Envisagez d'autres approches qui permettent aux actions du plan de données d'affecter les mêmes mesures correctives.

- Modification d'enregistrement Route 53 (plan de contrôle) ou Amazon Application Recovery Controller (plan de données)
- Surveillance de l'état Route 53 pour des mises à jour plus automatisées

Envisagez certains services dans une région secondaire, s'ils sont critiques, afin de permettre davantage d'actions du plan de contrôle et du plan de données dans une région non affectée.

- Amazon EC2 Auto Scaling ou Amazon EKS dans une région principale par rapport à Amazon EC2 Auto Scaling ou Amazon EKS dans une région secondaire et acheminement du trafic vers la région secondaire (action du plan de contrôle)
- Réalisez un réplica en lecture dans la région secondaire ou tentez la même action dans la région principale (action du plan de contrôle).

Ressources

Bonnes pratiques associées :

- Définition de la disponibilité
- REL11-BP01 Surveiller tous les composants de la charge de travail pour détecter les défaillances

Documents connexes :

- Partenaire APN : partenaires pouvant vous aider à automatiser votre tolérance aux pannes
- AWS Marketplace : produits pouvant être utilisés pour la tolérance aux pannes
- L'Amazon Builders' Library : éviter la surcharge des systèmes distribués en plaçant sous contrôle le plus petit service
- API Amazon DynamoDB (plan de contrôle et plan de données)
- Exécutions AWS Lambda (réparties entre le plan de contrôle et le plan de données)
- Plan de données AWS Elemental MediaStore
- Création d'applications hautement résilientes à l'aide d'Amazon Application Recovery Controller, partie 1 : pile dans une seule région
- Création d'applications hautement résilientes à l'aide d'Amazon Application Recovery Controller, partie 2 : pile dans plusieurs régions
- Création de mécanismes de reprise après sinistre à l'aide d'Amazon Route 53
- Qu'est-ce qu'Amazon Application Recovery Controller
- Plan de contrôle et plan de données Kubernetes

Vidéos connexes :

- Back to Basics - Using Static Stability
- Building resilient multi-site workloads using AWS global services

Exemples connexes :

- Présentation d'Amazon Application Recovery Controller
- L'Amazon Builders' Library : éviter la surcharge des systèmes distribués en plaçant sous contrôle le plus petit service

- Création d'applications hautement résilientes à l'aide d'Amazon Application Recovery Controller, partie 1 : pile dans une seule région
- Création d'applications hautement résilientes à l'aide d'Amazon Application Recovery Controller, partie 2 : pile dans plusieurs régions
- Stabilité statique avec les zones de disponibilité

Outils associés :

- Amazon CloudWatch
- AWS X-Ray

REL11-BP05 Utiliser la stabilité statique pour éviter les comportements bimodaux

Les charges de travail doivent être statiquement stables et ne fonctionner que dans un seul mode normal. On parle de comportement bimodal lorsque la charge de travail présente un comportement différent en mode normal et en mode d'échec.

Par exemple, vous pouvez essayer de récupérer une défaillance de la zone de disponibilité en lançant de nouvelles instances dans une zone de disponibilité différente. Il peut en résulter une réponse bimodale lors d'un mode de défaillance. Pour éviter ce type de comportement, vous devez créer des charges de travail stables statiquement et qui fonctionnent dans un seul mode. Dans cet exemple, ces instances auraient dû être provisionnées dans la deuxième zone de disponibilité avant la panne. Ce modèle de stabilité statique permet de vérifier que la charge de travail ne fonctionne que dans un seul mode.

Résultat escompté : les charges de travail ne présentent pas de comportement bimodal en mode normal et en mode d'échec.

Anti-modèles courants :

- Supposer que les ressources peuvent toujours être provisionnées quelle que soit l'étendue de la défaillance.
- Essayer d'acquérir dynamiquement des ressources lors d'une panne.
- Ne pas provisionner les ressources adéquates dans les zones ou les régions jusqu'à ce qu'une panne se produise.

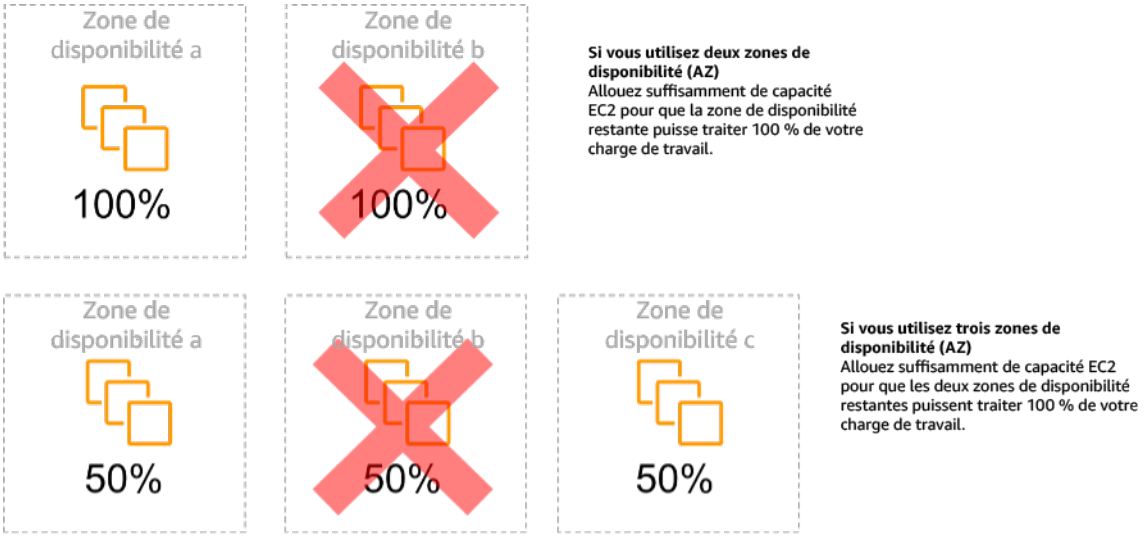
- Envisager des modèles statiques et stables pour les ressources informatiques uniquement.

Avantages liés au respect de cette bonne pratique : les charges de travail exécutées avec des modèles statiquement stables sont capables d'avoir des résultats prévisibles lors d'événements normaux et de défaillances.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : moyen

Directives d'implémentation

Un comportement bimodal survient lorsque votre charge de travail adopte un comportement différent en mode normal et en mode de défaillance (par exemple, en s'appuyant sur le lancement de nouvelles instances en cas de défaillance d'une zone de disponibilité). Exemple de comportement bimodal : les modèles Amazon EC2 stables provisionnent suffisamment d'instances dans chaque zone de disponibilité pour gérer la charge de travail si une zone de disponibilité était supprimée. Elastic Load Balancing ou Amazon Route 53 vérifieraient l'état pour éloigner une charge de l'instance défaillante. Une fois le trafic déplacé, utilisez AWS Auto Scaling pour remplacer de manière asynchrone les instances de la zone défaillante et les lancer dans les zones saines. La stabilité statique du déploiement de calcul (par exemple, des conteneurs ou des instances EC2) garantit une fiabilité optimale.



Stabilité statique des instances EC2 dans les zones de disponibilité

Cela doit être comparé au coût de ce modèle et à la valeur commerciale du maintien de la charge de travail dans tous les cas de résilience. Il est moins coûteux de provisionner moins de capacité de calcul et de compter sur le lancement de nouvelles instances en cas de panne. Cependant, pour les

pannes à grande échelle (comme une zone de disponibilité ou une panne régionale), cette approche se révèle moins efficace, car elle repose à la fois sur un plan opérationnel et sur la disponibilité de ressources suffisantes dans les zones ou les régions non affectées.

Votre solution doit également tenir compte de la fiabilité par rapport aux coûts nécessaires pour votre charge de travail. Les architectures de stabilité statique s'appliquent à différentes architectures, notamment les instances de calcul réparties dans les zones de disponibilité, les modèles de réplicas en lecture de bases de données, les modèles de clusters Kubernetes (EKS) et les architectures de basculement multi-régions.

Il est également possible de mettre en œuvre un modèle plus stable sur le plan statique en utilisant davantage de ressources dans chaque zone. En ajoutant davantage de zones, vous réduisez la quantité de calcul supplémentaire nécessaire à la stabilité statique.

Autre exemple de comportement bimodal : un délai d'expiration du réseau peut amener un système à tenter d'actualiser l'état de configuration de l'ensemble du système. Cela ajouterait une charge inattendue à un autre composant et pourrait provoquer sa défaillance, entraînant d'autres conséquences inattendues. Cette boucle de rétroaction négative a un impact sur la disponibilité de votre charge de travail. Vous pourriez donc créer des systèmes stables statiquement et fonctionnant dans un seul mode. Un modèle statiquement stable consisterait à effectuer un travail constant et à toujours actualiser l'état de la configuration selon une cadence fixe. Lorsqu'un appel échoue, la charge de travail utilise la valeur précédemment mise en cache et déclenche une alarme.

Un autre exemple de comportement bimodal consiste à autoriser les clients à contourner votre cache de charge de travail lorsque des défaillances se produisent. Cette solution peut sembler répondre aux besoins des clients, mais elle peut modifier considérablement les exigences de votre charge de travail et risque d'entraîner des échecs.

Évaluez les charges de travail critiques afin de déterminer celles qui nécessitent ce type de modèle de résilience. Pour celles qui sont jugées critiques, chaque composant de l'application doit être examiné. Voici quelques exemples de services nécessitant une évaluation de la stabilité statique :

- Calcul : Amazon EC2, EKS-EC2, ECS-EC2, EMR-EC2
- Bases de données : Amazon Redshift, Amazon RDS, Amazon Aurora
- Stockage : Amazon S3 (zone unique), Amazon EFS (montages), Amazon FSx (montages)
- Équilibreurs de charge : selon certains modèles

Étapes d'implémentation

- Créez des systèmes stables statiquement et qui fonctionnent dans un seul mode. Dans ce cas, provisionnez suffisamment d'instances dans chaque zone de disponibilité ou région pour gérer la capacité de la charge de travail si une zone de disponibilité ou une région était supprimée. Plusieurs services peuvent être utilisés pour l'acheminement vers des ressources saines, par exemple :
  - Routage DNS entre régions
  - Routage multi-régional MRAP Amazon S3
  - AWS Global Accelerator
  - Amazon Application Recovery Controller
- Configurez des réplicas en lecture de base de données pour tenir compte de la perte d'une instance primaire unique ou d'un réplica en lecture. Si le trafic est desservi par des réplicas en lecture, la quantité dans chaque zone de disponibilité et chaque région doit correspondre au besoin global en cas de défaillance de la zone ou de la région.
- Configurez les données critiques dans un stockage Amazon S3 conçu pour être statiquement stable pour les données stockées en cas de défaillance d'une zone de disponibilité. Si la classe de stockage Amazon S3 unizone – Accès peu fréquent est utilisée, elle ne doit pas être considérée comme statiquement stable, car la perte de cette zone minimise l'accès aux données stockées.
- Des équilibreurs de charge sont parfois configurés de manière incorrecte ou sciemment pour desservir une zone de disponibilité spécifique. Dans ce cas, le modèle statiquement stable peut consister à répartir une charge de travail sur plusieurs zones de disponibilité dans le cadre d'un modèle plus complexe. Le modèle original peut être utilisé pour réduire le trafic interzone pour des raisons de sécurité, de latence ou de coût.

Ressources

Bonnes pratiques Well-Architected connexes:

- Définition de la disponibilité
- REL11-BP01 Surveiller tous les composants de la charge de travail pour détecter les défaillances
- REL11-BP04 S'appuyer sur le plan de données et non sur le plan de contrôle pendant la récupération

Documents connexes :

- Minimiser les dépendances dans un plan de reprise après sinistre
- The Amazon Builders' Library: Static stability using Availability Zones
- Fault Isolation Boundaries
- Stabilité statique avec les zones de disponibilité
- RDS multizone
- Minimiser les dépendances dans un plan de reprise après sinistre
- Routage DNS entre régions
- Routage multi-régional MRAP Amazon S3
- AWS Global Accelerator
- Amazon Application Recovery Controller
- Amazon S3 à zone unique
- Équilibrage de charge entre zones

Vidéos connexes :

- Static stability in AWS: AWS re:Invent 2019: Introducing The Amazon Builders' Library (DOP328)

REL11-BP06 Envoyer des notifications lorsque des événements affectent la disponibilité

Des notifications sont envoyées en cas de détection de dépassement de seuils, même si l'événement à l'origine du problème a été automatiquement résolu.

La réparation automatisée permet à votre charge de travail d'être fiable. Cependant, elle peut également masquer les problèmes sous-jacents à résoudre. Implémentez une surveillance et des événements appropriés afin de pouvoir détecter les schémas de problèmes, y compris ceux résolus par la réparation automatique, afin de pouvoir résoudre les problèmes de cause racine.

Les systèmes résilients sont conçus de manière à ce que les événements de dégradation soient immédiatement communiqués aux équipes concernées. Ces notifications doivent être envoyées par un ou plusieurs canaux de communication.

Résultat escompté : des alertes sont immédiatement envoyées aux équipes chargées des opérations lorsque des seuils sont dépassés, tels que les taux d'erreur, la latence ou d'autres métriques

d'indicateurs clés de performance (KPI) critiques, afin que ces problèmes soient résolus dès que possible et que l'impact sur les utilisateurs soit évité ou minimisé.

Anti-modèles courants :

- Envoyer un trop grand nombre d'alarmes.
- Envoyer des alarmes non exploitables.
- Régler les seuils d'alarme à un niveau trop élevé (sensibilité excessive) ou trop faible (sensibilité insuffisante).
- Ne pas envoyer d'alarmes pour les dépendances externes.
- Ne pas prendre en compte les défaillances grises lors de la conception de la surveillance et des alarmes.
- Effectuer des réparations automatisées, mais ne pas notifier l'équipe appropriée que des réparations étaient nécessaires.

Avantages du respect de cette bonne pratique : les notifications de reprise permettent aux équipes commerciales et chargées des opérations d'être informées des dégradations de service, de sorte qu'elles peuvent réagir immédiatement pour minimiser à la fois le temps moyen de détection (MTTD) et le temps moyen de réparation (MTTR). Les notifications d'événements de reprise vous permettent également de ne pas ignorer les problèmes qui se produisent peu fréquemment.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : moyen. L'absence de mise en œuvre de mécanismes appropriés de surveillance et de notification des événements peut entraîner l'incapacité à détecter des schémas de problèmes, y compris ceux traités par la réparation automatisée. Une équipe ne sera informée de la dégradation du système que lorsque les utilisateurs contacteront le service clientèle ou par hasard.

Directives d'implémentation

Lors de la définition d'une stratégie de surveillance, le déclenchement d'une alarme est un événement courant. Cet événement contiendra probablement un identifiant pour l'alarme, l'état de l'alarme (comme `IN ALARM` ou `OK`) et les détails de ce qui l'a déclenchée. Dans de nombreux cas, un événement d'alarme doit être détecté et une notification par e-mail doit être envoyée. Voici un exemple d'action sur une alarme. La notification d'alarme est essentielle pour l'observabilité, car elle permet d'informer les bonnes personnes de l'existence d'un problème. Cependant, lorsque l'action sur les événements arrive à maturité dans votre solution d'observabilité, elle peut automatiquement remédier au problème sans nécessiter d'intervention humaine.

Une fois que les alarmes de suivi des KPI ont été établies, des alertes doivent être envoyées aux équipes concernées lorsque les seuils sont dépassés. Ces alertes peuvent également être utilisées pour déclencher des processus automatisés qui tenteront de remédier à la dégradation.

Pour une surveillance plus complexe des seuils, des alarmes composites doivent être envisagées. Les alarmes composites utilisent un certain nombre d'alarmes de surveillance des KPI pour créer une alerte basée sur la logique métier opérationnelle. Les alarmes CloudWatch peuvent être configurées pour envoyer des e-mails pour consigner des incidents dans des systèmes tiers de suivi des incidents à l'aide de l'intégration d'Amazon SNS ou d'Amazon EventBridge.

Étapes d'implémentation

Créez différents types d'alarmes en fonction des charges de travail surveillées, par exemple :

- Les alarmes d'application permettent de détecter si une partie de votre charge de travail ne fonctionne pas correctement.
- Les alarmes relatives à l'infrastructure indiquent à quel moment il faut mettre les ressources à l'échelle. Le système peut afficher les alarmes sur des tableaux de bord, envoyer des alertes via Amazon SNS ou par e-mail et fonctionner avec Auto Scaling pour une mise à l'échelle des ressources de la charge de travail entrante ou sortante.
- Des alarmes statiques simples peuvent être créées pour surveiller le dépassement d'un seuil statique par une métrique pendant un nombre spécifié de périodes d'évaluation.
- Des alarmes composites peuvent prendre en compte des alarmes complexes provenant de sources multiples.
- Une fois l'alarme créée, créez les événements de notification appropriés. Vous pouvez directement invoquer une API Amazon SNS pour envoyer des notifications et associer toute automatisation à des fins de correction ou de communication.
- Restez informé des dégradations de service avec AWS Health. Créez des notifications d'événements AWS Health spécialement adaptées aux e-mails et aux canaux de discussion via Notifications des utilisateurs AWS et intégrez-les de manière programmatique à vos outils de surveillance et d'alerte via Amazon EventBridge.

Ressources

Bonnes pratiques Well-Architected connexes:

- Définition de la disponibilité

Documents connexes:

- Création d'une alarme CloudWatch basée sur un seuil statique
- Qu'est-ce qu'Amazon EventBridge?
- Qu'est-ce qu'Amazon Simple Notification Service?
- Publication des métriques personnalisées
- Utilisation d'alarmes Amazon CloudWatch
- Configuration des alarmes CloudWatch Composite
- Les nouveautés en matière d'observabilité AWS à re:Invent 2022

Outils associés:

- CloudWatch
- CloudWatch X-Ray

REL11-BP07 Concevoir votre produit pour atteindre les objectifs de disponibilité et les contrats de niveau de service (SLA)

Concevez votre produit de manière à atteindre les objectifs de disponibilité et les contrats de niveau de service (SLA). Si vous publiez ou convenez en privé d'objectifs de disponibilité ou d'accords de niveau de service, vérifiez que votre architecture et vos processus opérationnels sont conçus pour les prendre en charge.

Résultat souhaité : chaque application dispose d'un objectif défini en matière de disponibilité et d'un contrat de niveau de service pour les indicateurs de performance, qui peuvent être surveillés et maintenus afin d'atteindre les résultats commerciaux.

Anti-modèles courants :

- Concevoir et déployer des charges de travail sans fixer de contrats de niveau de service.
- Les métriques des SLA sont fixées à un niveau trop élevé sans justification ni exigences commerciales.
- Fixer des contrats de niveau de service sans tenir compte des dépendances et des contrats de niveau de service sous-jacents.
- Les conceptions d'applications sont créées sans tenir compte du modèle de responsabilité partagée pour la résilience.

Avantages du respect de cette bonne pratique : la conception d'applications reposant sur des objectifs de résilience clés vous aide à atteindre les objectifs commerciaux et les attentes des clients. Ces objectifs contribuent à orienter le processus de conception de l'application qui évalue les différentes technologies et envisage divers compromis.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : moyen

Directives d'implémentation

La conception des applications doit tenir compte d'un ensemble diversifié d'exigences découlant d'objectifs commerciaux, opérationnels et financiers. Dans le cadre des exigences opérationnelles, les charges de travail doivent avoir des objectifs spécifiques en matière de métriques de résilience afin qu'elles puissent être correctement surveillées et prises en charge. Les métriques de résilience ne doivent pas être définies ou déduites après le déploiement de la charge de travail. Elles doivent être définies pendant la phase de conception et aider à guider les diverses décisions et compromis.

- Chaque charge de travail doit disposer de son propre ensemble de métriques de résilience. Ces métriques peuvent être différentes de celles d'autres applications commerciales.
- La réduction des dépendances peut avoir un impact positif sur la disponibilité. Chaque charge de travail doit tenir compte de ses dépendances et de leurs contrats de niveau de service. En général, sélectionnez les dépendances dont les objectifs de disponibilité sont égaux ou supérieurs à ceux de votre charge de travail.
- Envisagez des conceptions faiblement couplées afin que votre charge de travail puisse fonctionner correctement malgré l'altération des dépendances, lorsque cela est possible.
- Réduisez les dépendances du plan de contrôle, notamment lors de la reprise ou d'une dégradation. Évaluez les conceptions statiques stables pour les charges de travail critiques. Utilisez le partage des ressources pour augmenter la disponibilité de ces dépendances dans une charge de travail.
- L'observabilité et l'instrumentation sont essentielles pour respecter les contrats de niveau de service en réduisant le temps moyen de détection (MTTD) et le temps moyen de réparation (MTTR).
- Des défaillances moins fréquentes (MTBF plus long), des temps de détection des défaillances plus courts (MTTD plus court) et des temps de réparation plus courts (MTTR plus court) sont les trois facteurs utilisés pour améliorer la disponibilité des systèmes distribués.
- L'établissement et le respect des métriques de résilience pour une charge de travail sont à la base de toute conception efficace. Ces conceptions doivent tenir compte des compromis entre la complexité de la conception, les dépendances des services, les performances, la mise à l'échelle et les coûts.

Étapes d'implémentation

- Examinez et documentez la conception de la charge de travail en tenant compte des questions suivantes :
  - Où les plans de contrôle sont-ils utilisés dans la charge de travail ?
  - Comment la charge de travail met-elle en œuvre la tolérance aux pannes ?
  - Quels sont les modèles de conception pour la mise à l'échelle, la mise à l'échelle automatique, la redondance et les composants hautement disponibles ?
  - Quelles sont les exigences en matière de cohérence et de disponibilité des données ?
  - Y a-t-il des considérations relatives à l'économie des ressources ou à la stabilité statique des ressources ?
  - Quelles sont les dépendances des services ?
- Définissez les métriques SLA en fonction de l'architecture de la charge de travail tout en travaillant avec les parties prenantes. Tenez compte des SLA de toutes les dépendances utilisées par la charge de travail.
- Une fois l'objectif du SLA fixé, optimisez l'architecture pour le respecter.
- Une fois que la conception a été définie de manière à respecter le contrat de niveau de service, il faut mettre en œuvre les changements opérationnels, l'automatisation des processus et les runbooks qui visent également à réduire les délais d'attente et les temps de réponse.
- Une fois le déploiement effectué, surveillez et rendez compte du contrat de niveau de service.

Ressources

Bonnes pratiques associées :

- REL03-BP01 Choisissez comment segmenter votre charge de travail
- REL10-BP01 Déploiement de la charge de travail sur plusieurs emplacements
- REL11-BP01 Surveiller tous les composants de la charge de travail pour détecter les défaillances
- REL11-BP03 Automatiser la réparation sur toutes les couches
- REL12-BP04 Tester la résilience à l'aide de l'ingénierie du chaos
- REL13-BP01 Définir les objectifs de reprise pour les temps d'arrêt et les pertes de données
- Comprendre l'état de la charge de travail

Documents connexes :

- Disponibilité avec redondance
- Pilier de fiabilité - Disponibilité
- Mesurer la disponibilité
- AWS Fault Isolation Boundaries
- Modèle de responsabilité partagée pour la résilience
- Stabilité statique avec les zones de disponibilité
- Accords de niveau de service (SLA) AWS
- Conseils pour l'architecture cellulaire sur AWS
- Infrastructure AWS
- Livre blanc sur les modèles de résilience multi-AZ avancés

Services connexes :

- Amazon CloudWatch
- AWS Config
- AWS Trusted Advisor

Test de la fiabilité

Une fois que vous avez conçu votre charge de travail pour qu'elle soit résiliente aux sollicitations de la production, les tests sont le seul moyen de s'assurer qu'elle fonctionne comme prévu et d'obtenir la résilience voulue.

Exécutez un test pour vérifier que votre charge de travail répond aux exigences fonctionnelles et non fonctionnelles, car les bogues ou les goulots d'étranglement des performances peuvent avoir un impact sur sa fiabilité. Testez la résilience de votre charge de travail pour vous aider à détecter les bogues latents présents uniquement dans la production. Pratiquez régulièrement ces tests.

Bonnes pratiques

- REL12-BP01 Utiliser des playbooks pour enquêter sur les causes des défaillances
- REL12-BP02 Effectuer une analyse post-incident
- REL12-BP03 Tester les exigences de capacité de mise à l'échelle et de performances
- REL12-BP04 Tester la résilience à l'aide de l'ingénierie du chaos

- REL12-BP05 Organiser régulièrement des tests de simulation de panne

REL12-BP01 Utiliser des playbooks pour enquêter sur les causes des défaillances

Consignez le processus d'enquête dans des playbooks afin de faciliter l'application de réponses cohérentes et rapides face aux scénarios de défaillance qui ne sont pas bien compris. Les playbooks sont les étapes prédéfinies suivies pour identifier les facteurs adjuvants à un scénario de défaillance. Les résultats des étapes du processus sont utilisés pour déterminer les prochaines mesures à prendre jusqu'à ce que la question soit identifiée ou remontée.

Le playbook est une planification proactive que vous devez appliquer afin de pouvoir prendre efficacement des mesures réactives. Lorsque des scénarios de défaillance ne figurant pas dans le playbook sont rencontrés en production, commencez par résoudre le problème (éteindre l'incendie). Procédez ensuite à une rétrospective en examinant les étapes suivies pour résoudre le problème et utilisez-les pour ajouter une nouvelle entrée dans le playbook.

Notez que les playbooks sont utilisés en réponse à des incidents spécifiques, tandis que les runbooks le sont pour obtenir des résultats spécifiques. En règle générale, les runbooks sont employés pour les activités de routine et les playbooks pour répondre à des événements non réguliers.

Anti-modèles courants :

- Planification du déploiement d'une charge de travail sans connaître les processus permettant de diagnostiquer les problèmes ou de réagir aux incidents.
- Décisions imprévues sur les systèmes à partir desquels peut se faire la collecte des journaux et métriques lors de l'examen d'un événement.
- Non-conservation des métriques et événements pendant suffisamment longtemps pour pouvoir récupérer les données.

Avantages du respect de cette bonne pratique : la capture de playbooks garantit le respect constant des processus. La codification de vos playbooks limite l'introduction d'erreurs à partir de l'activité manuelle. L'automatisation des playbooks accélère le temps de réponse à un événement en évitant aux membres de l'équipe d'intervenir ou en leur fournissant des informations supplémentaires lorsque leur intervention commence.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : élevé

Directives d'implémentation

- Utilisez des playbooks pour identifier les problèmes. Les playbooks sont des processus documentés pour enquêter sur les problèmes. Mettez en œuvre des réponses cohérentes et rapides aux échecs en documentant les processus dans des playbooks. Les playbooks doivent contenir les informations et les instructions nécessaires pour permettre à une personne compétente de recueillir les informations pertinentes, identifier les causes potentielles de défaillance, isoler les pannes et déterminer les facteurs adjuvants (c'est-à-dire effectuer une analyse post-incident).
  - Mettez en œuvre les playbooks en tant que code. Effectuez vos opérations en tant que code en scriptant vos playbooks afin d'en assurer la cohérence et de limiter les erreurs causées par les processus manuels. Les playbooks peuvent être composés de plusieurs scripts représentant les différentes étapes qui pourraient être nécessaires pour identifier les facteurs contribuant à un problème. Les activités de runbook peuvent être invoquées ou effectuées dans le cadre d'activités de playbook, ou peuvent demander l'exécution d'un playbook en réponse à des événements identifiés.
    - Automatisez vos playbooks opérationnels avec AWS Systems Manager
    - Run Command d'AWS Systems Manager
    - AWS Systems Manager Automation
    - Présentation de AWS Lambda
    - Qu'est-ce qu'Amazon EventBridge ?
    - Utilisation d'alarmes Amazon CloudWatch

Ressources

Documents connexes :

- AWS Systems Manager Automation
- Run Command d'AWS Systems Manager
- Automatisez vos playbooks opérationnels avec AWS Systems Manager
- Utilisation d'alarmes Amazon CloudWatch
- Utilisation de scripts Canary (Amazon CloudWatch Synthetics)
- Qu'est-ce qu'Amazon EventBridge ?
- Présentation de AWS Lambda

Exemples connexes :

- Automatisation des opérations avec les playbooks et les runbooks

REL12-BP02 Effectuer une analyse post-incident

Passez en revue les événements ayant un impact sur le client et identifiez les facteurs adjuvants et les mesures préventives. Utilisez ces informations pour développer des mesures d'atténuation afin de limiter ou d'empêcher la récurrence. Développez des procédures pour fournir des réponses rapides et efficaces. Publiez, le cas échéant, les facteurs adjuvants et les mesures correctives adaptées au public ciblé. Vous devez disposer d'une méthode pour communiquer ces causes à d'autres si nécessaire.

Évaluez pourquoi les tests existants n'ont pas permis de résoudre le problème. Ajoutez des tests pour ce cas si aucun test correspondant n'existe.

Résultat escompté : vos équipes ont adopté une approche cohérente et convenue pour gérer l'analyse post-incident. L'un des mécanismes est le processus de correction d'erreur (COE). Celui-ci aide vos équipes à identifier, comprendre et traiter les causes profondes des incidents, tout en mettant en place des mécanismes et des barrières de protection pour limiter la probabilité qu'un incident se reproduise.

Anti-modèles courants :

- Trouver des facteurs adjuvants sans pour autant continuer à chercher plus en profondeur d'autres problèmes et approches potentiels pour atténuer les risques.
- Identification limitée aux causes d'erreur humaine et sans formation ou automatisation pouvant empêcher les erreurs humaines.
- Se concentrer sur les reproches plutôt que sur la compréhension des causes profondes, ce qui crée une culture de la peur et empêche de communiquer ouvertement
- Absence de partage d'informations, qui entrave la circulation des résultats de l'analyse de l'incident et empêche les autres de bénéficier des enseignements tirés
- Absence de mécanisme permettant de capturer les connaissances institutionnelles, ce qui engendre une perte d'informations précieuses en ne conservant pas les enseignements tirés sous la forme de bonnes pratiques actualisées, et entraîne la répétition d'incidents ayant des causes profondes identiques ou similaires

Avantages du respect de cette bonne pratique : une analyse post-incident et le partage des résultats permettent à d'autres charges de travail d'atténuer les risques si elles ont mis en œuvre les mêmes facteurs adjuvants. Elle permet aussi de mettre en œuvre l'atténuation des risques ou la récupération automatisée avant qu'un incident ne se produise.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : élevé

Directives d'implémentation

Une bonne analyse post-incident permet de proposer des solutions courantes pour les problèmes avec des modèles d'architecture utilisés dans d'autres compartiments de vos systèmes.

La documentation et la résolution des problèmes sont l'une des pierres angulaires du processus COE. Il est recommandé de définir une méthode normalisée pour documenter les causes profondes et de veiller à ce qu'elles soient examinées et traitées. Attribuez clairement la responsabilité du processus d'analyse post-incident. Désignez une équipe ou une personne chargée de superviser les enquêtes et le suivi de l'incident.

Encouragez une culture axée sur l'apprentissage et l'amélioration plutôt que sur les reproches. Insistez sur le fait que l'objectif est de prévenir de futurs incidents, et non de pénaliser des individus.

Élaborez des procédures bien définies pour mener les analyses post-incident. Ces procédures doivent décrire les étapes à suivre, les informations à collecter et les principales questions à aborder lors de l'analyse. Enquêtez en profondeur sur les incidents, en allant au-delà des causes immédiates afin d'identifier les causes profondes et les facteurs contributifs. Utilisez des techniques telles que les « cinq pourquoi » pour approfondir les problèmes sous-jacents.

Tenez un répertoire des enseignements tirés des analyses des incidents. Ces connaissances institutionnelles peuvent servir de référence pour les incidents futurs et les efforts de prévention. Partagez les résultats et les réflexions tirées des analyses post-incident, et envisagez d'organiser des réunions de synthèse post-incident ouvertes à tous pour discuter des enseignements tirés.

Étapes d'implémentation

- Veillez à ce que l'analyse post-incident soit exempte de tout reproche. Cela permet aux personnes impliquées dans l'incident de faire preuve d'objectivité quant aux actions correctives proposées, et de promouvoir une auto-évaluation et une collaboration honnêtes entre les équipes.
- Définissez une méthode standardisée pour documenter les problèmes critiques. Voici un exemple de structure :

  - Que s’est-il passé ?
  - Quel a été l’impact sur vos clients et votre activité ?
  - Quelle était la cause profonde ?
  - Quelles sont les données à votre disposition pour étayer votre raisonnement ?
    - Par exemple, des métriques et des graphiques.
  - Quelles ont été les principales répercussions, notamment en termes de sécurité ?
    - Lors de la conception des charges de travail, vous faites un compromis entre les piliers en fonction de votre activité. Ces décisions professionnelles peuvent orienter vos priorités en matière d’ingénierie. Vous pouvez opter pour l’optimisation afin de réduire les coûts au détriment de la fiabilité dans les environnements de développement ou, pour les solutions stratégiques, vous pouvez optimiser la fiabilité pour des coûts plus importants. La sécurité est toujours une priorité, car vous devez protéger vos clients.
  - Quelles leçons avez-vous apprises ?
  - Quelles mesures correctives allez-vous prendre ?
    - Éléments d’action
    - Éléments connexes
- Élaborez des procédures opérationnelles standard bien définies pour mener les analyses post-incident.
- Mettez en place un processus standardisé de signalement des incidents. Documentez tous les incidents de manière exhaustive, y compris le rapport d’incident initial, les journaux, les communications et les mesures prises pendant l’incident.
- N’oubliez pas qu’un incident n’est pas forcément une panne. Il peut s’agir d’un accident évité de justesse ou d’un système qui fonctionne de manière inattendue tout en remplissant sa fonction.
- Améliorez sans cesse votre processus d’analyse post-incident en fonction des retours et des enseignements tirés.
- Capturez les principales conclusions dans un système de gestion des connaissances et examinez les modèles qui devraient être ajoutés aux guides du développeur ou aux listes de contrôle de prédéploiement.

Ressources

Documents connexes :

- Pourquoi mettre en place la correction des erreurs (COE)

Vidéos connexes :

- Amazon’s approach to failing successfully
- AWS re:Invent 2021 - Amazon Builders’ Library: Operational Excellence at Amazon

REL12-BP03 Tester les exigences de capacité de mise à l’échelle et de performances

Utilisez des techniques telles que les tests de charge pour valider que la charge de travail répond aux exigences de mise à l’échelle et de performances.

Dans le cloud, vous pouvez créer un environnement de test à l’échelle de la production pour votre charge de travail à la demande. Au lieu de vous fier à un environnement de test réduit, qui pourrait entraîner des prévisions inexactes des comportements de production, vous pouvez utiliser le cloud pour provisionner un environnement de test qui reflète étroitement votre environnement de production attendu. Cet environnement vous permet de réaliser des tests dans le cadre d’une simulation plus précise des conditions réelles auxquelles votre application est confrontée.

Parallèlement aux tests de performance, il est essentiel de vérifier que vos ressources de base, vos paramètres de mise à l’échelle, vos quotas de service et votre conception de la résilience fonctionnent comme prévu sous charge. Cette approche globale garantit que votre application peut être mise à l’échelle de manière fiable et fonctionner selon les besoins, même dans les conditions les plus exigeantes.

Résultat escompté : votre charge de travail conserve son comportement attendu même lorsqu’elle est soumise à des pics de charge. Vous abordez de manière proactive tous les problèmes liés aux performances susceptibles de survenir au fur et à mesure que l’application grandit et évolue.

Anti-modèles courants :

- Vous utilisez des environnements de test qui ne correspondent pas étroitement à l’environnement de production.
- Vous considérez les tests de charge comme une activité ponctuelle distincte plutôt que comme une partie intégrante du pipeline d’intégration continue (CI) du déploiement.
- Vous ne définissez pas d’exigences de performances claires et mesurables, telles que des cibles de temps de réponse, de débit et de capacité de mise à l’échelle.
- Vous effectuez des tests avec des scénarios de charge non réalistes ou insuffisants, et vous ne parvenez pas à tester les pics de charge, les pics soudains ou une charge élevée prolongée.

- Vous n’effectuez pas de test de stress de la charge de travail en dépassant les limites de charge attendues.
- Vous utilisez des outils de test de charge ou de profilage des performances inadaptés ou inappropriés.
- Vous ne disposez pas de systèmes complets de surveillance et d’alerte pour effectuer le suivi des métriques de performances et détecter les anomalies.

Avantages liés au respect de cette bonne pratique :

- Les tests de charge vous aident à identifier les goulots d’étranglement potentiels de performances de votre système avant sa mise en production. Lorsque vous simulez le trafic et les charges de travail au niveau de la production, vous pouvez identifier les domaines dans lesquels votre système peut avoir du mal à gérer la charge, tels que de longs délais de réponse, des contraintes de ressources ou des défaillances du système.
- En testant votre système dans différentes conditions de charge, vous pouvez mieux comprendre les exigences en matière de ressources pour prendre en charge votre charge de travail. Ces informations peuvent vous aider à prendre des décisions éclairées concernant l’allocation des ressources et à éviter un surprovisionnement ou un sous-provisionnement des ressources.
- Pour identifier les points de défaillance potentiels, vous pouvez observer les performances de votre charge de travail dans des conditions de charge élevée. Ces informations vous aident à améliorer la fiabilité et la résilience de votre charge de travail en mettant en œuvre des mécanismes de tolérance aux pannes, des stratégies de basculement ou des mesures de redondance, selon le cas.
- Vous identifiez et traitez les problèmes de performances à un stade précoce, ce qui vous permet d’éviter les conséquences coûteuses de pannes du système, de longs délais de réponse et d’utilisateurs mécontents.
- Les données de performances et les informations de profilage détaillées collectées lors des tests peuvent vous aider à résoudre les problèmes liés aux performances susceptibles de survenir en production. Cela peut conduire à une réponse et une résolution plus rapides des incidents, ce qui réduit l’impact sur les utilisateurs et les opérations de votre organisation.
- Dans certains secteurs, les tests de performances proactifs peuvent aider votre charge de travail à respecter les normes de conformité, réduisant ainsi le risque de pénalités ou de problèmes juridiques.

Niveau d’exposition au risque si cette bonne pratique n’est pas respectée : élevé

Directives d'implémentation

La première étape consiste à définir une stratégie de tests complète qui couvre tous les aspects des exigences de mise à l'échelle et de performances. Pour commencer, définissez clairement les objectifs de niveau de service (SLO) de votre charge de travail en fonction des besoins de votre entreprise, tels que le débit, l'histogramme de latence et le taux d'erreur. Concevez ensuite une suite de tests capables de simuler différents scénarios de charge allant d'une utilisation moyenne à des pics soudains et des pics de charge prolongés, et vérifiez que le comportement de la charge de travail respecte vos objectifs de niveau de service. Ces tests doivent être automatisés et intégrés dans votre pipeline d'intégration et de déploiement continus afin de détecter les régressions de performances de façon précoce dans le processus de développement.

Pour tester efficacement la mise à l'échelle et les performances, investissez dans les outils et l'infrastructure appropriés. Cela inclut des outils de test de charge capables de générer un trafic utilisateur réaliste, des outils de profilage des performances pour identifier les goulots d'étranglement et des solutions de surveillance pour suivre les métriques clés. Il est important de vérifier que vos environnements de test correspondent étroitement à l'environnement de production en termes d'infrastructure et de conditions d'environnement afin que les résultats de vos tests soient aussi précis que possible. Pour faciliter la réplication et la mise à l'échelle fiables de configurations de type production, utilisez une infrastructure en tant que code et des applications basées sur des conteneurs.

Les tests de mise à l'échelle et de performances sont un processus continu et non une activité ponctuelle. Mettez en œuvre une surveillance et des alertes complètes pour suivre les performances de l'application en production, et utilisez ces données pour affiner en permanence vos stratégies de test et vos efforts d'optimisation. Analysez régulièrement les données de performances pour identifier les problèmes émergents, tester les nouvelles stratégies de mise à l'échelle et mettre en œuvre des optimisations afin d'améliorer l'efficacité et la fiabilité de l'application. Lorsque vous adoptez une approche itérative et que vous tirez constamment des enseignements des données de production, vous pouvez vérifier que votre application peut s'adapter aux demandes variables des utilisateurs et maintenir une résilience et des performances optimales au fil du temps.

Étapes d'implémentation

1. Établissez des exigences de performances claires et mesurables, telles que des cibles de temps de réponse, de débit et de capacité de mise à l'échelle. Ces exigences doivent être basées sur les modèles d'utilisation de votre charge de travail, les attentes des utilisateurs et les besoins de votre entreprise.

2. Sélectionnez et configurez un outil de test de charge capable d'imiter avec précision les modèles de charge et le comportement des utilisateurs dans votre environnement de production.
3. Configurez un environnement de test correspondant étroitement à l'environnement de production, y compris aux conditions d'infrastructure et d'environnement, afin d'améliorer la précision des résultats de vos tests.
4. Créez une suite de tests couvrant un large éventail de scénarios, allant de modèles d'utilisation moyenne à des pics de charge, à des pics rapides et à des charges élevées prolongées. Intégrez ces tests dans vos processus d'intégration et de déploiement continus afin de détecter les régressions de performances de façon précoce dans le processus de développement.
5. Effectuez des tests de charge pour simuler le trafic utilisateur réel et comprendre le comportement de votre application dans différentes conditions de charge. Pour effectuer un test de stress de votre application, dépassez la charge attendue et observez son comportement, tel qu'une dégradation du temps de réponse, l'épuisement des ressources ou des défaillances du système, afin d'identifier le point de rupture de votre application et d'élaborer des stratégies de mise à l'échelle. Évaluez la capacité de mise à l'échelle de votre charge de travail en augmentant progressivement la charge, et mesurez l'impact sur les performances pour identifier les limites de mise à l'échelle et planifier les besoins futurs de capacité.
6. Mettez en œuvre une surveillance et des alertes complètes pour suivre les métriques de performances, détecter les anomalies et lancer des actions de mise à l'échelle ou des notifications lorsque les seuils sont dépassés.
7. Surveillez et analysez en permanence les données de performances pour identifier les domaines à améliorer. Itérez sur vos stratégies de test et vos efforts d'optimisation.

Ressources

Bonnes pratiques associées :

- REL01-BP04 Surveiller et gérer les quotas
- REL06-BP01 Surveiller tous les composants de la charge de travail (génération)
- REL06-BP03 Envoyer des notifications (traitement et alarmes en temps réel)

Documents connexes :

- Tests de charge des applications
- Test de charge distribuée sur AWS

- Surveillance des performances d'application
- Politique de test d'Amazon EC2

Exemples connexes :

- Test de charge distribuée sur AWS (GitHub)

Outils associés :

- Amazon CodeGuru Profiler
- Amazon CloudWatch RUM
- Apache JMeter
- K6
- Vegeta
- Hey
- ab
- wrk
- Test de charge distribuée sur AWS

REL12-BP04 Tester la résilience à l'aide de l'ingénierie du chaos

Exécutez régulièrement des expériences de chaos dans des environnements dont les conditions se rapprochent autant que possible de la production pour comprendre comment nos systèmes réagissent à des conditions défavorables.

Résultat escompté :

la résilience de la charge de travail est régulièrement vérifiée en appliquant l'ingénierie du chaos sous la forme d'expériences d'injection de pannes ou de charge inattendue, en plus des tests de résilience qui confirment le comportement attendu connu de votre charge de travail lors d'un événement. Associez l'ingénierie du chaos aux tests de résilience pour avoir l'assurance que votre charge de travail peut résister en cas de défaillance des composants et récupérer suite à des perturbations inattendues avec peu ou pas d'impact.

Anti-modèles courants :

- Conception à des fins de résilience, mais pas de vérification du fonctionnement de la charge de travail dans son ensemble en cas de défaillances.
- Pas d'expériences dans des conditions concrètes et pour la charge prévue.
- Pas de traitement de vos expériences en tant que code ou de maintenance de vos expériences tout au long du cycle de développement.
- Pas d'exécution d'expériences de chaos dans le cadre de votre pipeline CI/CD, ainsi qu'en dehors des déploiements.
- Pas d'utilisation des analyses passées post-incident pour déterminer les défaillances à tester.

Avantages du respect de cette bonne pratique : l'injection de défaillances pour vérifier la résilience de votre charge de travail vous permet d'avoir l'assurance que les procédures de récupération de votre conception résiliente fonctionneront en cas de défaillances réelles.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : moyen

Directives d'implémentation

L'ingénierie du chaos offre la possibilité à vos équipes d'injecter en continu des perturbations concrètes (simulations) de manière contrôlée au niveau du fournisseur de services, de l'infrastructure, de la charge de travail et des composants, avec peu ou pas d'impact pour vos clients. Ainsi, vos équipes tirent les leçons de ces défaillances et observent, mesurent et améliorent la résilience de vos charges de travail, tout en confirmant que les alertes se déclenchent et que les équipes sont informées en cas d'événement.

Une pratique de l'ingénierie du chaos en continu peut mettre en évidence des défaillances dans vos charges de travail qui, si elles ne sont pas résolues, peuvent impacter de manière négative la disponibilité et le fonctionnement.

> **Note**
>
> L'ingénierie du chaos est la discipline d'expérimentation d'un système. Elle permet de s'assurer de la capacité du système à résister à des conditions de production difficiles. – Principes de l'ingénierie du chaos

Si un système est capable de résister à ces perturbations, l'expérience de chaos doit être maintenue en tant que test de régression automatisé. De cette façon, les expériences de chaos doivent être réalisées dans le cadre de votre cycle de développement des systèmes et de votre pipeline CI/CD.

Pour veiller à ce que votre charge de travail résiste en cas de défaillance des composants, injectez des événements concrets dans le cadre de vos expériences. Par exemple, expérimentez une perte des instances Amazon EC2 ou un basculement de l'instance de base de données Amazon RDS principale, puis vérifiez que votre charge de travail n'est pas impactée (ou très peu). Utilisez plusieurs défaillances des composants pour simuler des événements capables de causer une perturbation dans une zone de disponibilité.

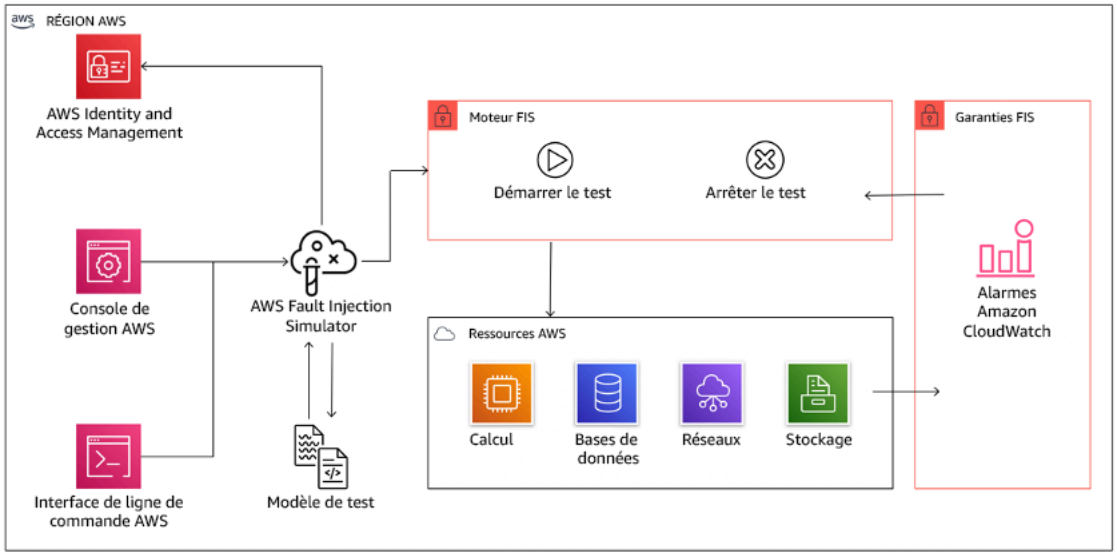
Pour les défaillances de niveau application (telles que les plantages), commencez par des tests de stress comme l'épuisement de la mémoire et du processeur.

Afin de valider des mécanismes de remplacement ou de basculement pour les dépendances externes dues aux pannes réseau intermittentes, vos composants doivent simuler un tel événement en bloquant l'accès aux fournisseurs tiers pendant une durée spécifiée pouvant aller de quelques secondes à plusieurs heures.

D'autres modes de dégradation peuvent entraîner des fonctionnalités limitées et ralentir les réponses, ce qui se traduit par une perturbation de vos services. Généralement, cette dégradation résulte d'une latence accrue sur les services critiques et d'une communication réseau peu fiable (perte de paquets). Les expériences avec ces défaillances, dont les effets de mise en réseau tels que la latence, les messages supprimés et les défaillances DNS, peuvent inclure l'incapacité de résoudre un nom, d'atteindre un service DNS ou de se connecter aux services dépendants.

Outils de l'ingénierie du chaos :

AWS Fault Injection Service (AWS FIS) est un service entièrement géré permettant l'exécution d'expériences d'injection de pannes qui peuvent être utilisées dans le cadre de votre pipeline CD, ou en dehors du pipeline. AWS FIS s'impose donc comme un choix judicieux lors des tests de simulation de pannes. Il prend en charge l'introduction simultanée de défaillances sur différents types de ressources, notamment Amazon EC2, Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS) et Amazon RDS. Ces défaillances incluent l'arrêt des ressources, les basculements forcés, le stress du processeur ou de la mémoire, la limitation, la latence et la perte de paquets. Comme il est intégré aux alarmes Amazon CloudWatch, vous pouvez définir des conditions d'arrêt comme barrières de protection pour annuler une expérience si elle provoque un impact inattendu.



AWS Fault Injection Service s'intègre aux ressources AWS pour vous permettre d'exécuter des expériences d'injection de pannes pour vos charges de travail.

Il existe également plusieurs options tierces pour les expériences d'injection de pannes. Il s'agit notamment d'outils open source tels que Chaos Toolkit, Chaos Mesh et Litmus Chaos, ainsi que d'options commerciales telles que Gremlin. Pour élargir le champ des pannes pouvant être injectées sur AWS, AWS FIS s'intègre à Chaos Mesh et Litmus Chaos, ce qui vous permet de coordonner les flux de travail d'injection de pannes entre plusieurs outils. Par exemple, vous pouvez exécuter un test de stress sur un processeur de pod à l'aide des défaillances Chaos Mesh ou Litmus, tout en arrêtant un pourcentage de nœuds de cluster sélectionné de façon aléatoire grâce aux actions des défaillances AWS FIS.

Étapes d'implémentation

1. Déterminez les défaillances à utiliser pour les expériences.

   Évaluez la conception de votre charge de travail à des fins de résilience. Ces conceptions (créées selon les bonnes pratiques du cadre Well-Architected) tiennent compte des risques basés sur les dépendances critiques, les événements passés, les problèmes connus et les exigences de conformité. Répertoriez chaque élément de la conception destiné à maintenir la résilience et les défaillances qu'il entend réduire. Pour plus d'informations sur la création de telles listes, consultez le livre blanc consacré à l'examen de la préparation opérationnelle, qui explique comment créer un processus visant à empêcher que de tels incidents ne se reproduisent. Le processus de Failure Modes and Effects Analysis (FMEA) ou d'analyse des modes de défaillance et de leurs effets vous propose un framework pour réaliser une analyse de niveau composant des défaillances et de leur

impact sur votre charge de travail. Le FMEA est décrit plus en détail par Adrian Cockcroft dans Failure Modes and Continuous Resilience.

2. Attribuer une priorité à chaque défaillance.

   Commencez par définir une classification grossière telle que élevée, moyenne et basse. Pour évaluer les priorités, tenez compte de la fréquence de la défaillance et de son impact sur la charge de travail dans son ensemble.

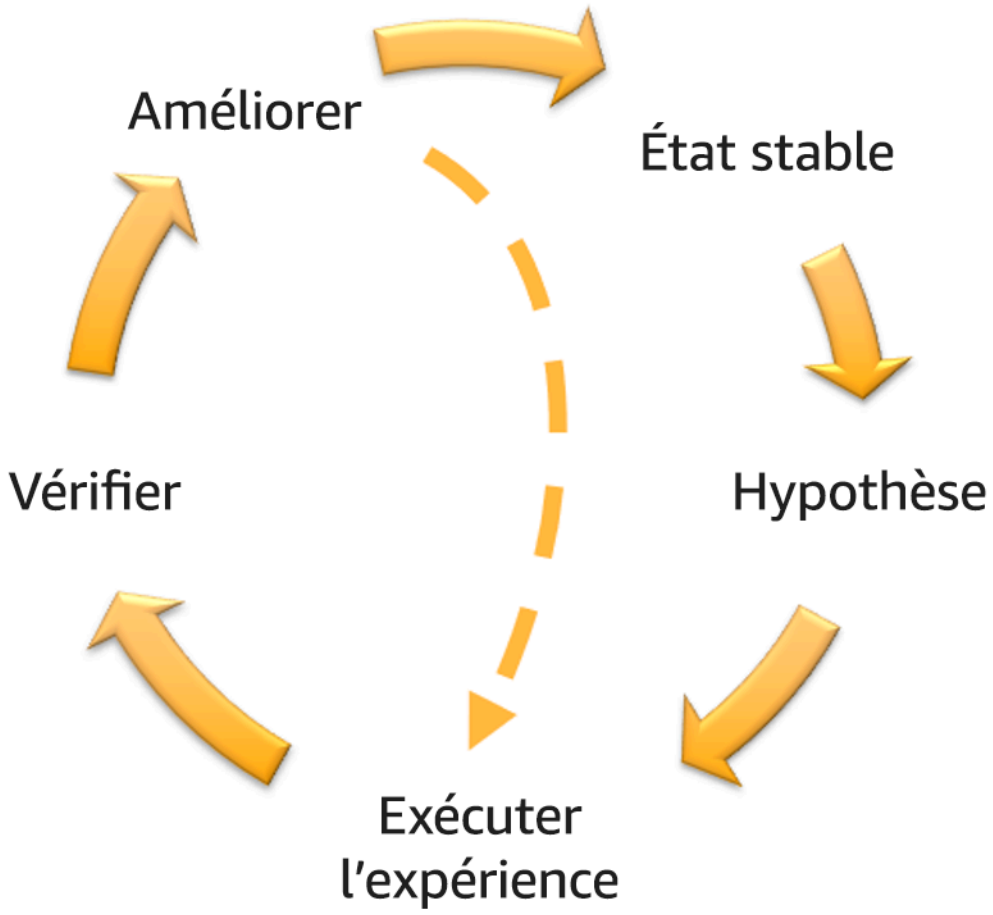
   Lors de la prise en compte de la fréquence d'une défaillance donnée, analysez les données passées de cette charge de travail, le cas échéant. Si aucune donnée passée n'est disponible, utilisez les données des autres charges de travail s'exécutant dans un environnement semblable.

   Lors de la prise en compte de l'impact d'une défaillance donnée, souvenez-vous qu'en général plus le champ de la défaillance est large, plus grand est l'impact. Tenez compte également de la conception de la charge de travail et de son objectif. Par exemple, la capacité à accéder aux magasins de données sources est essentielle pour une charge de travail effectuant des transformations et de l'analyse de données. Dans ce cas, vous donnerez la priorité aux expériences liées aux défaillances d'accès ainsi qu'aux accès limités et à l'insertion de la latence.

   Les analyses post-incident constituent une excellente source de données pour comprendre à la fois la fréquence et l'impact des modes de défaillance.

   Utilisez la priorité attribuée pour déterminer les défaillances à expérimenter en premier lieu, puis l'ordre dans lequel développer de nouvelles expériences d'injection de pannes.

3. Suivre le volant d'inertie de l'ingénierie du chaos et de la résilience continue figurant sur la figure suivante pour chaque expérience réalisée.



Volant d'inertie de l'ingénierie du chaos et de la résilience continue réalisé grâce à la méthode scientifique d'Adrian Hornsby.

a. Définir l'état stable comme le résultat mesurable d'une charge de travail qui indique un comportement normal.

   Votre charge de travail présente un état stable si elle fonctionne de manière fiable et comme prévu. Par conséquent, confirmez que votre charge de travail est saine avant de définir un état stable. L'état stable ne signifie pas forcément sans impact pour la charge de travail en cas de défaillance, car un certain pourcentage des défaillances n'excède pas des limites supportables. L'état stable constitue le repère que vous observerez pendant l'expérience, qui mettra en évidence des anomalies si votre hypothèse formulée dans l'étape suivante ne donne pas les résultats escomptés.

Par exemple, un état stable d'un système de paiements peut être défini comme le traitement de 300 TPS avec un taux de réussite de 99 % et un temps de transmission aller-retour de 500 ms.

b. Formuler une hypothèse sur la façon dont la charge de travail réagira à la défaillance.

Une bonne hypothèse repose sur la façon dont la charge de travail est destinée à réduire la défaillance pour maintenir l'état stable. L'hypothèse indique que vu qu'il s'agit d'une défaillance d'un type particulier, le système ou la charge de travail maintiendra un état stable, car la charge de travail a été conçue avec une atténuation des risques spécifique. Le type particulier de défaillance et d'atténuation des risques doit être spécifié dans l'hypothèse.

Le modèle suivant peut être utilisé pour l'hypothèse (mais une autre formulation est aussi acceptable) :

> **Note**
>
> En cas de *`panne spécifique`*, le *`nom de la charge de travail décrira les contrôles d'atténuation`* visant à maintenir l'*`impact des métriques commerciales ou techniques`*.

Par exemple :

- Si 20 % des nœuds du node-group Amazon EKS sont supprimés, l'API Transaction Create continue de répondre au 99e centile des demandes en moins de 100 ms (état stable). Les nœuds Amazon EKS seront opérationnels dans les cinq minutes, et les pods seront planifiés et traiteront le trafic huit minutes après le début de l'expérience. Les alertes se déclencheront sous trois minutes.
- En cas de défaillance d'une seule instance Amazon EC2, la surveillance de l'état Elastic Load Balancing du système de commandes permet à Elastic Load Balancing d'envoyer uniquement des demandes aux instances saines restantes, tandis qu'Amazon EC2 Auto Scaling remplace l'instance en échec, tout en maintenant une augmentation des erreurs (5xx) côté serveur (état stable) inférieure à 0,01 %.
- Si l'instance de base de données Amazon RDS principale échoue, la charge de travail de collecte des données Chaîne d'approvisionnement basculera et se connectera à l'instance de base de données Amazon RDS de secours pour maintenir les erreurs de lecture ou d'écriture de base de données (état stable) inférieures à 1 minute.

c. Exécuter l'expérience en injectant la défaillance.

Une expérience doit par défaut être sécurisée et tolérée par la charge de travail. Si vous savez que la charge de travail va échouer, n'exécutez pas l'expérience. L'ingénierie du chaos doit être utilisée pour rechercher les risques connus ou inconnus. Les risques connus sont des choses dont vous êtes conscient mais que vous ne comprenez pas complètement, et les risques inconnus sont des choses dont vous n'êtes pas conscient et que vous ne comprenez pas complètement. Exécuter une expérience sur une charge de travail que vous savez défaillante ne vous apportera rien de plus. Votre expérience doit être soigneusement préparée, disposer d'un champ d'impact défini et fournir un mécanisme de protection pouvant être appliqué en cas de perturbations inattendues. Si votre vérification préalable indique que votre charge de travail doit résister à l'expérience, exécutez cette dernière. Il existe plusieurs moyens d'injecter les défaillances. Pour les charges de travail sur AWS, AWS FIS fournit de nombreuses simulations de pannes prédéfinies appelées actions. Vous pouvez également définir des actions personnalisées qui s'exécutent dans AWS FIS à l'aide des documents AWS Systems Manager.

Nous déconseillons l'utilisation de scripts personnalisés pour les expériences de chaos, sauf si ces derniers sont capables de comprendre l'état actuel de la charge de travail, d'émettre des journaux, de fournir des mécanismes de protection pour annuler une expérience et des conditions d'arrêt dans la mesure du possible.

Un framework ou des outils efficaces capables de prendre en charge l'ingénierie du chaos doivent suivre l'état actuel d'une expérience, émettre des journaux et fournir des mécanismes de protection pour prendre en charge l'exécution contrôlée d'une expérience. Commencez par un service établi comme AWS FIS qui vous permet d'exécuter des expériences avec un champ clairement défini et des mécanismes de sécurité capables de protéger l'expérience en cas de perturbations inattendues. Pour en savoir plus sur une plus grande variété d'expériences utilisant AWS FIS, consultez également l'atelier Applications résilientes et Well-Architected avec l'ingénierie du chaos. AWS Resilience Hub analysera votre charge de travail et créera des expériences que vous pourrez choisir d'implémenter et d'exécuter dans AWS FIS.

> **Note**
>
> Pour chaque expérience, vous devez bien comprendre le champ et son impact. Nous recommandons que les défaillances soient d'abord simulées sur un environnement hors production avant d'être exécutées en production.

Les expériences doivent être menées en production sous une charge réelle à l'aide de déploiements Canary qui permettent de déployer à la fois un système de contrôle et un déploiement de système expérimental, dans la mesure du possible. L'exécution d'expériences pendant les heures creuses est une bonne pratique pour réduire l'impact potentiel de la première expérience en production. De plus, si l'utilisation du trafic client réel s'avère trop risquée, vous pouvez exécuter des expériences à l'aide du trafic synthétique sur l'infrastructure de production pour des déploiements de système de contrôles et d'expériences. Lorsqu'une exécution en production n'est pas possible, exécutez les expériences dans des environnements de pré-production aussi proches que possible de la production.

Vous devez définir des barrières de protection pour veiller à ce que l'expérience n'impacte pas le trafic de la production ou d'autres systèmes au-delà des limites acceptables. Définissez des conditions d'arrêt pour stopper une expérience si elle atteint le seuil d'une métrique de barrière de protection défini par vos soins. Ces conditions doivent inclure les métriques de l'état stable de la charge de travail, ainsi que celles sur les composants dans lesquels vous injectez la défaillance. Un moniteur synthétique (également appelée un utilisateur canary) est une métrique que vous devez généralement inclure en tant que proxy utilisateur. Les conditions d'arrêt de AWS FIS sont prises en charge dans le cadre d'un modèle de test, autorisant jusqu'à cinq conditions d'arrêt par modèle.

L'un des principes de l'ingénierie du chaos est de minimiser le champ de l'expérience et son impact :

Bien qu'un impact négatif à court terme soit autorisé, l'ingénieur du chaos a la responsabilité et l'obligation de minimiser et de maîtriser les conséquences des expériences.

Pour vérifier le champ et l'impact potentiel, vous pouvez dans un premier temps exécuter l'expérience dans un environnement hors production, en vérifiant que les seuils des conditions d'arrêt s'activent comme prévu pendant l'expérience et que l'observabilité est en place pour détecter une exception, plutôt que d'exécuter l'expérience directement en production.

Lorsque vous exécutez des expériences d'injection de pannes, vérifiez que toutes les parties responsables sont bien informées. Communiquez avec les équipes appropriées, telles que les équipes en charge des opérations, les équipes chargées de la fiabilité du service et le service client pour leur indiquer quand les expériences seront exécutées et à quoi ils doivent s'attendre. Donnez à ces équipes les outils de communication nécessaires pour informer les personnes en charge de l'exécution de l'expérience si elles constatent des effets négatifs.

Vous devez restaurer la charge de travail et ses systèmes sous-jacents dans leur état fonctionnel et connu d'origine. En général, la conception résiliente de la charge de travail lui permet de s'auto-réparer. Cependant, certaines conceptions défaillantes ou échecs d'expériences peuvent laisser votre charge de travail dans un état d'échec inattendu. À la fin de l'expérience, vous devez en être conscient et restaurer la charge de travail et les systèmes. Avec AWS FIS, vous pouvez définir une configuration de barrière de protection (également appelée post action) dans les paramètres d'action. Une post action restaure la cible dans l'état dans lequel elle se trouvait avant l'exécution de l'action. Qu'elles soient automatisées (comme lorsque vous utilisez AWS FIS) ou manuelles, ces post actions doivent faire partie d'un playbook décrivant la façon de détecter et de gérer les échecs.

d. Vérifier l'hypothèse.

   Principes de l'ingénierie du chaos donne des conseils sur la façon de vérifier l'état stable de votre charge de travail :

   Concentrez-vous sur le résultat mesurable d'un système, plutôt que sur les attributs internes du système. Les mesures de ce résultat sur une courte période de temps constituent un proxy pour l'état stable du système. Le débit général du système, les taux d'erreur et les centiles de latence peuvent tous être des métriques d'intérêt représentant un comportement d'état stable. En se focalisant sur les modèles de comportement systémique pendant les expériences, l'ingénierie du chaos vérifie que le système fonctionne, au lieu d'essayer de confirmer qu'il fonctionne.

   Dans nos deux exemples précédents, nous incluons la métrique de l'état stable inférieure à 0,01 % d'augmentation des erreurs (5xx) côté serveur et la métrique inférieure à 1 minute d'erreurs de lecture ou d'écriture de base de données.

   Les erreurs 5xx constituent une bonne métrique, car elles sont une conséquence du mode de défaillance dont le client de la charge de travail fera l'expérience directement. La mesure des erreurs de base de données est correcte en tant que conséquence directe de la défaillance, mais doit être également complétée par une mesure d'impact, telle que les échecs de demandes client ou les erreurs remontées. Par ailleurs, incluez une surveillance synthétique (également appelée utilisateur canary) sur n'importe quelle API ou URI directement accessible par le client de votre charge de travail.

e. Améliorer la conception de la charge de travail à des fins de résilience.

   Si l'état stable n'a pas été maintenu, enquêtez sur les moyens d'améliorer la conception de la charge de travail afin de réduire la défaillance, tout en appliquant les bonnes pratiques

du pilier AWS Well-Architected Reliability. Vous trouverez des conseils et des ressources supplémentaires dans la AWS Builder's Library, qui contient des articles sur la manière d'améliorer vos surveillances de l'état ou d'utiliser des nouvelles tentatives avec retard dans le code de votre application, entre autres.

Une fois ces changements implémentés, exécutez de nouveau l'expérience (illustrée par la ligne pointillée dans le volant d'inertie de l'ingénierie du chaos) pour déterminer son efficacité. Si l'étape de vérification indique que l'hypothèse est vraie, alors la charge de travail sera en état stable et le cycle continuera.

4. Exécuter régulièrement des expériences.

   Une expérience de chaos est un cycle, et les expériences doivent être exécutées régulièrement dans le cadre de l'ingénierie du chaos. Lorsqu'une charge de travail correspond à l'hypothèse d'une expérience, cette dernière doit être automatisée pour s'exécuter en continu en tant que test de régression de votre pipeline CI/CD. Pour savoir comment procéder, consultez ce blog sur la façon de mener des expériences AWS FIS à l'aide d'AWS CodePipeline. Ce laboratoire sur les expériences AWS FIS récurrentes dans un pipeline CI/CD vous permet de travailler sur le terrain.

   Les expériences d'injection de pannes font également partie des tests de simulation de pannes (consultez REL12-BP05 Organiser régulièrement des tests de simulation de panne). Les tests de simulation de pannes simulent une défaillance ou un événement pour vérifier les systèmes, les processus et la réponse de l'équipe. L'objectif est d'effectuer les actions que l'équipe effectuerait si un événement exceptionnel se produisait.

5. Enregistrer et stocker les résultats des expériences.

   Les résultats des expériences d'injection de pannes doivent être enregistrés et conservés. Incluez toutes les données nécessaires (telles que l'heure, la charge de travail et les conditions) afin de pouvoir analyser ultérieurement les résultats de l'expérience et les tendances. Les exemples de résultats peuvent inclure des captures d'écran des tableaux de bord, des fichiers CSV de la base de données de votre/vos métriques ou un registre manuscrit des événements et observations pendant l'expérience. La journalisation des expériences avec AWS FIS peut faire partie de cette capture de données.

Ressources

Bonnes pratiques associées:

- REL08-BP03 Intégrer les tests de résilience dans le cadre de votre déploiement

- REL13-BP03 Effectuer un test de validation de la mise en œuvre de la reprise après sinistre

Documents connexes:

- Qu’est-ce qu AWS Fault Injection Service?
- Qu'est-ce que AWS Resilience Hub?
- Principes de l’ingénierie du chaos
- Ingénierie du chaos : préparation de votre première expérience
- Ingénierie de résilience : apprendre à intégrer les pannes
- Témoignages d’utilisation de l’ingénierie du chaos
- Éviter les solutions de secours dans les systèmes distribués
- Déploiement canary pour des expériences de chaos

Vidéos connexes :

- AWS re:Invent 2020: Testing resiliency using chaos engineering (ARC316)
- AWS re:Invent 2019: Improving resiliency with chaos engineering (DOP309-R1)
- AWS re:Invent 2019: Performing chaos engineering in a serverless world (CMY301)

Outils associés:

- AWS Fault Injection Service
- AWS Marketplace: Gremlin Chaos Engineering Platform
- Chaos Toolkit
- Chaos Mesh
- Litmus

REL12-BP05 Organiser régulièrement des tests de simulation de panne

Organisez des tests de simulation de panne pour tester régulièrement vos procédures de réponse aux événements et aux déficiences ayant un impact sur la charge de travail. Impliquez les mêmes équipes qui seraient chargées de traiter les scénarios de production. Ces exercices permettent de mettre en œuvre des mesures visant à prévenir l’impact des événements de production sur les utilisateurs. Lorsque vous mettez en pratique vos procédures de réponse dans des conditions

réalistes, vous pouvez identifier et corriger toute lacune ou faiblesse avant l'avènement d'un événement réel.

Les tests de simulation de panne simulent des événements dans des environnements de type production pour tester les systèmes, les processus et la réponse de votre équipe. L'objectif est d'effectuer les mêmes actions que l'équipe effectuerait si l'événement se produisait réellement. Ces exercices vous aident à comprendre où apporter des améliorations et comment développer une expérience de gestion des événements et des déficiences au sein de votre organisation. Ces exercices doivent être effectués régulièrement afin que votre équipe développe des automatismes pour mieux réagir.

Les tests de simulation de panne préparent les équipes à gérer les événements de production en toute confiance. Les équipes expérimentées sont plus à même de détecter différents scénarios et d'y réagir rapidement. Cela se traduit par une amélioration significative de l'état de préparation et de la posture de résilience.

Résultat escompté : vous planifiez et effectuez régulièrement des tests de simulation sur la résilience. Ces tests de simulation de panne sont considérés comme un élément normal et attendu de l'activité de l'entreprise. Votre organisation a développé une culture de préparation et lorsque des problèmes de production surviennent, vos équipes sont bien préparées pour réagir promptement, résoudre efficacement les problèmes et atténuer leur impact sur les clients.

Anti-modèles courants :

- Vous documentez vos procédures sans jamais vous exercer à les appliquer.
- Vous excluez les décideurs d'entreprise des exercices de test.
- Vous organisez des tests de simulation de panne, mais vous n'informez pas toutes les parties prenantes concernées.
- Vous vous concentrez uniquement sur les défaillances techniques, mais vous n'impliquez pas les parties prenantes de l'entreprise.
- Vous n'incorporez pas les leçons apprises lors des tests de simulation de panne dans vos processus de reprise.
- Vous blâmez les équipes pour les échecs et les bogues.

Avantages liés au respect de cette bonne pratique :

- Amélioration des compétences en matière de réponse : lors des tests de simulation de panne, les équipes s'exercent à réaliser leurs tâches et testent leurs mécanismes de communication,

ce qui leur permet de réagir de façon plus coordonnée et efficace dans le cadre de situations de production.

- Identification et traitement des dépendances : les environnements complexes impliquent souvent des dépendances complexes entre différents systèmes, services et composants. Les tests de simulation de panne peuvent vous aider à identifier et à traiter ces dépendances, ainsi qu'à vérifier que vos systèmes et services critiques sont correctement couverts par vos procédures de dossier d¦exploitation et peuvent être augmentés verticalement ou récupérés en temps opportun.
- Promotion d'une culture de résilience : les tests de simulation de panne peuvent aider à développer un état d'esprit de résilience au sein d'une organisation. Lorsqu'ils impliquent des parties prenantes et des équipes interfonctionnelles, ces exercices favorisent la prise de conscience, la collaboration et une compréhension commune de l'importance de la résilience dans l'ensemble de l'organisation.
- Amélioration et adaptation continues : des tests de simulation de panne réguliers vous aident à évaluer en permanence vos stratégies de résilience et à les adapter, afin de les maintenir pertinentes et efficaces face à des circonstances changeantes.
- Renforcement de la confiance dans le système : des tests de simulation de panne réussis peuvent vous aider à renforcer la confiance dans la capacité du système à résister aux perturbations et à s'en remettre.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : moyen

Directives d'implémentation

Une fois que vous avez conçu et mis en œuvre les mesures de résilience nécessaires, organisez des tests de simulation de panne pour confirmer que tout fonctionne comme prévu en production. Les tests de simulation de panne, en particulier la première fois, doivent impliquer tous les membres de l'équipe, et l'ensemble des parties prenantes et des participants doivent être informés à l'avance de la date, de l'heure et des scénarios simulés.

Pendant les tests de simulation de panne, les équipes impliquées simulent divers événements et scénarios potentiels conformément aux procédures prescrites. Les participants surveillent de près et évaluent l'impact de ces événements simulés. Si le système fonctionne comme prévu, les mécanismes automatisés de détection, de mise à l'échelle et de réparation automatique devraient s'activer et n'entraîner que peu ou pas d'impact sur les utilisateurs. Si l'équipe constate un impact négatif, elle doit annuler le test et résoudre les problèmes identifiés, soit par des moyens automatisés, soit par une intervention manuelle documentée dans les dossiers d¦exploitation applicables.

Pour améliorer continuellement la résilience, il est essentiel de documenter et d'incorporer les leçons apprises. Ce processus constitue une boucle de rétroaction qui capture systématiquement les informations exploitables recueillies pendant les tests de simulation de panne et les utilise pour améliorer les systèmes, les processus et les compétences des équipes.

Pour vous aider à reproduire des scénarios réels dans lesquels des services ou des composants du système peuvent tomber en panne de façon inattendue, injectez des défauts simulés dans un exercice de test de simulation. Les équipes peuvent tester la résilience et la tolérance aux pannes de leurs systèmes et simuler leurs processus de réponse aux incidents et de reprise dans un environnement contrôlé.

Dans AWS, vos tests de simulation de panne peuvent être réalisés avec des réplicas de votre environnement de production en utilisant une infrastructure en tant que code. Ce processus vous permert d'effectuer des tests dans un environnement sûr qui ressemble étroitement à votre environnement de production. Envisagez d'utiliser AWS Fault Injection Service pour créer différents scénarios de panne. Utilisez des services tels qu'Amazon CloudWatch et AWS X-Ray pour surveiller le comportement du système pendant les tests de simulation de panne. Utilisez AWS Systems Manager pour gérer et exécuter les playbooks, et AWS Step Functions pour orchestrer les flux de travail récurrents des tests de simulation de panne.

Étapes d'implémentation

- Établissez un programme de tests de simulation de panne : élaborez un programme structuré qui définit la fréquence, la portée et les objectifs des tests de simulation de panne. Impliquez les principales parties prenantes et les experts du domaine concerné dans la planification et l'exécution de ces exercices.
- Préparez les tests de simulation de panne :
  1. Identifiez les principaux services essentiels à l'entreprise qui seront au centre des tests de simulation de panne. Cataloguez et cartographiez les personnes, les processus et les technologies qui prennent en charge ces services.
  2. Définissez l'ordre du jour des tests de simulation de panne et préparez les équipes impliquées à participer à l'événement. Préparez vos services d'automatisation pour simuler les scénarios planifiés et exécuter les processus de récupération appropriés. Les services AWS tels que AWS Fault Injection Service, AWS Step Functions et AWS Systems Manager peuvent vous aider à automatiser divers aspects des tests de simulation de panne, tels que l'injection des défauts et le lancement des actions de récupération.

- Exécutez votre simulation : dans le cadre des tests de simulation de panne, exécutez le scénario planifié. Observez et documentez la façon dont les personnes, les processus et les technologies réagissent à l'événement simulé.
- Réalisez le bilan de l'exercice : après les tests de simulation de panne, organisez une séance rétrospective pour passer en revue les enseignements tirés. Identifiez les domaines d'amélioration et les actions nécessaires pour améliorer la résilience opérationnelle. Consignez vos résultats et effectuez le suivi des modifications nécessaires pour améliorer vos stratégies de résilience et votre préparation aux travaux à entreprendre.

Ressources

Bonnes pratiques associées :

- REL12-BP01 Utiliser des playbooks pour enquêter sur les causes des défaillances
- REL12-BP04 Tester la résilience à l'aide de l'ingénierie du chaos
- OPS04-BP01 Identification des indicateurs de rendement clés
- OPS07-BP03 Utilisation de runbooks pour effectuer des procédures
- OPS10-BP01 Utilisation d'un processus pour la gestion des événements, des incidents et des problèmes

Documents connexes :

- Qu'est-ce qu'AWS GameDay ?

Vidéos connexes:

- AWS re:Invent 2023 - Practice like you play: How Amazon scales resilience to new heights

Exemples connexes :

- AWS Atelier – Surmonter la tempête : déclenchement d'un chaos contrôlé pour systèmes résilients
- Élaboration de tests de simulation de panne en vue de soutenir la résilience opérationnelle

Planification de la reprise après sinistre

La mise en place de sauvegardes et de composants de charge de travail redondants constitue le début de votre stratégie de DR. L'objectif de délai de reprise (RTO) et l'objectif de point de reprise (RPO) sont vos objectifs pour la restauration de votre charge de travail. Définissez-les en fonction des besoins de l'entreprise. Mettez en œuvre une stratégie pour atteindre ces objectifs, en particulier en tenant compte de l'emplacement et de la fonction des données et des ressources de charge de travail. La probabilité d'une perturbation et le coût de la reprise sont également des facteurs clés qui permettent de déterminer la valeur opérationnelle de la reprise après sinistre d'une charge de travail.

La disponibilité et la reprise après sinistre s'appuient sur les mêmes bonnes pratiques, telles que la surveillance des pannes, le déploiement sur plusieurs sites et le basculement automatique. Cependant, la disponibilité se concentre sur les composants de la charge de travail, tandis que la reprise après sinistre se concentre sur des copies distinctes de l'ensemble de la charge de travail. La reprise après sinistre a des objectifs différents de la disponibilité, car elle se concentre sur le temps de récupération après un sinistre.

Bonnes pratiques

- REL13-BP01 Définir les objectifs de reprise pour les temps d'arrêt et les pertes de données
- REL13-BP02 Utiliser des stratégies de reprise définies pour répondre aux objectifs de reprise
- REL13-BP03 Effectuer un test de validation de la mise en œuvre de la reprise après sinistre
- REL13-BP04 Gérer l'écart de configuration au niveau du site ou de la région de reprise après sinistre
- REL13-BP05 Automatiser la reprise

REL13-BP01 Définir les objectifs de reprise pour les temps d'arrêt et les pertes de données

Les défaillances peuvent avoir un impact sur votre activité de plusieurs manières. Tout d'abord, les défaillances peuvent entraîner une interruption de service (durée d'indisponibilité). Ensuite, les défaillances peuvent entraîner la perte, l'incohérence ou l'obsolescence des données. Pour déterminer la manière de réagir et de récupérer après une défaillance, définissez un objectif de délai de reprise (RTO) et un objectif de point de reprise (RPO) pour chaque charge de travail. L'objectif de délai de reprise (RTO) est le délai maximal acceptable entre l'interruption du service et son rétablissement. L'objectif de point de reprise (RPO) est le temps maximal acceptable après le dernier point de récupération des données.

Résultat escompté : chaque charge de travail dispose d'un RTO et d'un RPO basés sur des considérations techniques et l'impact sur l'activité.

Anti-modèles courants:

- Vous n'avez pas défini d'objectifs de récupération.
- Vous sélectionnez des objectifs de récupération arbitraires.
- Vous sélectionnez des objectifs de récupération trop souples qui ne répondent pas aux objectifs de l'entreprise.
- Vous n'avez pas évalué l'impact de la durée d'indisponibilité et de la perte de données.
- Vous sélectionnez des objectifs de récupération non réalistes pour la configuration de votre charge de travail, tels qu'un délai de récupération nul ou une perte de données nulle, qui ne sont pas réalisables.
- Vous sélectionnez des objectifs de récupération plus rigoureux que les objectifs métier réels. Cela entraîne une mise en œuvre de la récupération plus coûteuse et plus compliquée que ce dont la charge de travail a besoin.
- Vous sélectionnez des objectifs de récupération incompatibles avec ceux d'une charge de travail dépendante.
- Vous ne tenez pas compte des exigences réglementaires et de conformité.

Avantages liés au respect de cette bonne pratique : lorsque vous définissez des RTO et des RPO pour vos charges de travail, vous définissez des objectifs de récupération clairs et mesurables en fonction des besoins de votre entreprise. Une fois ces objectifs définis, vous pouvez créer des plans de reprise après sinistre (DR) spécialement conçus pour les atteindre.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : élevé

Directives d'implémentation

Créez une matrice ou une fiche qui vous aidera à planifier la reprise après sinistre. Dans cette matrice, créez différentes catégories ou niveaux de charge de travail en fonction de leur impact sur l'activité (critique, élevé, moyen ou faible) et des RTO et RPO associés à cibler pour chacun d'entre eux. La matrice suivante fournit un exemple (notez que vos valeurs RTO et RPO peuvent différer) que vous pouvez suivre :

Matrice de reprise après sinistre						
		Objectif de point de reprise				
		Moins de 1 minute	Moins de 1 heure	Moins de 6 heures	Moins de 1 jour	+ de 1 jour
Durée maximale d'interruption	Moins de 10 minutes	Critique	Critique	Débit	Moyenne entreprise	Moyenne entreprise
	Moins de 2 heures	Critique	Débit	Moyenne entreprise	Moyenne entreprise	Faible
	Moins de 8 heures	Débit	Moyenne entreprise	Moyenne entreprise	Faible	Faible
	Moins de 24 heures	Moyenne entreprise	Moyenne entreprise	Faible	Faible	Faible
	+ de 24 heures	Moyenne entreprise	Faible	Faible	Faible	Faible

Exemple de matrice de reprise après sinistre

Pour chaque charge de travail, vous devez étudier et comprendre l'impact de la durée d'indisponibilité et de la perte de données sur votre activité. Cet impact augmente généralement avec la durée d'indisponibilité et la perte de données, mais sa forme peut varier en fonction du type de charge de travail. Par exemple, une durée d'indisponibilité d'une heure peut avoir un impact faible, mais après cela, l'impact peut croître rapidement. L'impact peut prendre de nombreuses formes, y compris la forme d'un impact financier (tel qu'une perte de chiffre d'affaires), d'un impact sur la réputation (y compris la perte de confiance des clients), d'un impact opérationnel (comme une paie manquée ou une baisse de productivité) et d'un risque réglementaire. Une fois terminé, attribuez la charge de travail au niveau approprié.

Lorsque vous analysez l'impact d'une panne, posez-vous les questions suivantes :

1. Quelle est la durée maximale d'indisponibilité de la charge de travail avant qu'elle n'ait un impact inacceptable sur l'activité ?
2. Quelle est l'ampleur et la nature de l'impact d'une interruption de la charge de travail sur l'entreprise ? Tenez compte de tous les types d'impact, notamment financier, de réputation, opérationnel et réglementaire.
3. Quelle est la quantité maximale de données pouvant être perdues ou irrécupérables avant que l'activité ne subisse un impact inacceptable ?
4. Les données perdues peuvent-elles être recréées à partir d'autres sources (également appelées données dérivées) ? Si tel est le cas, considérez également les RPO de toutes les données sources utilisées pour recréer les données de la charge de travail.

5. Quels sont les objectifs de récupération et les attentes de disponibilité des charges de travail dont celle-ci dépend (en aval) ? Vos objectifs en matière de charge de travail doivent être réalisables compte tenu des capacités de récupération de ses dépendances en aval. Envisagez des solutions de contournement ou d'atténuation des dépendances en aval susceptibles d'améliorer la capacité de récupération de cette charge de travail.
6. Quels sont les objectifs de récupération et les attentes de disponibilité des charges de travail qui dépendent de celle-ci (en amont) ? Les objectifs de charge de travail en amont peuvent exiger que cette charge de travail soit dotée de capacités de récupération plus strictes qu'il n'y paraît initialement.
7. Les objectifs de récupération sont-ils différents en fonction du type d'incident ? Par exemple, vous pouvez avoir des RTO et des RPO différents selon que l'incident a un impact sur une zone de disponibilité ou sur une région entière.
8. Vos objectifs de récupération changent-ils au cours de certains événements ou à certaines périodes de l'année ? Par exemple, vous pouvez avoir des RTO et des RPO différents pour les fêtes de fin d'année, lors d'événements sportifs, en périodes de soldes et lors du lancement de nouveaux produits.
9. Comment les objectifs de récupération s'alignent-ils sur la stratégie de reprise après sinistre de votre secteur d'activité et de votre organisation ?
10. Y a-t-il des ramifications juridiques ou contractuelles à prendre en compte ? Par exemple, avez-vous l'obligation contractuelle de fournir un service avec un RTO ou un RPO donné ? Quelles pénalités pourriez-vous encourir en cas de non-respect de cette obligation ?
11. Devez-vous maintenir l'intégrité des données pour répondre aux exigences réglementaires ou de conformité ?

La fiche suivante peut vous aider à évaluer chaque charge de travail. Vous pouvez modifier cette fiche en fonction de vos besoins spécifiques, par exemple en ajoutant des questions supplémentaires.

Étape 2 : Questions principales	S'applique à la charge de travail ?	RPO de la charge de travail	RPO de la charge de travail	Ajustement de RTO.	Ajustement de RPO.	Instructions
[1] durée maximale pendant laquelle la charge de travail peut être inactive						mesuré en temps depuis le début de la panne jusqu'à la récupération
[2] quantité maximale de données pouvant être perdues						mesuré dans le temps depuis le dernier jeu de données restaurable
[3a] dépendances en amont						saisissez les objectifs de récupération en amont les plus stricts
[3b] dépendances en aval						saisissez les objectifs de récupération en aval les moins stricts
[3a] dépendances en amont rapprochées						Si la valeur en amont est inférieure aux valeurs actuelles et la valeur en aval supérieure,
[3b] dépendances en aval rapprochées						manipulez les dépendances pour les rapprocher et entrez les valeurs rapprochées ici
[3] dépendances						réduisez les valeurs pour répondre aux dépendances en amont ou augmentez-les selon les fonctionnalités de dépendance en aval
Étape 2 : Questions supplémentaires						Indiquez si la question s'applique. Dans le cas contraire, ignorez-la
RTO/RPO de base						Transférez les valeurs RTO et RPO d'en haut jusqu'ici
[4] type de panne	[ ]O / [ ]N					Indiquez les objectifs de récupération pour les événements avec les exigences les plus strictes
[5] objectifs temporels spécifiques	[ ]O / [ ]N					Indiquez les objectifs de récupération pour les durées avec les exigences les plus strictes
[6] clients perturbés	[ ]O / [ ]N					Représentez graphiquement les clients impactés en fonction du temps d'arrêt ou de la perte de données. Utilisez ces informations pour saisir le RTO et le RPO maximum autorisés en fonction de l'impact sur le client
[7] impact sur la réputation	[ ]O / [ ]N					Déterminez avec l'entreprise le RTO et le RPO maximum en fonction de l'impact sur la réputation
[8] impact opérationnel	[ ]O / [ ]N					Indiquez un RTO et un RPO maximum en fonction de l'impact opérationnel
[9] alignement organisationnel	[ ]O / [ ]N					Indiquez le RTO et le RPO maximum pour les charges de travail de ce type selon les exigences LOB et organisationnelles
[10] obligations contractuelles	[ ]O / [ ]N					Indiquez un RTO et un RPO maximum en fonction des obligations contractuelles
[11] conformité réglementaire	[ ]O / [ ]N					Indiquez le RTO et le RPO maximum en fonction de la conformité réglementaire applicable
cible basée sur des questions supplémentaires						Prenez la valeur minimale (valeur plus stricte) des Q 11-4 et entrez-la ici
cible ajustée						Si les objectifs de la ligne ci-dessus ne peuvent pas être atteints, collaborez avec les parties prenantes pour assouplir les contraintes et entrez un nouveau minimum ici
RTO/RPO ajusté						Indiquez les valeurs RPO/RTO de base, ou la cible ajustée, selon la valeur la plus basse
Étape 3						
Mapper vers une catégorie ou un niveau prédéfini						Ajustez les deux valeurs vers le bas (méthode plus stricte) pour vous aligner sur le niveau défini le plus proche

Fiche

Étapes d'implémentation

1. Identifiez les parties prenantes et les équipes techniques responsables de chaque charge de travail et interagissez avec elles.
2. Créez des catégories ou des niveaux de criticité pour déterminer l'impact de la charge de travail dans votre organisation. Exemples de catégories : critique, élevé, moyen et faible. Pour chaque catégorie, choisissez un RTO et un RPO qui reflètent les objectifs et les exigences de votre activité.
3. Attribuez l'une des catégories d'impact que vous avez créées à l'étape précédente à chaque charge de travail. Pour déterminer la correspondance d'une charge de travail à une catégorie, tenez compte de l'importance de la charge de travail pour l'activité et de l'impact de son interruption ou d'une perte de données, et utilisez les questions ci-dessus pour vous guider. Cela se traduit par un RTO et un RPO pour chaque charge de travail.
4. Pour chaque charge de travail, tenez compte du RTO et du RPO déterminés à l'étape précédente. Impliquez les équipes stratégiques et techniques chargées de la charge de travail afin de déterminer si les objectifs doivent être ajustés. Par exemple, les parties prenantes de l'entreprise

peuvent déterminer que des objectifs plus stricts sont requis. Par ailleurs, les équipes techniques peuvent décider de la nécessité de modifier les cibles pour les rendre réalisables dans le cadre des contraintes technologiques et des ressources disponibles.

Ressources

Bonnes pratiques associées :

- REL09-BP04 Effectuer une récupération périodique des données pour vérifier l'intégrité et les processus de sauvegarde
- REL12-BP01 Utiliser des playbooks pour enquêter sur les causes des défaillances
- REL13-BP02 Utiliser des stratégies de reprise définies pour répondre aux objectifs de reprise
- REL13-BP03 Effectuer un test de validation de la mise en œuvre de la reprise après sinistre

Documents connexes :

- AWS Blog d'architecture : série sur la reprise après sinistre
- Reprise après sinistre des charges de travail sur AWS : reprise dans le cloud (livre blanc AWS)
- Gestion des politiques de résilience avec AWS Resilience Hub
- Partenaire APN : partenaires pouvant faciliter la reprise après sinistre
- AWS Marketplace: produits pouvant être utilisés pour la reprise après sinistre

Vidéos connexes :

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications
- Disaster Recovery of Workloads on AWS

REL13-BP02 Utiliser des stratégies de reprise définies pour répondre aux objectifs de reprise

Définissez une stratégie de reprise après sinistre qui répond aux objectifs de reprise de votre charge de travail. Choisissez une stratégie telle que : sauvegarde et restauration, mode secours (actif/passif) ou actif/actif.

Résultat escompté : pour chaque charge de travail, il existe une stratégie de reprise après sinistre définie et implémentée qui permet à cette charge de travail d'atteindre les objectifs de reprise. Les stratégies de reprise après sinistre entre les charges de travail utilisent des modèles réutilisables (comme les stratégies décrites précédemment).

Anti-modèles courants :

- Mettre en œuvre des procédures de récupération incohérentes pour les charges de travail avec des objectifs de reprise après sinistre similaires.
- Conserver l'implémentation ad hoc de la stratégie de reprise après sinistre lorsqu'un sinistre se produit.
- Ne pas avoir de plan de reprise après sinistre.
- Être dépendant des opérations du plan de contrôle pendant la récupération.

Avantages liés au respect de cette bonne pratique :

- L'utilisation de stratégies de reprise définies vous permet d'utiliser des outils et des procédures de test courantes.
- L'utilisation de stratégies de reprise définies améliore le partage des connaissances entre les équipes et la mise en œuvre de la reprise après sinistre sur les charges de travail qu'elles possèdent.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : élevé. Sans une stratégie de reprise après sinistre planifiée, mise en œuvre et testée, il est peu probable que vous atteigniez vos objectifs de reprise en cas de sinistre.

Directives d'implémentation

Une stratégie de reprise après sinistre repose sur la capacité à rétablir votre charge de travail sur un site de reprise si votre emplacement principal ne parvient plus à exécuter cette charge de travail. Les objectifs de récupération les plus courants sont le RTO et le RPO, comme indiqué dans REL13-BP01 Définir les objectifs de reprise pour les temps d'arrêt et les pertes de données.

Une stratégie de reprise après sinistre sur plusieurs zones de disponibilité (AZ) au sein d'une seule Région AWS peut vous prémunir contre les événements catastrophiques tels que les incendies, les inondations et les pannes de courant majeures. S'il est nécessaire de mettre en œuvre une protection

contre un événement improbable qui empêcherait votre charge de travail de s'exécuter dans une Région AWS donnée, optez pour une stratégie de reprise après sinistre qui utilise plusieurs régions.

Lors de la conception d'une stratégie de reprise après sinistre dans plusieurs régions, vous devez choisir l'une des approches suivantes. Elles sont classées par ordre croissant de coût et de complexité, et par ordre décroissant de RTO et RPO. La région de restauration fait référence à une région Région AWS autre que la région principale utilisée pour votre charge de travail.

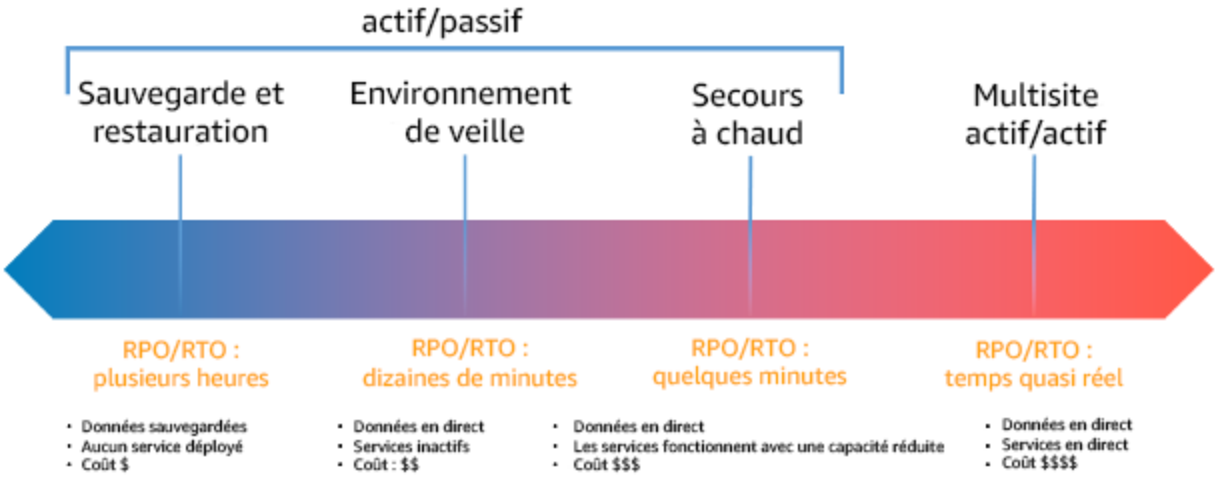


Figure 17 : stratégies de reprise après sinistre

- Sauvegarde et restauration (RPO en heures, RTO de 24 heures maximum) : sauvegardez vos données et applications dans la région de reprise après sinistre. L'utilisation de sauvegardes automatisées ou continues permet une reprise ponctuelle (PITR), ce qui peut réduire le RPO à seulement 5 minutes dans certains cas. En cas de sinistre, vous déployez votre infrastructure (en utilisant l'infrastructure en tant que code pour réduire le RTO), déployez votre code et restaurez les données sauvegardées pour vous remettre d'un sinistre dans la région de reprise.
- Veilleuse (RPO de quelques minutes, RTO de dizaines de minutes) : allouez une copie de votre infrastructure de charge de travail principale dans la région de reprise. Répliquez vos données dans la région de reprise et créez-y des sauvegardes. Les ressources requises pour prendre en charge la réplication et la sauvegarde des données, telles que les bases de données et le stockage d'objets, sont toujours actives. D'autres éléments tels que les serveurs d'applications ou le calcul sans serveur ne sont pas déployés, mais peuvent être créés si nécessaire avec la configuration et le code d'application requis.

- Secours semi-automatique (RPO de quelques secondes, RTO de quelques minutes) : maintenez une version réduite verticalement d'une charge de travail entièrement fonctionnelle qui s'exécute toujours dans la région de reprise. Les systèmes stratégiques sont entièrement dupliqués et sont toujours opérationnels, mais avec une flotte réduite verticalement. Les données sont répliquées dans la région de reprise et y sont hébergées. Lorsque vient le moment de la reprise, le système est rapidement mis à l'échelle pour gérer la charge de production. Plus l'échelle du secours à chaud est élevée, plus la dépendance au RTO et au plan de contrôle est faible. Lorsqu'elle est entièrement mise à l'échelle, on parle de veille permanente.
- Multi-région (multi-site) active-active (RPO proche de zéro, RTO potentiellement nul) : votre charge de travail est déployée et dessert activement le trafic à partir de plusieurs Régions AWS. Cette stratégie vous oblige à synchroniser les données entre les régions. Il est important d'éviter ou de gérer les éventuels conflits causés par des écritures sur le même enregistrement dans deux réplicas régionaux différents, ce qui peut être complexe. La réplication des données est utile pour la synchronisation des données et vous protège contre certains types de sinistres. Toutefois, elle ne vous protège pas contre la corruption ou la destruction des données à moins que votre solution n'inclue également des options de récupération ponctuelle.

> **Note**
>
> La différence entre l'environnement en veille et le secours à chaud est parfois difficile à cerner. Ces deux stratégies incluent un environnement dans votre région de reprise avec des copies des ressources de votre région principale. L'environnement en veille diffère en ce qu'il ne peut pas traiter les demandes sans qu'une action supplémentaire soit entreprise au préalable, tandis que le secours à chaud peut gérer le trafic (à des niveaux de capacité réduits) immédiatement. L'environnement en veille vous oblige à allumer des serveurs, à déployer éventuellement une infrastructure supplémentaire (non essentielle) et à augmenter verticalement, tandis que le secours à chaud nécessite uniquement une augmentation verticale (tout est déjà déployé et en cours d'exécution). Choisissez entre ces options en fonction de vos besoins en matière de RTO et de RPO.
>
> Si le coût est un problème et que vous souhaitez atteindre des objectifs de RPO et RTO similaires à ceux définis dans la stratégie de secours à chaud, vous pouvez envisager des solutions natives du cloud, comme Reprise après sinistre AWS Elastic, qui adoptent l'approche de l'environnement de veille et offrent des objectifs de RPO et RTO améliorés.

Étapes d'implémentation

1. Déterminez une stratégie de reprise après sinistre qui répond aux exigences de récupération pour cette charge de travail.

   Le choix d'une stratégie de reprise après sinistre vise à trouver un juste milieu entre la réduction des temps d'arrêt et de la perte de données (RTO et RPO) et le coût et la complexité liées à la mise en œuvre de cette stratégie. Évitez de mettre en œuvre une stratégie plus stricte que nécessaire, car cela entraînerait des coûts inutiles.

   Par exemple, dans le diagramme suivant, l'entreprise a déterminé son RTO maximal autorisé ainsi que la limite de dépenses possible pour sa stratégie de restauration de service. Compte tenu des objectifs de l'entreprise, les stratégies de reprise après sinistre en veille et secours à chaud satisfont à la fois aux critères de RTO et de coût.

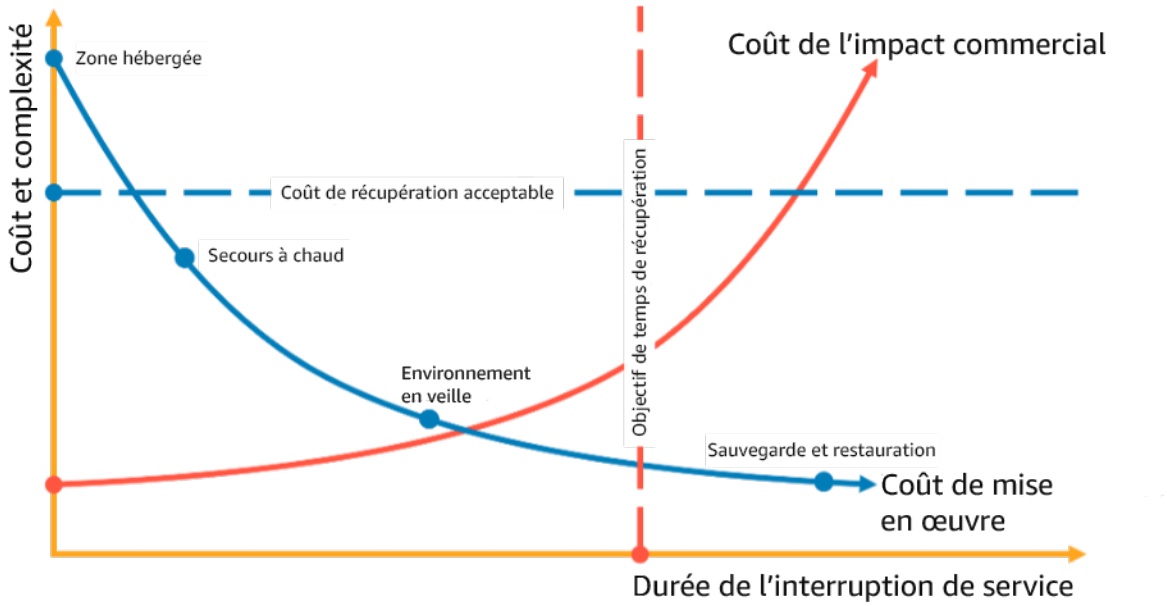


   Figure 18 : choix d'une stratégie de reprise après sinistre basée sur le RTO et le coût

   Pour en savoir plus, consultez Plan de continuité d'activité (BCP).

2. Passez en revue les modèles de mise en œuvre de la stratégie de reprise après sinistre sélectionnée.

   Cette étape consiste à comprendre comment mettre en œuvre la stratégie sélectionnée. Les stratégies reposent sur l'utilisation de Régions AWS comme site principal et site de reprise. Cependant, vous pouvez également choisir d'utiliser des zones de disponibilité dans une seule région comme stratégie de reprise après sinistre, ce qui permet d'exploiter des éléments de plusieurs de ces stratégies.

Dans les étapes suivantes, vous pouvez appliquer la stratégie à votre charge de travail spécifique.

Sauvegarde et restauration

Sauvegarde et restauration est la stratégie la moins complexe à mettre en œuvre, mais nécessite plus de temps et d'efforts pour la restauration de la charge de travail, ce qui entraîne un RTO et un RPO plus élevés. Il est conseillé de toujours faire des sauvegardes de vos données et de les copier sur un autre site (comme une autre Région AWS).

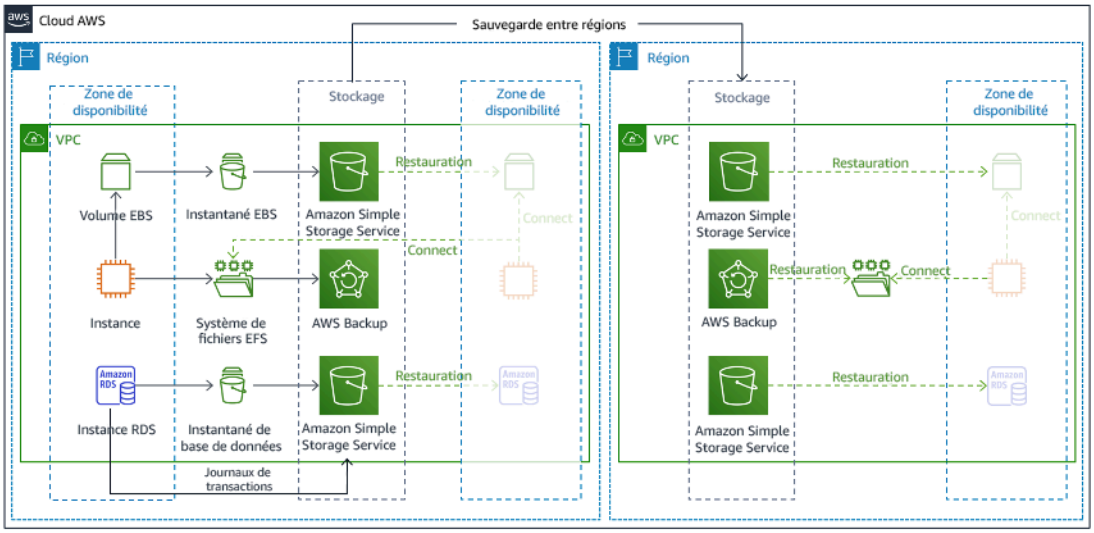


Figure 19 : architecture de sauvegarde et de restauration

Pour en savoir plus sur cette stratégie, consultezArchitecture de reprise après sinistre (DR) sur AWS, partie 2 : sauvegarde et restauration avec récupération rapide.

Veilleuse

L'approche de veilleuse, vous permet de répliquer vos données depuis la région principale vers la région de reprise. Les ressources principales utilisées pour l'infrastructure de charge de travail sont déployées dans la région de reprise, mais des ressources supplémentaires et toutes les dépendances sont toujours nécessaires pour en faire une pile fonctionnelle. Par exemple, dans la figure 20, aucune instance de calcul n'est déployée.

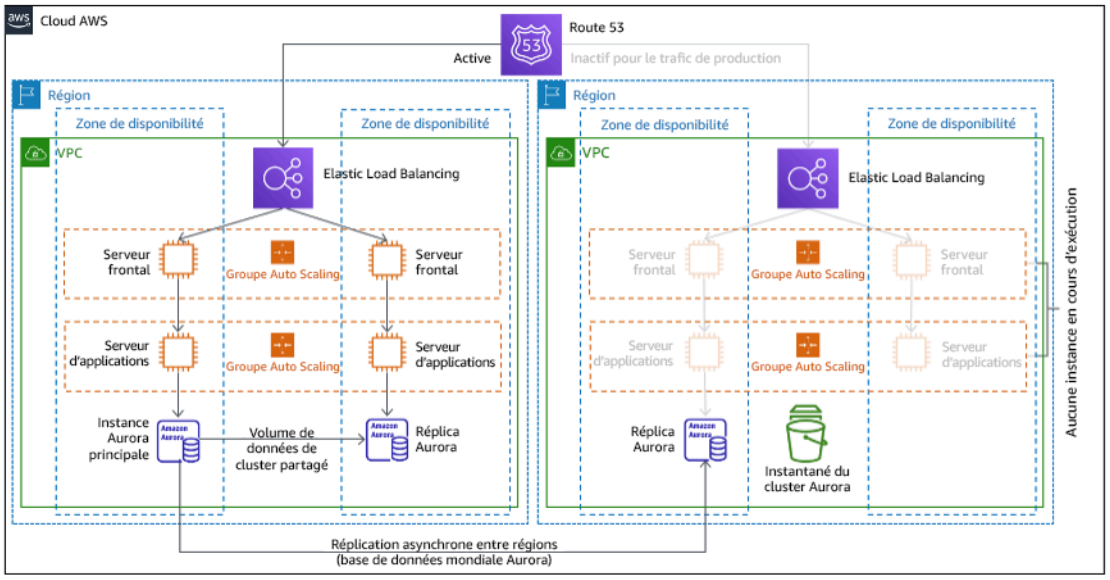


Figure 20 : architecture avec environnement en veille

Pour en savoir plus sur cette stratégie, consultez Architecture de reprise après sinistre sur AWS, partie 3 : environnement en veille et secours à chaud.

Secours semi-automatique

L'approche du secours semi-automatique consiste à s'assurer qu'il existe une copie réduite verticalement, mais entièrement fonctionnelle, de votre environnement de production dans une autre région. Cette approche étend le concept d'environnement en veille et réduit le temps de récupération, car votre charge de travail reste active dans une autre région. Si la région de reprise est déployée à pleine capacité, on parle de veille permanente.

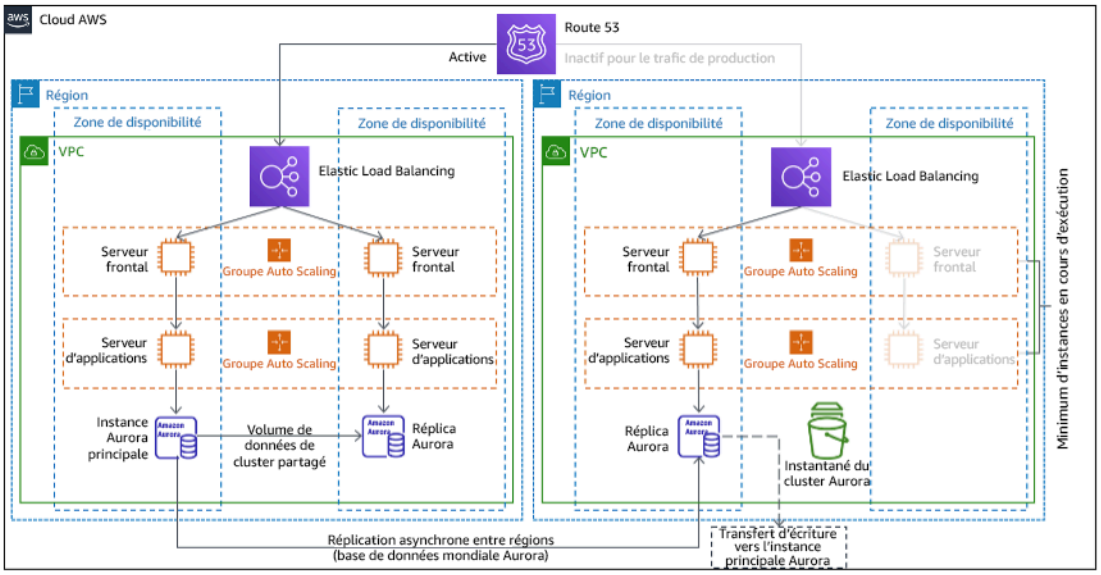


Figure 21 : Architecture de secours à chaud

L'utilisation du secours à chaud ou de l'environnement en veille nécessite une augmentation verticale des ressources dans la région de reprise. Pour vérifier que la capacité est disponible en cas de besoin, envisagez de l'utiliser pour les réservations de capacité pour les instances EC2. Si vous utilisez AWS Lambda, la simultanéité provisionnée peut fournir des environnements d'exécution afin qu'ils soient prêts à répondre immédiatement aux invocations de votre fonction.

Pour plus de détails sur cette stratégie, consultez Architecture de reprise après sinistre sur AWS, partie 3 : environnement en veille et secours à chaud.

Multisite actif/actif

Vous pouvez exécuter votre charge de travail simultanément dans plusieurs régions dans le cadre d'une stratégie multisite active/active. Une stratégie multisite actif/actif dessert le trafic de toutes les régions dans lesquelles il est déployé. Les clients peuvent sélectionner cette stratégie pour des raisons autres que la reprise après sinistre. Elle peut être utilisée pour augmenter la disponibilité ou lors du déploiement d'une charge de travail auprès d'une audience mondiale (pour rapprocher le point de terminaison des utilisateurs et/ou déployer des piles localisées pour l'audience de cette région). En tant que stratégie de reprise après sinistre, si la charge de travail ne peut pas être prise en charge dans l'une des Régions AWS vers lesquelles elle est déployée, cette région est évacuée, et les régions restantes sont utilisées pour assurer la disponibilité. La stratégie de reprise après sinistre multisite actif/actif est la plus complexe sur le plan opérationnel et ne doit être sélectionnée que lorsque les besoins de l'entreprise l'exigent.

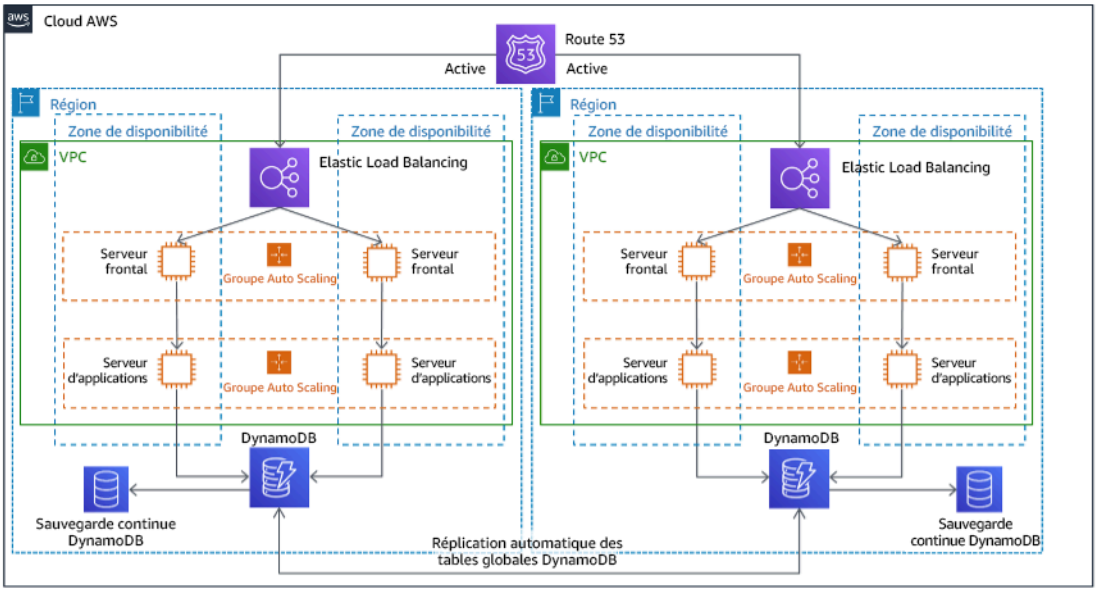


Figure 22 : architecture multisite de type actif/actif

Pour en savoir plus sur cette stratégie, consultez Architecture de reprise après sinistre sur AWS, partie 4 : multisite actif/actif.

Reprise après sinistre AWS Elastic

Si vous envisagez d'adopter une stratégie de veilleuse ou de secours à chaud pour la reprise après sinistre, Reprise après sinistre AWS Elastic peut proposer une autre approche offrant de meilleurs avantages. Elastic Disaster Recovery peut offrir un objectif de RPO et de RTO similaire à celui du mode de secours à chaud, tout en conservant l'approche peu coûteuse de la veilleuse. Elastic Disaster Recovery réplique vos données de votre région principale vers votre région de reprise, en utilisant une protection continue des données pour atteindre un RPO mesuré en secondes et un RTO mesurable en minutes. Seules les ressources nécessaires à la réplication des données sont déployées dans la région de reprise, ce qui permet de limiter les coûts, à l'instar de la stratégie de l'environnement de veille. En cas d'utilisation de Elastic Disaster Recovery, le service coordonne et orchestre la récupération des ressources informatiques lorsqu'elle est initiée dans le cadre d'un basculement ou d'une opération.

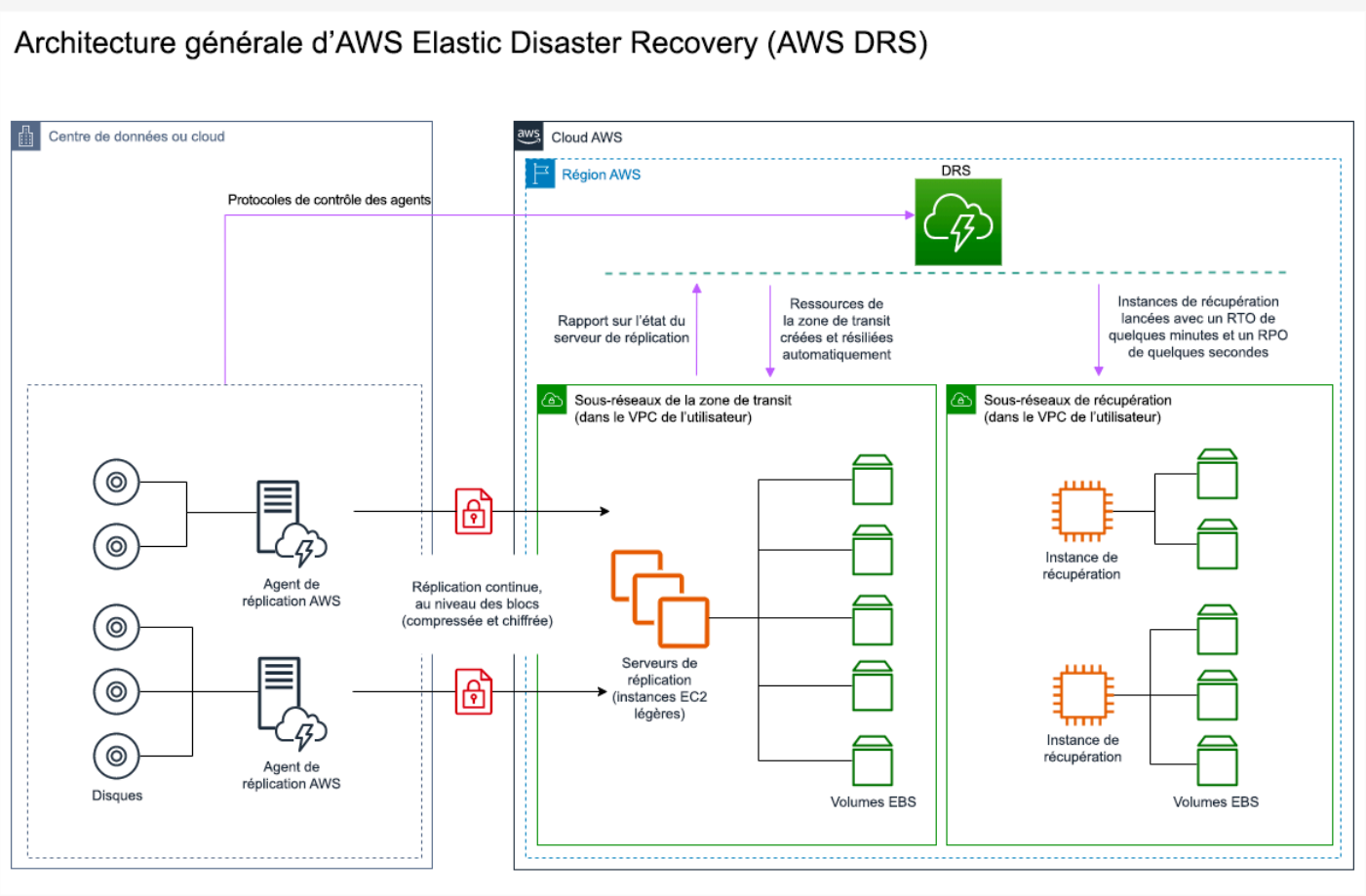


Figure 23 : architecture Reprise après sinistre AWS Elastic

Pratiques supplémentaires de protection des données

Avec toutes les stratégies, vous devez également vous prémunir contre les catastrophes liées aux données. La réplication continue des données vous protège contre certains types de sinistres, mais ne vous protège pas toujours contre la corruption ou la destruction des données, à moins que votre stratégie n'inclue également la gestion des versions des données stockées ou des options de récupération ponctuelle. Vous devez également sauvegarder les données répliquées sur le site de reprise pour créer des sauvegardes ponctuelles en plus des réplicas.

Utilisation de plusieurs zones de disponibilité (AZ) dans une seule Région AWS

Lorsque vous utilisez plusieurs AZ dans une même région, l'implémentation de la reprise après sinistre exploite plusieurs éléments des stratégies ci-dessus. Vous devez d'abord créer une architecture haute disponibilité (HA), en utilisant plusieurs AZ, comme illustré à la figure 23. Cette architecture utilise une approche multisite actif/actif, car les instances Amazon EC2 et Elastic

Load Balancer disposent de ressources déployées dans plusieurs zones de disponibilité, ce qui permet de traiter activement les demandes. L'architecture fait également appel au mode de veille permanente : en cas de défaillance de l'instance Amazon RDS principale (ou de l'AZ elle-même), l'instance de secours est promue en instance principale.

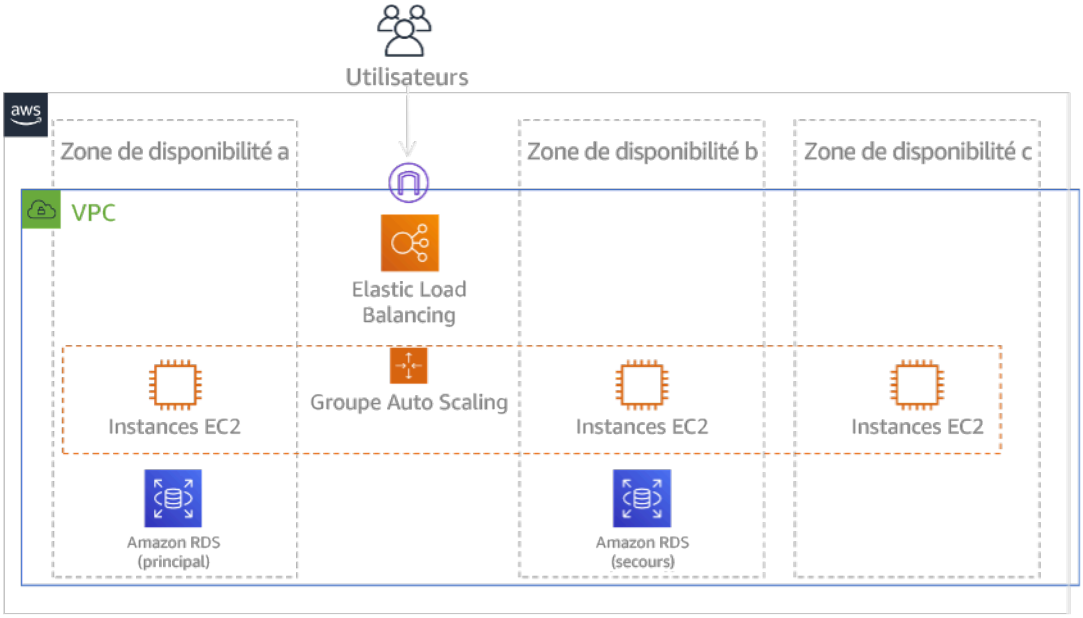


Figure 24 : architecture de multi-AZ

En plus de cette architecture haute disponibilité, vous devez ajouter des sauvegardes de toutes les données requises pour exécuter votre charge de travail. Une telle mesure est particulièrement importante pour les données limitées à une seule zone, telles que les volumes Amazon EBS ou les clusters Amazon Redshift. Si une zone de disponibilité tombe en panne, vous devrez restaurer ces données dans une autre zone de disponibilité. Dans la mesure du possible, vous devez également copier les sauvegardes de données dans une autre Région AWS comme couche de protection supplémentaire.

Une approche alternative moins courante à la reprise après sinistre multi-AZ à une seule région est illustrée dans le billet de blog intitulé Création d'applications hautement résilientes à l'aide d'Amazon Application Recovery Controller, partie 1 : pile dans une seule région. Dans ce cas, la stratégie consiste à maintenir autant que possible l'isolement entre les zones de disponibilité, à l'instar du fonctionnement des régions. Avec cette stratégie alternative, vous pouvez choisir une approche active/active ou active/passive.

> **Note**
>
> Certaines charges de travail sont soumises à des exigences réglementaires en matière de résidence des données. Si cela s'applique à votre charge de travail dans une localité qui n'a actuellement qu'une seule Région AWS, plusieurs régions ne répondront pas aux besoins de votre entreprise. Les stratégies multi-AZ assurent une bonne protection contre la plupart des catastrophes.

3. Évaluez les ressources de votre charge de travail et déterminez quelle sera leur configuration dans la région de reprise avant le basculement (pendant le fonctionnement normal).

   Pour l'infrastructure et les ressources AWS, utilisez l'infrastructure sous forme de code tel que AWS CloudFormation ou des outils tiers tels que Hashicorp Terraform. Pour un déploiement sur plusieurs comptes et régions en une seule opération, vous pouvez utiliser AWS CloudFormation StackSets. Pour les stratégies « Multisite actif/actif » et « Veille permanente », l'infrastructure déployée dans la région de reprise dispose des mêmes ressources que la région principale. Pour les stratégies « Environnement en veille » et « Secours à chaud », l'infrastructure déployée nécessitera des actions supplémentaires pour être prête pour la production. À l'aide des paramètres et de la logique conditionnelle de CloudFormation, vous pouvez contrôler si une pile déployée est active ou en veille avec un seul modèle. En utilisant Elastic Disaster Recovery, le service répliquera et orchestrera la restauration des configurations d'applications et des ressources informatiques.

   Toutes les stratégies de reprise après sinistre nécessitent que les sources de données soient sauvegardées dans la Région AWS, puis que ces sauvegardes soient copiées dans la région de restauration. AWS Backup fournit une vue centralisée dans laquelle vous pouvez configurer, planifier et surveiller les sauvegardes de ces ressources. Pour les stratégies « Environnement en veille », « Secours à chaud » et « Multisite actif/actif », vous devez également répliquer les données de la région principale vers les ressources de données de la région de reprise, telles que des instances de base de données Amazon Relational Database Service (Amazon RDS) ou les tables Amazon DynamoDB. Ces ressources de données sont donc actives et prêtes à répondre aux demandes dans la région de reprise.

   Pour en savoir plus sur le fonctionnement des services AWS dans les différentes régions, consultez cette série de blogs sur la création d'une application multi-régionale avec des services AWS.

4. Déterminez et mettez en œuvre la manière dont vous préparerez votre région de reprise pour le basculement en cas de besoin (lors d'un sinistre).

   Pour la stratégie multisite actif/actif, le basculement consiste à évacuer une région et à s'appuyer sur les régions actives restantes. En général, ces régions sont prêtes à accepter du trafic. Pour les stratégies « Environnement en veille » et « Secours à chaud », vos actions de reprise devront déployer les ressources manquantes, telles que les instances EC2 de la figure 20, ainsi que toute autre ressource manquante.

   Pour toutes les stratégies ci-dessus, vous devrez peut-être promouvoir les instances en lecture seule des bases de données au rang d'instances principales en lecture/écriture.

   Pour la sauvegarde et la restauration, la restauration des données à partir de la sauvegarde crée des ressources pour ces données, telles que des volumes EBS, des instances de base de données RDS et des tables DynamoDB. Vous devez également restaurer l'infrastructure et déployer le code. Vous pouvez utiliser AWS Backup pour restaurer les données dans la région de reprise. Pour plus d'informations, consultez REL09-BP01 Identifier et sauvegarder toutes les données qui doivent être sauvegardées, ou reproduire les données à partir de sources. La reconstruction de l'infrastructure inclut la création de ressources telles que des instances EC2 en plus du Virtual Private Cloud (VPC) Amazon, des sous-réseaux et des groupes de sécurité nécessaires. Vous pouvez automatiser une grande partie du processus de restauration. Pour savoir comment procéder, consultez ce billet de blog.

5. Déterminez et mettez en œuvre la manière dont vous redirigerez le trafic vers le basculement en cas de besoin (lors d'un sinistre).

   Cette opération de basculement peut être lancée automatiquement ou manuellement. Le basculement lancé automatiquement sur la base de la surveillance de l'état ou d'alarmes doit être utilisé avec prudence, car un basculement inutile (fausse alerte) entraînerait des coûts tels que l'indisponibilité et la perte de données. Le basculement manuel est donc souvent utilisé. Dans ce cas, nous vous conseillons tout de même d'automatiser les étapes de basculement, de sorte que vous n'ayez à appuyer que sur un bouton pour lancer le basculement.

   Il existe plusieurs options de gestion du trafic à prendre en compte lors de l'utilisation des services AWS. L'une des options consiste à utiliser Amazon Route 53. Amazon Route 53 vous permet d'associer plusieurs points de terminaison IP dans une ou plusieurs Régions AWS avec un nom de domaine Route 53. Pour mettre en œuvre le basculement initié manuellement, vous pouvez utiliser Amazon Application Recovery Controller, qui fournit une API de plan de données hautement disponible pour rediriger le trafic vers la région de récupération. Lors de la mise en œuvre du

basculement, utilisez les opérations du plan de données et évitez celles du plan de contrôle, comme décrit dans REL11-BP04 S'appuyer sur le plan de données et non sur le plan de contrôle pendant la récupération.

Pour en savoir plus à ce sujet et sur d'autres options, consultez cette section du livre blanc sur la reprise après sinistre.

6. Élaborez un plan pour déterminer la façon dont votre charge de travail se rétablira.

   Failback consiste à renvoyer l'exploitation de la charge de travail à la région principale, après qu'un événement de sinistre s'est atténué. La mise en service de l'infrastructure et du code dans la région principale suit généralement les mêmes étapes que celles utilisées initialement. Elle s'appuie notamment sur l'infrastructure en tant que code et les pipelines de déploiement de code. Le défi posé par failback consiste à restaurer les magasins de données et à garantir leur cohérence avec la région de reprise en cours d'exécution.

   Lors de l'état de basculement, les bases de données de la région de reprise sont actives et disposent des données à jour. L'objectif est alors de resynchroniser les données de la région de reprise vers la région principale, en s'assurant qu'elle est à jour.

   Certains services AWS effectuent cette opération automatiquement. Si vous utilisez des tables globales Amazon DynamoDB, même si la table de la région principale devenait indisponible, DynamoDB reprendrait la propagation de toutes les écritures en attente lorsqu'elle se reconnecterait. Si vous utilisez Amazon Aurora Global Database et que vous utilisez un basculement planifié géré, la topologie de réplication existante de la base de données globale Aurora est conservée. Par conséquent, l'ancienne instance en lecture/écriture de la région principale deviendra un réplica et recevra les mises à jour de la région de reprise.

   Dans les cas où cela n'est pas automatique, vous devrez rétablir la base de données dans la région principale en tant que réplica de la base de données dans la région de reprise. Dans de nombreux cas, cela implique la suppression de l'ancienne base de données principale et la création de nouveaux réplicas.

   Après un basculement, si vous pouvez poursuivre l'exécution dans la région de reprise, envisagez d'en faire la nouvelle région principale. Vous devriez alors suivre toutes les étapes ci-dessus pour convertir l'ancienne région principale en région de reprise. Certaines organisations effectuent une rotation planifiée, en échangeant périodiquement leurs régions principale et de reprise (par exemple tous les trois mois).

Toutes les étapes nécessaires au basculement et au rétablissement doivent être conservées dans un playbook accessible à tous les membres de l'équipe et révisé périodiquement.

En utilisant Elastic Disaster Recovery, le service permettra d'orchestrer et d'automatiser le processus de failback. Pour en savoir plus, consultez la section Réalisation d'un failback.

Niveau d'effort du plan d'implémentation : élevé

Ressources

Bonnes pratiques associées:

- the section called "REL09-BP01 Identifier et sauvegarder toutes les données qui doivent être sauvegardées, ou reproduire les données à partir de sources"
- the section called "REL11-BP04 S'appuyer sur le plan de données et non sur le plan de contrôle pendant la récupération"
- the section called "REL13-BP01 Définir les objectifs de reprise pour les temps d'arrêt et les pertes de données"

Documents connexes:

- AWS Blog d'architecture : série sur la reprise après sinistre
- Reprise après sinistre des charges de travail sur AWS : reprise dans le cloud (livre blanc AWS)
- Options de reprise après sinistre dans le cloud
- Créer une solution dorsale active-active sans serveur sur plusieurs régions en une heure
- Solution dorsale sans serveur sur plusieurs régions – rechargé
- RDS : réplication d'un réplica en lecture entre les régions
- Route 53 : configuration du basculement DNS
- S3 : réplication entre régions
- Qu'est-ce que AWS Backup?
- Qu'est-ce qu'Amazon Application Recovery Controller ?
- AWS Reprise après sinistre Elastic
- HashiCorp Terraform : Démarrage – AWS

- Partenaire APN : partenaires pouvant faciliter la reprise après sinistre
- AWS Marketplace: produits pouvant être utilisés pour la reprise après sinistre

Vidéos connexes :

- Disaster Recovery of Workloads on AWS
- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2)
- Get Started with AWS Elastic Disaster Recovery | Amazon Web Services

REL13-BP03 Effectuer un test de validation de la mise en œuvre de la reprise après sinistre

Testez régulièrement le basculement vers votre site de reprise pour vérifier qu'il fonctionne correctement et que les RTO et RPO sont respectés.

Anti-modèles courants :

- Ne jamais exécuter de basculements en production.

Avantages du respect de cette bonne pratique : en testant régulièrement votre plan de reprise après sinistre, vous vous assurez qu'il fonctionnera quand il le faudra et que votre équipe sait comment exécuter la stratégie.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : élevé

Directives d'implémentation

S'il y a bien un modèle à éviter, c'est celui qui consiste à développer des chemins de récupération rarement testés. Par exemple, vous pouvez avoir un magasin de données secondaire qui est utilisé pour les requêtes en lecture seule. Lorsque vous écrivez dans un magasin de données et que l'instance principale connaît une défaillance, vous pouvez basculer vers le magasin de données secondaire. Si vous ne testez pas fréquemment ce basculement, vous constaterez peut-être que vos hypothèses sur les capacités du magasin de données secondaire sont incorrectes. La capacité du magasin de données secondaire, qui peut avoir été suffisante lors de votre dernier test, peut ne plus être en mesure de tolérer la charge dans le cadre de ce scénario. Notre expérience a montré que le seul chemin de récupération après erreur qui fonctionne est celui que vous testez fréquemment. C'est

pourquoi l'idéal est de n'avoir qu'un petit nombre de chemins de récupération. Vous pouvez établir des modèles de reprise et tester ceux-ci régulièrement. Si vous avez un chemin de récupération complexe ou critique, vous devez toujours exécuter régulièrement cette panne en production pour vous assurer du bon fonctionnement de ce chemin de récupération. Dans l'exemple que nous venons de présenter, vous devez procéder régulièrement au basculement vers l'instance de secours, quel que soit le besoin.

Étapes d'implémentation

1. Préparez vos charges de travail pour la reprise. Testez régulièrement vos chemins de récupération. L'informatique orientée récupération identifie les caractéristiques des systèmes qui améliorent la récupération : isolement et redondance, capacité de l'ensemble du système à réduire les modifications, capacité à surveiller et déterminer l'état de santé, capacité à fournir des diagnostics, reprise automatique, conception modulaire et capacité à redémarrer. Entraînez votre chemin de reprise pour vérifier qu'il peut s'effectuer au moment et à l'état spécifiés. Utilisez vos runbooks au cours de cette reprise pour documenter les problèmes et trouver des solutions pour les résoudre avant le prochain test.
2. Pour les charges de travail basées sur Amazon EC2, utilisez Reprise après sinistre AWS Elastic pour implémenter et lancer des instances de test dans le cadre de votre stratégie de reprise après sinistre. Reprise après sinistre AWS Elastic permet d'exécuter des tests de manière efficace, ce qui vous aide à vous préparer en cas de basculement. Vous pouvez également lancer fréquemment vos instances en utilisant Elastic Disaster Recovery à des fins de test et d'opération sans rediriger le trafic.

Ressources

Documents connexes:

- Partenaire APN : partenaires pouvant faciliter la reprise après sinistre
- AWS Blog d'architecture : série sur la reprise après sinistre
- AWS Marketplace : produits pouvant être utilisés pour la reprise après sinistre
- Reprise après sinistre AWS Elastic
- Reprise après sinistre des charges de travail sur AWS : reprise dans le cloud (livre blanc AWS)
- Préparation au basculement Reprise après sinistre AWS Elastic
- Projet informatique orientée reprise Berkeley/Stanford
- Qu'est qu'AWS Fault Injection Simulator (AWS FIS) ?

Vidéos connexes :

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications
- AWS re:Invent 2019: Backup-and-restore and disaster-recovery solutions with AWS

REL13-BP04 Gérer l'écart de configuration au niveau du site ou de la région de reprise après sinistre

Pour mener à bien une procédure de reprise après sinistre (DR), votre charge de travail doit être en mesure de reprendre son fonctionnement normal en temps opportun, sans aucune perte de fonctionnalité ni de données une fois que l'environnement de reprise après sinistre a été mis en ligne. Pour atteindre cet objectif, il est essentiel de maintenir une infrastructure, des données et des configurations cohérentes entre votre environnement de reprise après sinistre et l'environnement principal.

Résultat escompté : la configuration et les données de votre site de reprise après sinistre sont identiques à celles du site principal, ce qui permet une récupération rapide et complète au moment requis.

Anti-modèles courants :

- Vous ne mettez pas à jour les emplacements de récupération lorsque des modifications sont apportées aux emplacements principaux, ce qui entraîne une obsolescence des configurations, susceptible d'entraver les efforts de récupération.
- Vous ne tenez pas compte des limitations potentielles telles que les différences de service entre les emplacements principaux et de récupération, ce qui peut entraîner des échecs inattendus lors du basculement.
- Vous vous appuyez sur des processus manuels pour mettre à jour et synchroniser l'environnement de reprise après sinistre, ce qui augmente le risque d'erreur humaine et d'incohérence.
- Vous ne détectez pas une dérive de configuration et avez une fausse impression de l'état de préparation du site de reprise après sinistre avant un incident.

Avantages liés au respect de cette bonne pratique : la cohérence entre l'environnement de reprise après sinistre et l'environnement principal améliore considérablement les chances de réussite de la récupération après un incident et réduit le risque d'échec de la procédure de récupération.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : élevé

Directives d’implémentation

Une approche globale de la gestion de la configuration et de la préparation au basculement peut vous aider à vérifier que le site de reprise après sinistre est régulièrement mis à jour et prêt à prendre le relais en cas de défaillance du site principal.

Pour garantir la cohérence entre votre environnement principal et votre environnement de reprise après sinistre (DR), assurez-vous que vos pipelines de distribution répartissent les applications à la fois sur votre site principal et sur votre site de reprise après sinistre. Déployez les modifications sur les sites de reprise après une période d’évaluation appropriée (on parle alors de déploiements échelonnés) afin de détecter les problèmes sur le site principal et d’arrêter le déploiement avant qu’ils ne se propagent. Mettez en œuvre une surveillance pour détecter les dérives de configuration et suivre les modifications et la conformité dans l’ensemble de vos environnements. Procédez à des corrections automatisées sur le site de reprise après sinistre pour qu’il reste totalement cohérent et prêt à prendre le relais en cas d’incident.

Étapes d’implémentation

1. Vérifiez que la région de reprise après sinistre contient les services et fonctionnalités AWS nécessaires à la bonne exécution de votre plan de reprise après sinistre.
2. Utilisez une infrastructure en tant que code (IaC). Maintenez l’exactitude de vos modèles de configuration de l’infrastructure de production et des applications, et appliquez-les régulièrement à votre environnement de reprise après sinistre. AWS CloudFormation peut détecter des dérives entre ce que vos modèles CloudFormation spécifient et ce qui est réellement déployé.
3. Configurez des pipelines CI/CD pour déployer des applications et des mises à jour d’infrastructure dans tous les environnements, y compris les sites principaux et de reprise après sinistre. Les solutions CI/CD telles qu’AWS CodePipeline peuvent automatiser le processus de déploiement, ce qui réduit le risque de dérive de la configuration.
4. Échelonnez les déploiements entre l’environnement principal et l’environnement de reprise après sinistre. Cette approche permet de déployer et de tester initialement les mises à jour dans l’environnement principal, ce qui permet d’isoler les problèmes sur le site principal avant qu’ils ne soient propagés au site de reprise après sinistre. Cette approche empêche la transmission simultanée des défauts en production et au site de reprise après sinistre et préserve l’intégrité de l’environnement de reprise après sinistre.
5. Surveillez en permanence les configurations des ressources dans l’environnement principal et l’environnement de reprise après sinistre. Des solutions telles qu’AWS Config peuvent aider à

renforcer la conformité des configurations et à détecter les dérives, ce qui permet de maintenir la cohérence des configurations dans tous les environnements.

6. Mettez en œuvre des mécanismes d'alerte pour suivre et signaler toute dérive de configuration, ainsi que toute interruption ou tout retard de réplication des données.
7. Automatisez la correction des dérives de configuration détectées.
8. Planifiez des audits et des contrôles de conformité réguliers pour vérifier l'alignement continu entre les configurations principale et de reprise après sinistre. Les examens périodiques vous aident à maintenir la conformité aux règles définies et à identifier les éventuelles anomalies à corriger.
9. Recherchez des disparités au niveau de la capacité provisionnée, des quotas de service, des limitations et des différences de configuration et de version AWS.

Ressources

Bonnes pratiques associées :

- REL01-BP01 Connaître les quotas de service et les contraintes
- REL01-BP02 Gérer les quotas de service entre les comptes et les régions
- REL01-BP04 Surveiller et gérer les quotas
- REL13-BP03 Effectuer un test de validation de la mise en œuvre de la reprise après sinistre

Documents connexes :

- Correction des ressources AWS non conformes par AWS Config Rules
- AWS Systems Manager Automation
- AWS CloudFormation : Détection de modifications non gérées de la configuration des piles et des ressources
- AWS CloudFormation: détection de tout écart à l'échelle d'une pile CloudFormation
- AWS Systems Manager Automation
- Reprise après sinistre des charges de travail sur AWS : reprise dans le cloud (livre blanc AWS)
- Comment mettre en œuvre une solution de gestion de configuration d'infrastructure sur AWS?
- Correction des ressources AWS non conformes par AWS Config Rules

Vidéos connexes:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2)

Exemples connexes :

- CloudFormation Registry
- Quota Monitor for AWS
- Mise en œuvre de la correction automatique des dérives pour AWS CloudFormation à l'aide d'Amazon CloudWatch et d'AWS Lambda
- AWS Blog d'architecture : série sur la reprise après sinistre
- AWS Marketplace: produits pouvant être utilisés pour la reprise après sinistre
- Automatiser les déploiements sécurisés et sans intervention

REL13-BP05 Automatiser la reprise

Mettez en œuvre des mécanismes de reprise testés et automatisés, à la fois fiables, observables et reproductibles afin de réduire le risque et l'impact sur l'activité d'une panne.

Résultat escompté : vous avez mis en œuvre un flux de travail d'automatisation bien documenté, standardisé et entièrement testé pour les processus de récupération. L'automatisation de la récupération corrige automatiquement les problèmes mineurs qui présentent un faible risque d'indisponibilité ou de perte de données. Vous êtes en mesure d'invoquer rapidement des processus de récupération pour des incidents graves, d'observer le comportement de correction pendant leur fonctionnement et de mettre fin aux processus si vous observez des situations dangereuses ou des défaillances.

Anti-modèles courants :

- Dans le cadre de votre plan de reprise, vous dépendez de composants ou de mécanismes défaillants ou dégradés.
- Vos processus de récupération nécessitent une intervention manuelle, telle que l'accès à la console (également appelé ClickOps).
- Vous lancez automatiquement les procédures de récupération dans les situations présentant un risque élevé d'indisponibilité ou de perte de données.

- Vous omettez d'inclure un mécanisme permettant d'annuler une procédure de récupération (comme un système Andon ou un bouton d'arrêt d'urgence) qui ne fonctionne pas ou qui présente des risques supplémentaires.

Avantages liés au respect de cette bonne pratique :

- Fiabilité, prévisibilité et cohérence accrues des opérations de récupération.
- Capacité à atteindre des objectifs de reprise plus stricts, notamment l'objectif de délai de reprise (RTO) et l'objectif de point de reprise (RPO).
- Diminution du risque d'échec de la récupération lors d'un incident.
- Réduction du risque d'échec associé aux processus de récupération manuels susceptibles de provoquer des erreurs humaines.

Niveau d'exposition au risque si cette bonne pratique n'est pas respectée : moyen

Directives d'implémentation

Pour mettre en œuvre la restauration automatique, vous avez besoin d'une approche globale qui utilise les services et les bonnes pratiques AWS. Pour commencer, identifiez les composants critiques et les points de défaillance potentiels de votre charge de travail. Développez des processus automatisés capables de récupérer vos charges de travail et vos données en cas de panne sans intervention humaine.

Développez l'automatisation de la récupération en utilisant les principes de l'infrastructure en tant que code (IaC). Cela rend votre environnement de récupération cohérent avec l'environnement source et permet de contrôler la version de vos processus de récupération. Pour orchestrer des flux de travail de récupération complexes, envisagez des solutions telles que AWS Systems Manager Automations ou AWS Step Functions.

L'automatisation des processus de récupération présente des avantages considérables et peut vous aider à atteindre plus facilement votre objectif de délai de reprise (RTO) et votre objectif de point de reprise (RPO). Toutefois, vous pouvez rencontrer des situations inattendues susceptibles de provoquer un échec ou de créer de nouveaux risques, tels qu'une durée d'indisponibilité et une perte de données supplémentaires. Pour atténuer ce risque, offrez la possibilité d'arrêter rapidement une automatisation de récupération en cours. Une fois celle-ci arrêtée, vous pouvez enquêter et prendre des mesures correctives.

Pour les charges de travail prises en charge, envisagez des solutions telles qu'AWS Elastic Disaster Recovery (AWS DRS) pour fournir un basculement automatisé. AWS DRS réplique en continu vos machines (notamment le système d'exploitation, la configuration d'état du système, les bases de données, les applications et les fichiers) dans une zone intermédiaire de votre Compte AWS cible et de votre région préférée. En cas d'incident, AWS DRS automatise la conversion de vos serveurs répliqués en charges de travail entièrement provisionnées dans votre région de récupération sur AWS.

La maintenance et l'amélioration de la récupération automatisée sont un processus continu. Testez et affinez continuellement vos procédures de récupération sur la base des enseignements acquis, et tenez-vous au fait des nouveaux services et fonctionnalités AWS susceptibles d'améliorer vos capacités de récupération.

Étapes d'implémentation

1. Planifier une récupération automatisée
    a. Réalisez un examen approfondi de l'architecture, des composants et des dépendances de votre charge de travail afin d'identifier et de planifier des mécanismes de récupération automatisés. Classez les dépendances de votre charge de travail en dépendances strictes et souples. Les dépendances strictes sont celles sans lesquelles la charge de travail ne peut pas fonctionner et que rien ne peut substituer. Les dépendances souples sont celles que la charge de travail utilise habituellement, mais qui peuvent être remplacées par des systèmes ou des processus de substitution temporaires ou qui peuvent être traitées par une dégradation appropriée.
    b. Établissez des processus pour identifier et récupérer les données manquantes ou corrompues.
    c. Définissez les étapes permettant de confirmer le rétablissement d'un état stable après l'exécution des actions de récupération.
    d. Envisagez toutes les actions nécessaires pour préparer le système récupéré à être pleinement opérationnel, telles que la préparation et le remplissage des caches.
    e. Tenez compte des problèmes susceptibles d'être rencontrés au cours du processus de récupération et de la manière de les détecter et de les corriger.
    f. Envisagez des scénarios dans lesquels le site principal et son plan de contrôle sont inaccessibles. Vérifiez que les actions de récupération peuvent être effectuées indépendamment, sans avoir recours au site principal. Envisagez des solutions telles qu'Amazon Application Recovery Controller (ARC) pour rediriger le trafic sans qu'il soit nécessaire de muter manuellement les enregistrements DNS.
2. Développer un processus de récupération automatisé

a. Mettez en œuvre des mécanismes automatisés de détection des pannes et de basculement pour une récupération sans intervention manuelle. Créez des tableaux de bord avec des outils tels qu'Amazon CloudWatch pour rendre compte de la progression et de l'état des procédures de récupération automatisées. Incluez des procédures pour valider la réussite de la récupération. Fournissez un mécanisme permettant d'annuler une récupération en cours.

b. Créez des playbooks comme processus de secours pour les pannes qui ne permettent pas une récupération automatique, et tenez compte de votre plan de reprise après sinistre.

c. Testez les processus de récupération comme indiqué dans le document REL13-BP03.

3. Préparer la récupération

a. Évaluez l'état de votre site de reprise et déployez-y les composants stratégiques à l'avance. Pour plus de détails, consultez REL13-BP04.

b. Définissez des rôles, des responsabilités et des processus décisionnels clairs pour les opérations de récupération, en impliquant les parties prenantes et les équipes sur l'ensemble de l'organisation.

c. Définissez les conditions pour lancer vos processus de récupération.

d. Créez un plan pour annuler le processus de récupération et revenir à votre site principal si nécessaire ou une fois que celui-ci est considéré comme sûr.

Ressources

Bonnes pratiques associées :

- REL07-BP01 Utiliser l'automatisation lors de l'obtention des ressources ou de leur mise à l'échelle
- REL11-BP01 Surveiller tous les composants de la charge de travail pour détecter les défaillances
- REL13-BP02 Utiliser des stratégies de reprise définies pour répondre aux objectifs de reprise
- REL13-BP03 Effectuer un test de validation de la mise en œuvre de la reprise après sinistre
- REL13-BP04 Gérer la dérive de configuration au niveau du site ou de la région de reprise après sinistre

Documents connexes :

- AWS Architecture Blog: Disaster Recovery Series
- Disaster Recovery of Workloads on AWS: Recovery in the Cloud (AWS Whitepaper)

- Orchestration de l’automatisation de la reprise après sinistre à l’aide d’Amazon Route 53 ARC et d’AWS Step Functions
- Création de dossiers d¦exploitation AWS Systems Manager Automation à l’aide d’AWS CDK
- AWS Marketplace : produits pouvant être utilisés pour la reprise après sinistre
- AWS Systems Manager Automation
- Reprise après sinistre Elastic AWS
- Utilisation d’Elastic Disaster Recovery pour le basculement et le failback
- Ressources de reprise après sinistre Elastic AWS
- Partenaire APN : partenaires pouvant faciliter la reprise après sinistre

Vidéos connexes :

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2)
- AWS re:Invent 2022: AWS On Air ft. AWS Failback for AWS Elastic Disaster Recovery

Conclusion

Que vous commenciez juste à découvrir les sujets de la disponibilité et de la fiabilité ou que vous soyez un utilisateur aguerri cherchant plus de connaissances pour optimiser la disponibilité des charges de travail critiques, nous espérons que ce livre blanc vous a poussé à réfléchir à ce sujet, vous a présenté de nouvelles idées ou a soulevé de nouvelles questions. Nous espérons que cela vous permettra d'obtenir une meilleure compréhension du niveau de disponibilité qui correspond aux besoins de votre entreprise et de la procédure de mise en œuvre de la fiabilité pour y parvenir. Nous vous encourageons à tirer parti des recommandations relatives à la conception, à l'exploitation et à la restauration proposées ici, ainsi que des connaissances et de l'expérience de nos AWS architectes de solutions. Nous aimerions avoir de vos nouvelles, en particulier si vous avez réussi à atteindre des niveaux élevés de disponibilité sur. AWS Contactez l'équipe de votre compte ou utilisez la fonction Contactez-nous sur notre site Web.

Collaborateurs

Les personnes qui ont contribué à ce document incluent :

- Michael Fischer, architecte principal de solutions, Amazon Web Services
- Seth Eliot, défenseur principal des développeurs, Amazon Web Services
- Mahanth Jayadeva, architecte de solutions Well-Architected, Amazon Web Services
- Amulya Sharma, architecte principal de solutions, Amazon Web Services
- Jason DiDomenico, architecte de solutions senior, Cloud Foundations, Amazon Web Services
- Marcin Bednarz, architecte principal de solutions, Amazon Web Services
- Tyler Applebaum, architecte principal de solutions, Amazon Web Services
- Rodney Lester, architecte principal des solutions, Amazon Web Services
- Joe Chapman, architect principal de solutions, Amazon Web Services
- Adrian Hornsby, ingénieur principal en développement système, Amazon Web Services
- Kevin Miller, vice-président de S3, Amazon Web Services
- Shannon Richards, responsable principal de programmes techniques, Amazon Web Services
- Laurent Domb, technologue en chef, Fed Fin, Amazon Web Services
- Kevin Schwarz, architecte de solutions principal, Amazon Web Services
- Rob Martell, architecte principal de la résilience du cloud, Amazon Web Services
- Priyam Reddy, architecte de solutions senior et responsable DR, Amazon Web Services
- Jeff Ferris, technologue principal, Amazon Web Services
- Matias Battaglia, architecte principal de solutions, Amazon Web Services

Suggestions de lecture

Pour en savoir plus, voir :

- AWS Framework Well-Architected
- Centre d’architecture AWS

Révisions du document

Pour être informé des mises à jour de ce livre blanc, abonnez-vous au flux RSS.

Modification	Description	Date
Mises à jour des conseils sur les bonnes pratiques	Les bonnes pratiques ont été mises à jour avec de nouvelles directives dans les domaines suivants : REL 1, REL 2, REL 4, REL 6, REL 7, REL 8, REL 10, REL 12 et REL 13. Les recommandations ont été étendues et clarifiées sur l'ensemble du pilier. Les recommandations des bonnes pratiques REL10-BP02 et REL12-BP03 ont été fusionnées avec d'autres bonnes pratiques. Les ressources ont été mises à jour sur l'ensemble du pilier.	6 novembre 2024
Mises à jour des conseils sur les bonnes pratiques	Petites mises à jour des meilleures pratiques dans REL 2, 4, 5, 6, 7 et 8.	27 juin 2024
Mises à jour des conseils sur les bonnes pratiques	Les meilleures pratiques ont été mises à jour avec de nouvelles directives dans les domaines suivants : concevoir des interactions dans un système distribué pour éviter les défaillances, concevoir des interactions dans un système distribué pour atténuer ou résister aux défaillances,	6 décembre 2023

	surveiller les ressources de charge de travail, concevoir votre charge de travail pour qu'elle s'adapte à l'évoluti on de la demande, mettre en œuvre les changements et tester la fiabilité.	
Mises à jour des conseils sur les bonnes pratiques	Les meilleures pratiques ont été mises à jour avec de nouvelles directives dans les domaines suivants : surveillez les ressources de charge de travail et concevez votre charge de travail pour résister aux défaillances des composants.	3 octobre 2023
Mises à jour des conseils sur les bonnes pratiques	Les meilleures pratiques ont été mises à jour avec de nouvelles directives dans les domaines suivants : concevoir l'architecture de votre service de charge de travail, concevoir des interacti ons dans un système distribué pour atténuer ou résister aux défaillances, et surveiller les ressources de charge de travail.	13 juillet 2023
Mise à jour mineure	Suppression du langage non inclusif.	13 avril 2023

Mises à jour du nouveau cadre	Les bonnes pratiques ont été mises à jour avec des recommandations et de nouvelles bonnes pratiques.	10 avril 2023
Livre blanc mis à jour	Les bonnes pratiques ont été mises à jour avec de nouvelles recommandations en matière d'implémentation.	15 décembre 2022
Mises à jour mineures	Chiffres corrigés et modificat ions mineures.	17 novembre 2022
Livre blanc mis à jour	Développement des bonnes pratiques et ajout de plans d'amélioration.	20 octobre 2022
Livre blanc mis à jour	Deux nouvelles meilleures pratiques ont été ajoutées au pilier de fiabilité dans les sections Utiliser l'isolation des défauts pour protéger votre charge de travail et concevoir votre charge de travail de manière à résister aux défaillances des composants.	5 mai 2022
Livre blanc mis à jour	Mise à jour des conseils de reprise après sinistre pour inclure Route 53 Application Recovery Controller. Ajout de références à DevOps Guru. Mise à jour de plusieurs liens de ressources et autres modifications éditoriales mineures.	26 octobre 2021

Mise à jour mineure	Ajout d'informations sur AWS Fault Injection Service (AWS FIS).	15 mars 2021
Mise à jour mineure	Mise à jour mineure du texte.	4 janvier 2021
Livre blanc mis à jour	Mise à jour de l'annexe A pour refléter l'objectif de conceptio n Disponibilité pour Amazon SQS, Amazon SNS et Amazon MQ. Réorganisation des lignes dans le tableau pour faciliter la recherche. Amélioration de l'explication des différenc es entre la disponibilité et la reprise après sinistre et comment elles contribuent toutes deux à la résilience. Détails supplémentaires sur la couverture des architect ures multi-régionales (pour la disponibilité) et des stratégie s multi-régionales (pour la reprise après sinistre). Mise à jour du livre référencé vers la dernière version. Détails supplémentaires sur les calculs de disponibilité pour inclure le calcul basé sur les demandes et les calculs de raccourci. Amélioration de la description des jeux de rôle.	7 décembre 2020
Mise à jour mineure	Mise à jour de l'annexe A pour refléter l'objectif de la conception de la disponibilité pour AWS Lambda	27 octobre 2020

Mise à jour mineure	Mise à jour de l’annexe A pour ajouter l’objectif de la conception de la disponibilité pour AWS Global Accelerator	24 juillet 2020

Mises à jour du nouveau cadre	Mises à jour importantes et contenu nouveau/révisé, y compris : ajout de la section « Architecture de la charge de travail » sur les bonnes pratiques, réorganisation des bonnes pratiques dans les sections Gestion des modifications et Gestion des pannes, ressources mises à jour, mise à jour pour inclure les dernières ressources et les derniers services AWS comme AWS Global Accelerator, AWSService Quotas et AWS Transit Gateway, ajout/mise à jour des définitions des termes Fiabilité, Disponibilité et Résilience, Livre blanc mieux adapté à l'outil AWS Well-Architected Tool (questions et bonnes pratiques) utilisé pour les évaluations Well-Architected, réorganisation des principes de conception, déplacement de Récupérer automatiquement une défaillance avant Tester les procédures de récupération, mise à jour des diagrammes et des formats pour les équations, suppression des sections Services clés de AWS et intégration à la place des références aux services clés dans les bonnes pratiques.	8 juillet 2020

Mise à jour mineure	Correction d’un lien rompu	1er octobre 2019
Livre blanc mis à jour	Mise à jour de l’annexe A	1er avril 2019
Livre blanc mis à jour	Ajout de recommandations de mise en réseau spécifiques à AWS Direct Connect et d’objectifs de conception de service supplémentaires	1er septembre 2018
Livre blanc mis à jour	Ajout des sections Principes de conception et Gestion des limites. Mise à jour des liens, suppression de l’ambiguïté des expressions en amont/en aval et ajout de références explicites aux rubriques restantes du pilier Fiabilité dans les scénarios de disponibilité.	1er juin 2018
Livre blanc mis à jour	Solution DynamoDB entre régions remplacée par DynamoDB Global Tables Ajout d’objectifs de conception de service	1er mars 2018
Mises à jour mineures	Correction mineure du calcul de la disponibilité pour inclure la disponibilité de l’application	1er décembre 2017
Livre blanc mis à jour	Mise à jour pour fournir des conseils sur les conceptions haute disponibilité, y compris les concepts, les bonnes pratiques et les exemples d’implémentation.	1er novembre 2017

Publication initiale	Pilier Fiabilité - AWS Well-Architected Framework publié.	1er novembre 2016

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 AWS produits et les pratiques actuelles, qui sont susceptibles d'être modifiées sans préavis, et (c) ne crée aucun engagement ni aucune assurance de la part de AWS ses filiales, fournisseurs ou concédants de licence. AWS les produits ou services sont fournis « tels quels » sans garanties, déclarations ou conditions d'aucune sorte, qu'elles soient explicites ou implicites. Les responsabilités et obligations AWS de ses clients sont régies par AWS des accords, et ce document ne fait partie d'aucun accord conclu entre AWS et ses clients et ne les modifie pas.

AWS Glossaire

Pour la AWS terminologie la plus récente, consultez le AWS glossaire dans la Glossaire AWS référence.