Choisir une stratégie de branchement Git pour les environnements multi-comptes DevOps

AWS Conseils prescriptifs

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS Conseils prescriptifs: Choisir une stratégie de branchement Git pour les environnements multi-comptes DevOps

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

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

Table of Contents

Introduction 1
Objectifs 1
Utilisation des pratiques CI/CD 2
Comprendre les DevOps environnements 4
Environnement Sandbox 5
Accès 5
Étapes de construction 5
Étapes de déploiement 6
Attentes avant de passer à l'environnement de développement 6
Environnement de développement 6
Accès 5
Étapes de construction 5
Étapes de déploiement 6
Attentes avant de passer à l'environnement de test 7
Environnement de test 8
Accès 5
Étapes de construction 5
Étapes de déploiement 6
Attentes avant de passer à l'environnement de mise en scène 9
Environnement de mise en scène 9
Accès 5
Étapes de construction 5
Étapes de déploiement 6
Attentes avant de passer à l'environnement de production 10
Environnement de production 11
Accès 5
Étapes de construction 5
Étapes de déploiement 6
Bonnes pratiques pour le développement basé sur Git 12
Stratégies de branchement Git 14
Stratégie de branchement principal 14
Aperçu visuel de la stratégie Trunk 15
Stratégie Branches in a Trunk 16
Avantages et inconvénients de la stratégie Trunk 18

GitHub Stratégie de branchement des flux 21
Aperçu visuel de la stratégie GitHub Flow 22
Branches dans une stratégie GitHub Flow 23
Avantages et inconvénients de la stratégie GitHub Flow 25
Stratégie de branchement Gitflow 27
Aperçu visuel de la stratégie Gitflow 28
Branches dans une stratégie Gitflow 30
Avantages et inconvénients de la stratégie Gitflow 34
Étapes suivantes 36
Ressources 37
AWS Conseils prescriptifs 37
Autres AWS conseils 37
Autres ressources 37
Collaborateurs 39
Conception 39
Révision 39
Rédaction technique 39
Historique du document 40
Glossaire 41
..... 41
A 42
B 45
C 47
D 50
E 55
F 57
G 58
H 59
I 61
L 63
M 64
O 69
P 71
Q 74
R 75
S 78

T 81
U 83
V 84
W 84
Z 85
..... lxxxvii

Choisir une stratégie de branchement Git pour les environnements multi-comptes DevOps

Amazon Web Services (contributeurs)

Février 2024 (historique du document)

L'adoption d'une approche basée sur le cloud et la fourniture de solutions logicielles AWS peuvent être transformatrices. Cela peut nécessiter des modifications de votre processus de cycle de vie de développement logiciel. Généralement, plusieurs Comptes AWS sont utilisés au cours du processus de développement dans le AWS Cloud. Pour réussir, il est essentiel de choisir une stratégie de branchement Git compatible à associer à vos DevOps processus. Le choix de la stratégie de branchement Git adaptée à votre organisation vous permet de communiquer de manière concise les DevOps normes et les meilleures pratiques au sein des équipes de développement. Le branchement Git peut être simple dans un environnement unique, mais il peut être source de confusion lorsqu'il est appliqué à plusieurs environnements, tels que les environnements de sandbox, de développement, de test, de préparation et de production. Le fait de disposer de plusieurs environnements augmente la complexité de la DevOps mise en œuvre.

Ce guide fournit des diagrammes visuels des stratégies de branchement de Git qui montrent comment une organisation peut implémenter un processus multi-comptes DevOps . Les guides visuels aident les équipes à comprendre comment intégrer leurs stratégies de branchement Git à leurs DevOps pratiques. L'utilisation d'un modèle de branchement standard, tel que GitHub Gitflow, Flow ou Trunk, pour gérer le référentiel de code source aide les équipes de développement à aligner leur travail. Ces équipes peuvent également utiliser les ressources de formation Git standard sur Internet pour comprendre et mettre en œuvre ces modèles et stratégies.

Pour connaître les DevOps meilleures pratiques en la matière AWS, consultez le DevOpsguide publié dans AWS Well-Architected. Pendant que vous consultez ce guide, faites preuve de diligence raisonnable pour sélectionner la stratégie de succursale adaptée à votre organisation. Certaines stratégies peuvent mieux correspondre à votre cas d'utilisation que d'autres.

Objectifs

Ce guide fait partie d'une série de documentation sur le choix et la mise en œuvre de stratégies de création de DevOps succursales pour les organisations qui en ont plusieurs Comptes AWS. Cette

série est conçue pour vous aider à appliquer la stratégie qui répond le mieux à vos exigences, à vos objectifs et à vos meilleures pratiques dès le départ, afin de rationaliser votre expérience dans le AWS Cloud. Ce guide ne contient pas de scripts DevOps exécutables car ils varient en fonction du moteur d'intégration continue et de livraison continue (CI/CD) et des cadres technologiques utilisés par votre organisation.

Ce guide explique les différences entre trois stratégies de branchement Git courantes : GitHub Flow, Gitflow et Trunk. Les recommandations de ce guide aident les équipes à identifier une stratégie de succursale conforme à leurs objectifs organisationnels. Après avoir lu ce guide, vous devriez être en mesure de choisir une stratégie de succursale pour votre organisation. Après avoir choisi une stratégie, vous pouvez utiliser l'un des modèles suivants pour vous aider à mettre en œuvre cette stratégie avec vos équipes de développement :

- Mettre en œuvre une stratégie de branchement Trunk pour les environnements multi-comptes DevOps
- Mettre en œuvre une stratégie GitHub de branchement Flow pour les environnements multi-comptes DevOps
- Mettre en œuvre une stratégie de branchement Gitflow pour les environnements multi-comptes DevOps

Il est important de noter que ce qui fonctionne pour une organisation, une équipe ou un projet peut ne pas convenir à d'autres. Le choix entre les stratégies de branchement Git dépend de divers facteurs, tels que la taille de l'équipe, les exigences du projet et l'équilibre souhaité entre la collaboration, la fréquence d'intégration et la gestion des versions.

Utilisation des pratiques CI/CD

AWS vous recommande de mettre en œuvre l'intégration continue et la livraison continue (CI/CD), qui sont le processus d'automatisation du cycle de vie des versions logicielles. Il automatise une grande partie ou la totalité des DevOps processus manuels traditionnellement nécessaires pour faire passer le nouveau code du développement à la production. Un pipeline CI/CD englobe les environnements sandbox, de développement, de test, de préparation et de production. Dans chaque environnement, le pipeline CI/CD fournit toute infrastructure nécessaire au déploiement ou au test du code. En utilisant le CI/CD, les équipes de développement peuvent apporter des modifications au code qui sont ensuite automatiquement testées et déployées. Les pipelines CI/CD fournissent également une gouvernance et des garde-fous aux équipes de développement. Ils appliquent la cohérence,

les normes, les meilleures pratiques et les niveaux d'acceptation minimaux pour l'acceptation et le déploiement des fonctionnalités. Pour plus d'informations, voir Pratiquer l'intégration continue et la livraison continue sur AWS.

Toutes les stratégies de branchement décrites dans ce guide sont bien adaptées aux pratiques CI/CD. La complexité du pipeline CI/CD augmente avec la complexité de la stratégie de branchement. Par exemple, Gitflow est la stratégie de branchement la plus complexe décrite dans ce guide. Les pipelines CI/CD utilisés dans le cadre de cette stratégie nécessitent davantage d'étapes (par exemple pour des raisons de conformité) et doivent prendre en charge plusieurs versions de production simultanées. L'utilisation du CI/CD devient également plus importante à mesure que la complexité de la stratégie de branchement augmente. En effet, le CI/CD met en place des garde-fous et des mécanismes pour les équipes de développement qui empêchent les développeurs de contourner intentionnellement ou non le processus défini.

AWS propose une suite de services de développement conçus pour vous aider à créer des pipelines CI/CD. Par exemple, AWS CodePipelineil s'agit d'un service de livraison continue entièrement géré qui vous aide à automatiser vos pipelines de publication pour des mises à jour rapides et fiables des applications et de l'infrastructure. AWS CodeCommitest conçu pour héberger en toute sécurité des référentiels Git évolutifs, AWS CodeBuildcompile le code source, exécute des tests et produit des packages ready-to-deploy logiciels. Pour plus d'informations, consultez la section Outils de développement sur AWS.

Comprendre les DevOps environnements

Pour comprendre les stratégies de branchement, vous devez comprendre l'objectif et les activités de chaque environnement. La mise en place de plusieurs environnements vous permet de séparer les activités de développement en plusieurs étapes, de surveiller ces activités et d'empêcher le lancement involontaire de fonctionnalités non approuvées. Vous pouvez en avoir un ou plusieurs Comptes AWS dans chaque environnement.

La plupart des organisations ont défini plusieurs environnements à utiliser. Cependant, le nombre d'environnements peut varier en fonction de l'organisation et des politiques de développement logiciel. Cette série de documentation part du principe que vous disposez des cinq environnements courants suivants qui couvrent votre pipeline de développement, bien qu'ils puissent porter des noms différents :

- Sandbox : environnement dans lequel les développeurs écrivent du code, commettent des erreurs et réalisent des travaux de validation de concept.
- Développement : environnement dans lequel les développeurs intègrent leur code pour s'assurer qu'il fonctionne comme une seule et même application cohérente.
- Tests : environnement dans lequel les équipes d'assurance qualité ou les tests d'acceptation ont lieu. Les équipes effectuent souvent des tests de performance ou d'intégration dans cet environnement.
- Stage : environnement de préproduction dans lequel vous confirmez que le code et l'infrastructure fonctionnent comme prévu dans des circonstances équivalentes à celles de la production. Cet environnement est configuré pour être aussi similaire que possible à l'environnement de production.
- Production : environnement qui gère le trafic provenant de vos utilisateurs finaux et de vos clients.

Cette section décrit chaque environnement en détail. Il décrit également les étapes de création, les étapes de déploiement et les critères de sortie pour chaque environnement afin que vous puissiez passer au suivant. L'image suivante montre ces environnements dans l'ordre.

Rubriques de cette section :

- Environnement Sandbox
- Environnement de développement
- Environnement de test
- Environnement de mise en scène
- Environnement de production

Environnement Sandbox

L'environnement sandbox est l'endroit où les développeurs écrivent du code, commettent des erreurs et réalisent des travaux de validation de concept. Vous pouvez effectuer un déploiement dans un environnement sandbox à partir d'un poste de travail local ou via un script sur un poste de travail local.

Accès

Les développeurs doivent avoir un accès complet à l'environnement sandbox.

Étapes de construction

Les développeurs exécutent manuellement le build sur leurs postes de travail locaux lorsqu'ils sont prêts à déployer des modifications dans l'environnement sandbox.

1. Utilisez git-secrets (GitHub) pour rechercher des informations sensibles
2. Lint le code source
3. Compilez et compilez le code source, le cas échéant
4. Réaliser des tests unitaires
5. Réaliser une analyse de la couverture du code
6. Effectuez une analyse de code statique
7. Construire une infrastructure sous forme de code (IaC)
8. Réaliser une analyse de sécurité IaC
9. Extraire des licences open source
10. Publier des artefacts de construction

Étapes de déploiement

Si vous utilisez les modèles Gitflow ou Trunk, les étapes de déploiement démarrent automatiquement lorsqu'une `feature` branche est correctement créée dans l'environnement sandbox. Si vous utilisez le modèle GitHub Flow, vous devez exécuter manuellement les étapes de déploiement suivantes. Les étapes de déploiement dans l'environnement sandbox sont les suivantes :

1. Télécharger les artefacts publiés
2. Effectuer le versionnement de la base de données
3. Effectuer le déploiement d'iAc
4. Réaliser des tests d'intégration

Attentes avant de passer à l'environnement de développement

- Création réussie de la `feature` branche dans l'environnement sandbox
- Un développeur a déployé et testé manuellement la fonctionnalité dans l'environnement sandbox

Environnement de développement

L'environnement de développement est l'endroit où les développeurs intègrent leur code afin de garantir qu'il fonctionne comme une seule application cohérente. Dans Gitflow, l'environnement de développement contient les dernières fonctionnalités incluses par demande de fusion et est prêt à être publié. Dans les stratégies GitHub Flow et Trunk, l'environnement de développement est considéré comme un environnement de test, et la base de code peut être instable et ne pas convenir au déploiement en production.

Accès

Attribuez des autorisations selon le principe du moindre privilège. Le moindre privilège est la bonne pratique de sécurité qui consiste à accorder les autorisations minimales nécessaires à l'exécution d'une tâche. Les développeurs devraient avoir moins accès à l'environnement de développement qu'à l'environnement sandbox.

Étapes de construction

La création d'une demande de fusion vers la `develop` branche (Gitflow) ou la `main` branche (Trunk ou GitHub Flow) lance automatiquement la construction.

1. Utilisez git-secrets (GitHub) pour rechercher des informations sensibles
2. Lint le code source
3. Compilez et compilez le code source, le cas échéant
4. Réaliser des tests unitaires
5. Réaliser une analyse de la couverture du code
6. Effectuez une analyse de code statique
7. Construisez iAc
8. Réaliser une analyse de sécurité IaC
9. Extraire des licences open source

Étapes de déploiement

Si vous utilisez le modèle Gitflow, les étapes de déploiement démarrent automatiquement lorsqu'une `develop` branche est correctement créée dans l'environnement de développement. Si vous utilisez le modèle GitHub Flow ou le modèle Trunk, les étapes de déploiement démarrent automatiquement lorsqu'une demande de fusion est créée pour la `main` branche. Les étapes de déploiement dans l'environnement de développement sont les suivantes :

1. Téléchargez les artefacts publiés à partir des étapes de construction
2. Effectuer le versionnement de la base de données
3. Effectuer le déploiement d'iAc
4. Réaliser des tests d'intégration

Attentes avant de passer à l'environnement de test

- Création et déploiement réussis de la `develop` branche (Gitflow) ou de la `main` branche (Trunk ou GitHub Flow) dans l'environnement de développement
- Les tests unitaires passent à 100 %

- Construction réussie d'iAc
- Les artefacts de déploiement ont été créés avec succès
- Un développeur a effectué une vérification manuelle pour confirmer que la fonctionnalité fonctionne comme prévu

Environnement de test

Le personnel chargé de l'assurance qualité (AQ) utilise l'environnement de test pour valider les fonctionnalités. Ils approuvent les modifications une fois les tests terminés. Après approbation, la branche passe à l'environnement suivant, celui du stage. Dans Gitflow, cet environnement et les autres environnements supérieurs ne sont disponibles que pour le déploiement à partir de `release` succursales. Une `release` branche est basée sur une `develop` branche qui contient les fonctionnalités planifiées.

Accès

Attribuez des autorisations selon le principe du moindre privilège. Les développeurs devraient avoir moins accès à l'environnement de test qu'à l'environnement de développement. Le personnel chargé de l'assurance qualité a besoin d'autorisations suffisantes pour tester la fonctionnalité.

Étapes de construction

Le processus de compilation dans cet environnement ne s'applique qu'aux corrections de bogues lors de l'utilisation de la stratégie Gitflow. La création d'une demande de fusion adressée à la `bugfix` branche lance automatiquement la construction.

1. Utilisez git-secrets (GitHub) pour rechercher des informations sensibles
2. Lint le code source
3. Compilez et compilez le code source, le cas échéant
4. Réaliser des tests unitaires
5. Réaliser une analyse de la couverture du code
6. Effectuez une analyse de code statique
7. Construisez iAc
8. Réaliser une analyse de sécurité IaC

9. Extraire des licences open source

Étapes de déploiement

Lancez automatiquement le déploiement de la `release` branche (Gitflow) ou de la `main` branche (Trunk ou GitHub Flow) dans l'environnement de test après le déploiement dans l'environnement de développement. Les étapes de déploiement dans l'environnement de test sont les suivantes :

1. Déployez la `release` branche (Gitflow) ou la `main` branche (Trunk ou GitHub Flow) dans l'environnement de test
2. Pause pour approbation manuelle par le personnel désigné
3. Télécharger les artefacts publiés
4. Effectuer le versionnement de la base de données
5. Effectuer le déploiement d'iAc
6. Réaliser des tests d'intégration
7. Réaliser des tests de performance
8. Approbation d'assurance qualité

Attentes avant de passer à l'environnement de mise en scène

- Les équipes de développement et d'assurance qualité ont effectué suffisamment de tests pour répondre aux exigences de votre organisation.
- L'équipe de développement a résolu tous les bogues découverts par le biais d'une `bugfix` branche.

Environnement de mise en scène

L'environnement de préparation est configuré pour être identique à l'environnement de production. Par exemple, l'étendue et la taille de la configuration des données doivent être similaires à celles des charges de travail de production. Utilisez l'environnement intermédiaire pour vérifier que le code et l'infrastructure fonctionnent comme prévu. Cet environnement est également le choix privilégié pour les cas d'utilisation professionnels, tels que les avant-premières ou les démonstrations destinées aux clients.

Accès

Attribuez des autorisations selon le principe du moindre privilège. Les développeurs doivent avoir le même accès à l'environnement de préparation qu'à l'environnement de production.

Étapes de construction

Aucune. Les mêmes artefacts que ceux utilisés dans l'environnement de test sont réutilisés dans l'environnement de préparation.

Étapes de déploiement

Lancez automatiquement le déploiement de la `release` branche (Gitflow) ou de la `main` branche (Trunk ou GitHub Flow) dans l'environnement de test après approbation et déploiement dans l'environnement de test. Les étapes de déploiement dans l'environnement de préparation sont les suivantes :

1. Déployez la `release` branche (Gitflow) ou la `main` branche (Trunk ou GitHub Flow) dans l'environnement de préparation
2. Pause pour approbation manuelle par le personnel désigné
3. Télécharger les artefacts publiés
4. Effectuer le versionnement de la base de données
5. Effectuer le déploiement d'iAc
6. (Facultatif) Effectuez des tests d'intégration
7. (Facultatif) Effectuer un test de charge
8. Obtenir l'approbation des approbateurs requis en matière de développement, d'assurance qualité, de produit ou d'entreprise

Attentes avant de passer à l'environnement de production

- Une version équivalente à la production a été déployée avec succès dans l'environnement de préparation
- (Facultatif) Les tests d'intégration et de charge ont été réussis

Environnement de production

L'environnement de production prend en charge le produit publié, en gérant les données réelles des clients réels. Il s'agit d'un environnement protégé auquel l'accès est attribué par le moindre privilège et l'accès élevé ne doit être autorisé que dans le cadre d'un processus d'exception audité pendant une période limitée.

Accès

Dans l'environnement de production, les développeurs doivent disposer d'un accès limité en lecture seule à l'AWS Management Console. Par exemple, les développeurs devraient être en mesure d'accéder aux données du journal pour les day-to-day opérations. Toutes les mises en production doivent être soumises à une étape d'approbation avant le déploiement.

Étapes de construction

Aucune. Les mêmes artefacts utilisés dans les environnements de test et de préparation sont réutilisés dans l'environnement de production.

Étapes de déploiement

Lancez automatiquement le déploiement de la `release` branche (Gitflow) ou de la `main` branche (Trunk ou GitHub Flow) dans l'environnement de production après approbation et déploiement dans l'environnement de préparation. Les étapes de déploiement dans l'environnement de production sont les suivantes :

1. Déployez la `release` branche (Gitflow) ou la `main` branche (Trunk ou GitHub Flow) dans l'environnement de production
2. Pause pour approbation manuelle par le personnel désigné
3. Télécharger les artefacts publiés
4. Effectuer le versionnement de la base de données
5. Effectuer le déploiement d'iAc

Bonnes pratiques pour le développement basé sur Git

Pour réussir à adopter le développement basé sur Git, il est important de suivre un ensemble de bonnes pratiques qui favorisent la collaboration, maintiennent la qualité du code et soutiennent l'intégration et la livraison continues (CI/CD). Outre les meilleures pratiques décrites dans ce guide, consultez le guide AWS DevOps Well-Architected. Voici quelques bonnes pratiques clés pour le développement basé sur Git sur AWS :

- Limitez la fréquence et limitez les modifications : encouragez les développeurs à apporter des modifications ou des fonctionnalités mineures et incrémentielles. Cela réduit le risque de conflits de fusion et facilite l'identification et la résolution rapides des problèmes.
- Utiliser des boutons de fonctionnalité : pour gérer la publication de fonctionnalités incomplètes ou expérimentales, utilisez des boutons de fonctionnalité ou des indicateurs de fonctionnalité. Cela vous permet de masquer, d'activer ou de désactiver des fonctionnalités spécifiques en production sans affecter la stabilité de la branche principale.
- Maintenir une suite de tests robuste — Une suite de tests complète et bien entretenue est essentielle pour détecter les problèmes à un stade précoce et vérifier que la base de code reste stable. Investissez dans l'automatisation des tests et donnez la priorité à la correction des tests défaillants.
- Adoptez l'intégration continue : utilisez des outils et des pratiques d'intégration continue pour créer, tester et intégrer automatiquement les modifications de code dans la `develop` branche (Gitflow) ou la `main` branche (Trunk ou GitHub Flow). Cela vous permet de détecter les problèmes à un stade précoce et de rationaliser le processus de développement.
- Réaliser des révisions du code : encouragez les pairs à évaluer le code afin de maintenir la qualité, de partager les connaissances et de détecter les problèmes potentiels avant qu'ils ne soient intégrés à la `main` branche. Utilisez des pull requests ou d'autres outils de révision du code pour faciliter ce processus.
- Surveillez et corrigez les versions défectueuses : lorsqu'une version échoue ou que les tests échouent, donnez la priorité à la résolution du problème dès que possible. Cela permet de maintenir la `develop` branche (Gitflow) ou la `main` branche (Trunk ou GitHub Flow) dans un état libérable et de minimiser l'impact sur les autres développeurs.
- Communiquez et collaborez — Favorisez une communication ouverte et une collaboration entre les membres de l'équipe. Assurez-vous que les développeurs sont au courant des travaux en cours et des modifications apportées à la base de code.

- Refactorisation continue : refactorisez régulièrement la base de code pour améliorer sa maintenabilité et réduire la dette technique. Encouragez les développeurs à laisser le code dans un meilleur état que celui dans lequel ils l'ont trouvé.
- Utilisez des branches éphémères pour les tâches complexes : pour les tâches plus importantes ou plus complexes, utilisez des branches éphémères (également appelées branches de tâches) pour travailler sur les modifications. Cependant, veillez à ce que la durée de vie de la branche soit courte, généralement inférieure à une journée. Fusionnez les modifications dans la `develop` branche (Gitflow) ou dans la `main` branche (Trunk ou GitHub Flow) dès que possible. Les fusions et révisions plus petites et plus fréquentes sont plus faciles à utiliser et à traiter pour une équipe qu'une seule demande de fusion de grande envergure.
- Former et soutenir l'équipe — Fournir une formation et un soutien aux développeurs qui découvrent le développement basé sur Git ou qui ont besoin de conseils pour adopter ses meilleures pratiques.

Stratégies de branchement Git

Dans l'ordre de la plus simple à la plus complexe, ce guide décrit en détail les stratégies de branchement basées sur Git suivantes :

- Trunk — Le développement basé sur le tronc est une pratique de développement logiciel dans laquelle tous les développeurs travaillent sur une seule branche, généralement appelée branche `trunk` or`main`. L'idée qui sous-tend cette approche est de maintenir la base de code dans un état continuellement publiable en intégrant fréquemment les modifications du code et en s'appuyant sur des tests automatisés et une intégration continue.
- GitHub Flow — GitHub Flow est un flux de travail léger basé sur les branches développé par. GitHub Il est basé sur l'idée de `feature` branches éphémères. Lorsqu'une fonctionnalité est terminée et prête à être déployée, elle est fusionnée dans la `main` branche.
- Gitflow — Avec une approche Gitflow, le développement est effectué dans des branches de fonctionnalités individuelles. Après approbation, vous fusionnez `feature` les branches dans une branche d'intégration généralement nommée`develop`. Lorsque suffisamment de fonctionnalités se sont accumulées dans la `develop` branche, une `release` branche est créée pour déployer les fonctionnalités dans les environnements supérieurs.

Chaque stratégie de branchement présente des avantages et des inconvénients. Bien qu'ils utilisent tous les mêmes environnements, ils n'utilisent pas tous les mêmes branches ni les mêmes étapes d'approbation manuelle. Dans cette section du guide, passez en revue chaque stratégie de succursale en détail afin de vous familiariser avec ses nuances et de déterminer si elle correspond au cas d'utilisation de votre organisation.

Rubriques de cette section :

- Stratégie de branchement principal
- GitHub Stratégie de branchement des flux
- Stratégie de branchement Gitflow

Stratégie de branchement principal

Le développement basé sur le tronc est une pratique de développement logiciel dans laquelle tous les développeurs travaillent sur une seule branche, généralement appelée branche `trunk` or`main`.

L'idée qui sous-tend cette approche est de maintenir la base de code dans un état continuellement publiable en intégrant fréquemment les modifications du code et en s'appuyant sur des tests automatisés et une intégration continue.

Dans le développement basé sur les troncs, les développeurs valident leurs modifications dans la `main` branche plusieurs fois par jour, dans le but de procéder à de petites mises à jour incrémentielles. Cela permet des boucles de feedback rapides, réduit le risque de conflits de fusion et favorise la collaboration entre les membres de l'équipe. Cette pratique souligne l'importance d'une suite de tests bien entretenue, car elle repose sur des tests automatisés pour détecter rapidement les problèmes potentiels et garantir que la base de code reste stable et publiable.

Le développement basé sur les troncs est souvent opposé au développement basé sur les fonctionnalités (également appelé branchement de fonctionnalités ou développement piloté par les fonctionnalités), où chaque nouvelle fonctionnalité ou correction de bogue est développée dans sa propre branche dédiée, distincte de la branche principale. Le choix entre le développement basé sur les troncs et le développement basé sur les fonctionnalités dépend de facteurs tels que la taille de l'équipe, les exigences du projet et l'équilibre souhaité entre la collaboration, la fréquence d'intégration et la gestion des versions.

Pour plus d'informations sur la stratégie de branchement Trunk, consultez les ressources suivantes :

- Mettre en œuvre une stratégie de branchement principal pour les DevOps environnements multi-comptes (directivesAWS prescriptives)
- Introduction au développement basé sur les troncs (site Web de développement basé sur les troncs)

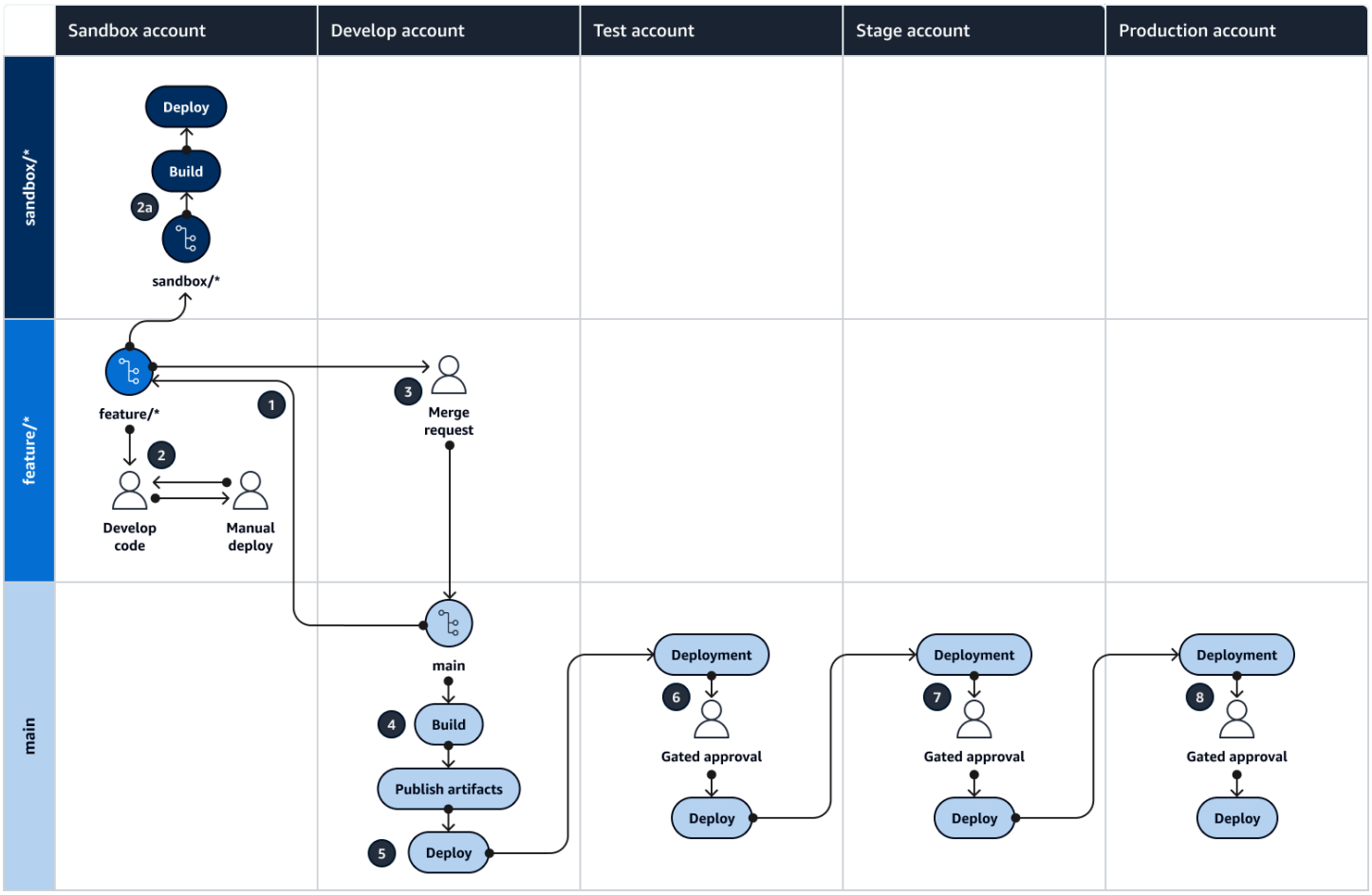
Rubriques de cette section :

- Aperçu visuel de la stratégie Trunk
- Stratégie Branches in a Trunk
- Avantages et inconvénients de la stratégie Trunk

Aperçu visuel de la stratégie Trunk

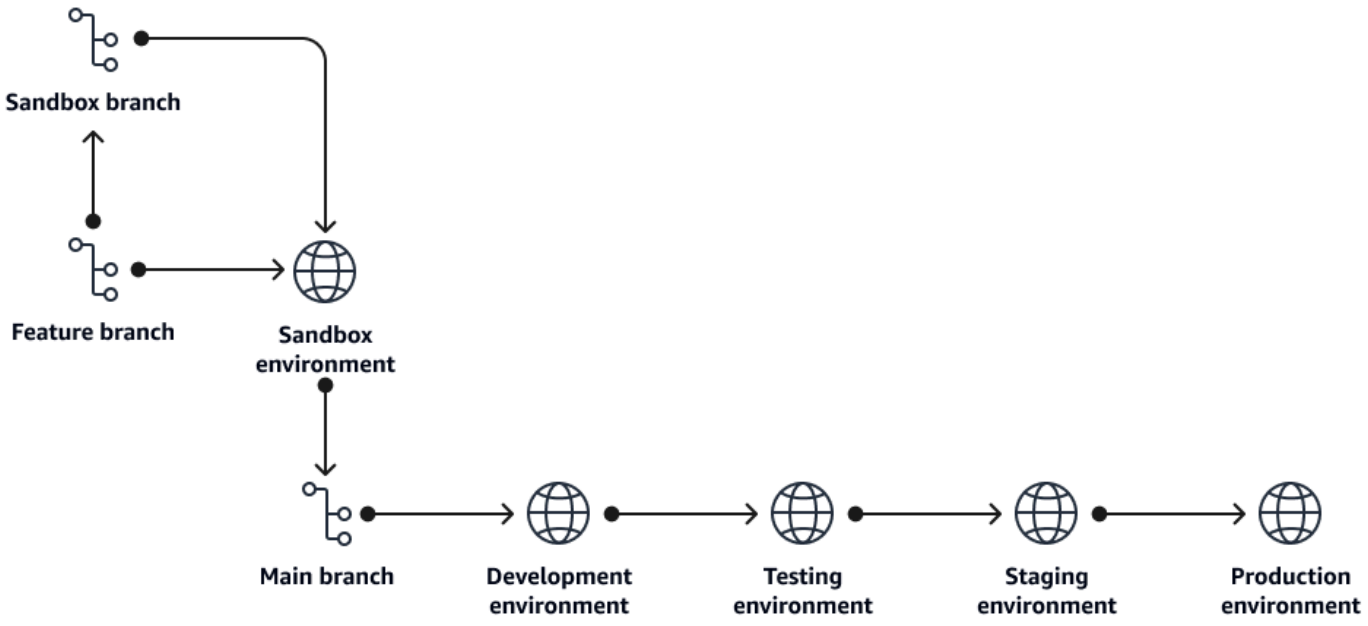
Le schéma suivant peut être utilisé comme un carré de Punnett (Wikipedia) pour comprendre la stratégie de branchement du tronc. Alignez les branches sur l'axe vertical avec les AWS environnements sur l'axe horizontal pour déterminer les actions à effectuer dans chaque scénario.

Les chiffres encerclés vous guident dans la séquence d'actions représentée dans le diagramme. Ce diagramme montre le flux de travail de développement d'une stratégie de branchement Trunk, depuis une `feature` branche dans l'environnement sandbox jusqu'à la version de production de la `main` branche. Pour plus d'informations sur les activités qui se produisent dans chaque environnement, consultez la section DevOps environnements de ce guide.



Stratégie Branches in a Trunk

Une stratégie de branchement Trunk comporte généralement les branches suivantes.



branche de fonctionnalités

Vous développez des fonctionnalités ou créez un correctif dans une `feature` branche. Pour créer une `feature` branche, vous devez la `main` quitter. Les développeurs itèrent, valident et testent le code dans une `feature` branche. Lorsqu'une fonctionnalité est terminée, le développeur en fait la promotion. Il n'y a que deux voies à suivre à partir d'une `feature` branche :

- Fusionner dans la `sandbox` branche
- Créez une demande de fusion dans la `main` succursale

Convention de dénomination :	`feature/<story number>_<developer initials>_<descriptor>`
Exemple de convention de dénomination :	`feature/123456_MS_Implement_Feature_A`

branche sandbox

Cette branche est une branche principale non standard, mais elle est utile pour le développement de pipelines CI/CD. La `sandbox` branche est principalement utilisée aux fins suivantes :

- Effectuez un déploiement complet dans l'environnement sandbox à l'aide des pipelines CI/CD
- Développez et testez un pipeline avant de soumettre des demandes de fusion pour des tests complets dans un environnement inférieur, tel que le développement ou les tests.

`Sandbox`les succursales sont de nature temporaire et sont destinées à être éphémères. Ils doivent être supprimés une fois les tests spécifiques terminés.

Convention de dénomination :	`sandbox/<story number>_<developer initials>_<descriptor>`
Exemple de convention de dénomination :	`sandbox/123456_MS_Test_Pipeline_Deploy`

branche principale

La `main` branche représente toujours le code en cours d'exécution en production. Le code est dérivé`main`, développé, puis fusionné à `main` nouveau vers. Les déploiements depuis `main` peuvent cibler n'importe quel environnement. Pour éviter toute suppression, activez la protection de branche pour la `main` branche.

Convention de dénomination :	`main`

branche hotfix

Il n'existe aucune `hotfix` branche dédiée dans un flux de travail basé sur des troncs. Les correctifs utilisent des `feature` branches.

Avantages et inconvénients de la stratégie Trunk

La stratégie de branchement Trunk convient parfaitement aux petites équipes de développement matures dotées de solides compétences en communication. Cela fonctionne également bien si vous disposez de versions continues et continues des fonctionnalités de l'application. Il n'est pas adapté si vos équipes de développement sont nombreuses ou fragmentées ou si vous avez prévu de publier de nombreuses fonctionnalités. Des conflits de fusion se produiront dans ce modèle. Sachez

donc que la résolution des conflits de fusion est une compétence clé. Tous les membres de l'équipe doivent être formés en conséquence.

Avantages

Le développement basé sur le tronc offre plusieurs avantages qui peuvent améliorer le processus de développement, rationaliser la collaboration et améliorer la qualité globale du logiciel. Voici quelques-uns des principaux avantages :

- Des boucles de feedback plus rapides : avec le développement basé sur les troncs, les développeurs intègrent fréquemment leurs modifications de code, souvent plusieurs fois par jour. Cela permet un feedback plus rapide concernant les problèmes potentiels et aide les développeurs à identifier et à résoudre les problèmes plus rapidement qu'ils ne le feraient dans un modèle de développement basé sur les fonctionnalités.
- Réduction des conflits de fusion : dans le développement basé sur les troncs, le risque de conflits de fusion importants et complexes est minimisé car les modifications sont intégrées en permanence. Cela permet de maintenir une base de code plus propre et de réduire le temps passé à résoudre les conflits. La résolution des conflits peut être à la fois chronophage et source d'erreurs dans le développement basé sur les fonctionnalités.
- Collaboration améliorée — Le développement basé sur Trunk encourage les développeurs à travailler ensemble sur la même branche, ce qui favorise une meilleure communication et une meilleure collaboration au sein de l'équipe. Cela peut accélérer la résolution des problèmes et renforcer la cohésion de l'équipe.
- Révisions de code simplifiées — Les modifications de code étant moins importantes et plus fréquentes dans le développement basé sur des troncs, il peut être plus facile de mener des révisions de code approfondies. Les modifications mineures sont généralement plus faciles à comprendre et à examiner, ce qui permet d'identifier plus efficacement les problèmes potentiels et les améliorations.
- Intégration et livraison continues — Le développement basé sur le tronc soutient les principes de l'intégration continue et de la livraison continue (CI/CD). En maintenant la base de code dans un état publiable et en intégrant fréquemment les modifications, les équipes peuvent adopter plus facilement les pratiques CI/CD, ce qui se traduit par des cycles de déploiement plus rapides et une amélioration de la qualité logicielle.
- Qualité de code améliorée — Grâce à des intégrations, des tests et des révisions de code fréquents, le développement basé sur les troncs peut contribuer à améliorer la qualité globale du

code. Les développeurs peuvent détecter et résoudre les problèmes plus rapidement, réduisant ainsi le risque d'accumulation de dettes techniques au fil du temps.

- Stratégie de branchement simplifiée — Le développement basé sur les troncs simplifie la stratégie de branchement en réduisant le nombre de succursales à longue durée de vie. Cela peut faciliter la gestion et la maintenance de la base de code, en particulier pour les grands projets ou les équipes.

Inconvénients

Le développement basé sur le tronc présente certains inconvénients, qui peuvent avoir un impact sur le processus de développement et la dynamique de l'équipe. Voici quelques inconvénients notables :

- Isolement limité — Comme tous les développeurs travaillent sur la même branche, leurs modifications sont immédiatement visibles par tous les membres de l'équipe. Cela peut entraîner des interférences ou des conflits, provoquer des effets secondaires imprévus ou endommager le bâtiment. En revanche, le développement basé sur les fonctionnalités isole mieux les modifications afin que les développeurs puissent travailler de manière plus indépendante.
- Pression accrue sur les tests — Le développement basé sur Trunk repose sur une intégration continue et des tests automatisés pour détecter rapidement les problèmes. Cependant, cette approche peut exercer une forte pression sur l'infrastructure de test et nécessite une suite de tests bien entretenue. Si les tests ne sont pas complets ou fiables, cela peut entraîner des problèmes non détectés dans la branche principale.
- Moins de contrôle sur les versions : le développement basé sur Trunk vise à maintenir la base de code dans un état continuellement publiable. Bien que cela puisse être avantageux, il n'est pas toujours adapté aux projets dont le calendrier de publication est strict ou à ceux qui nécessitent la publication conjointe de fonctionnalités spécifiques. Le développement basé sur les fonctionnalités permet de mieux contrôler le moment et la manière dont les fonctionnalités sont publiées.
- Taux de désabonnement de code — Les développeurs intégrant constamment les modifications dans la branche principale, le développement basé sur les troncs peut entraîner une augmentation du taux de désabonnement de code. Cela peut rendre difficile pour les développeurs de suivre l'état actuel de la base de code et peut être source de confusion lorsqu'ils tentent de comprendre l'effet des modifications récentes.
- Nécessite une solide culture d'équipe — Le développement basé sur Trunk exige un haut niveau de discipline, de communication et de collaboration entre les membres de l'équipe. Cela peut être difficile à maintenir, en particulier dans les grandes équipes ou lorsque vous travaillez avec des développeurs moins expérimentés avec cette approche.

- Défis d'évolutivité — À mesure que la taille de l'équipe de développement augmente, le nombre de modifications de code intégrées dans la branche principale peut augmenter rapidement. Cela peut entraîner des interruptions de compilation et des échecs de test plus fréquents, ce qui rend difficile le maintien de la base de code dans un état libérable.

GitHub Stratégie de branchement des flux

GitHub Flow est un flux de travail léger basé sur les branches développé par. GitHub GitHubFlow est basé sur l'idée de branches d'entités de courte durée qui sont fusionnées dans la branche principale lorsque la fonctionnalité est terminée et prête à être déployée. Les principes clés de GitHub Flow sont les suivants :

- Le branchement est léger : les développeurs peuvent créer des branches de fonctionnalités pour leur travail en quelques clics, ce qui améliore la capacité de collaboration et d'expérimentation sans affecter la branche principale.
- Déploiement continu — Les modifications sont déployées dès qu'elles sont fusionnées dans la branche principale, ce qui permet un retour d'information et une itération rapides.
- Demandes de fusion : les développeurs utilisent des demandes de fusion pour lancer un processus de discussion et de révision avant de fusionner leurs modifications dans la branche principale.

Pour plus d'informations sur GitHub Flow, consultez les ressources suivantes :

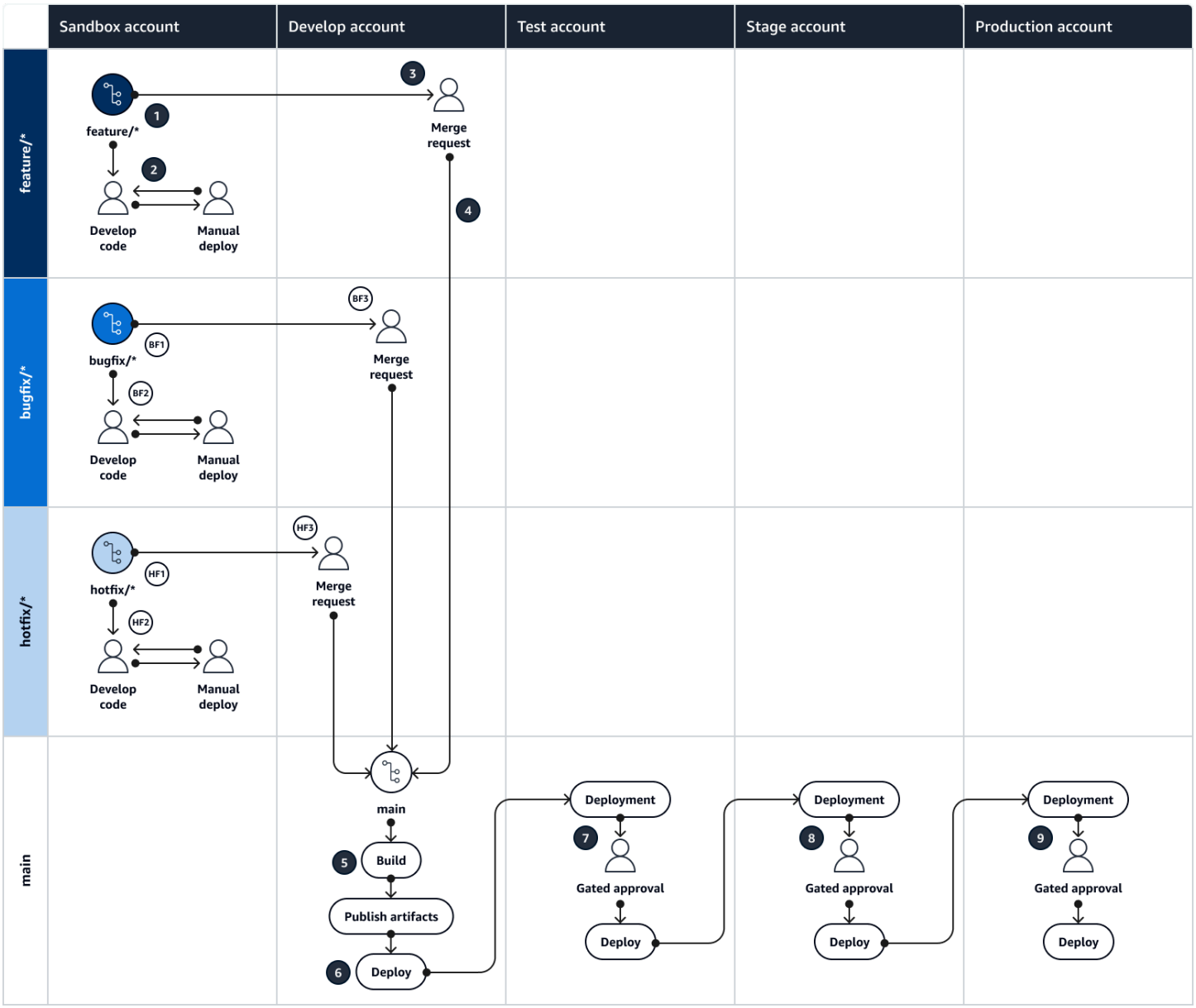
- Mettre en œuvre une stratégie de branchement GitHub Flow pour les DevOps environnements multi-comptes (directivesAWS prescriptives)
- GitHub Démarrage rapide de Flow (GitHub documentation)
- Pourquoi GitHub Flow ? (Site Web GitHub de Flow)

Rubriques de cette section :

- Aperçu visuel de la stratégie GitHub Flow
- Branches dans une stratégie GitHub Flow
- Avantages et inconvénients de la stratégie GitHub Flow

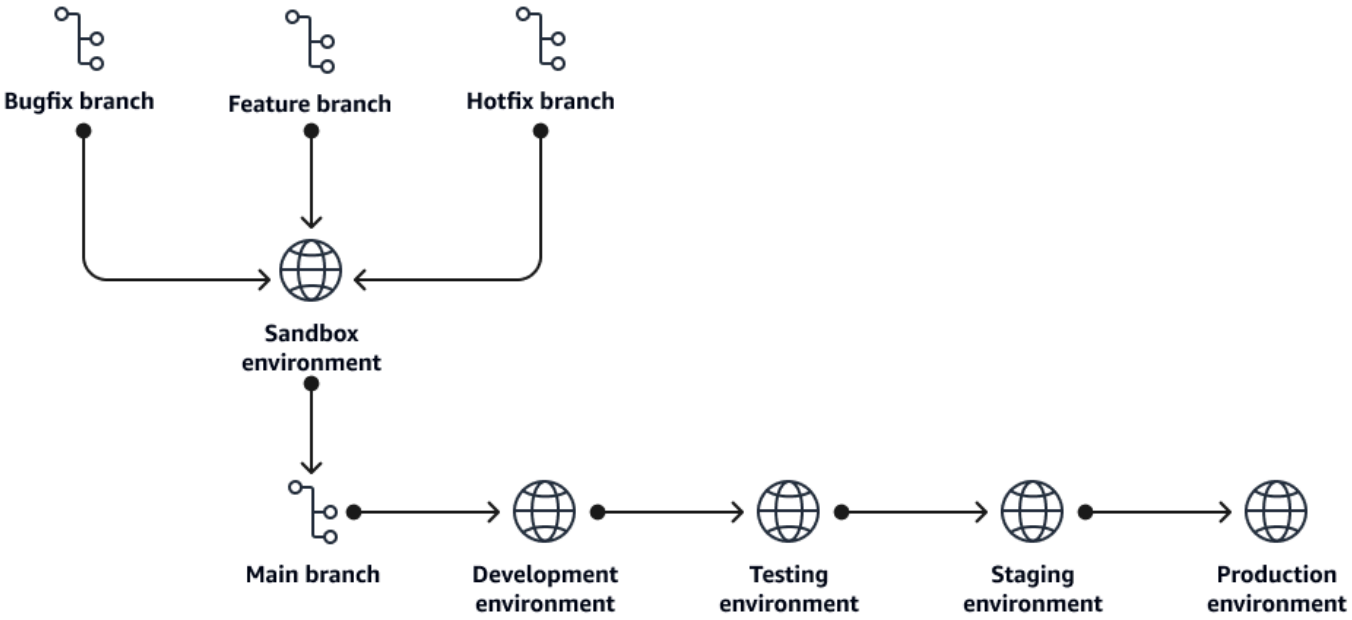
Aperçu visuel de la stratégie GitHub Flow

Le schéma suivant peut être utilisé comme un carré de Punnett pour comprendre la stratégie de branchement de GitHub Flow. Alignez les branches sur l'axe vertical avec les AWS environnements sur l'axe horizontal pour déterminer les actions à effectuer dans chaque scénario. Les chiffres encerclés vous guident dans la séquence d'actions représentée dans le diagramme. Ce diagramme montre le flux de travail de développement d'une stratégie de branchement GitHub Flow, depuis une branche fonctionnelle dans l'environnement sandbox jusqu'à la version de production de la branche principale. Pour plus d'informations sur les activités qui se produisent dans chaque environnement, consultez la section DevOps environnements de ce guide.



Branches dans une stratégie GitHub Flow

Une stratégie de branchement GitHub Flow comporte généralement les branches suivantes.



branche de fonctionnalités

Vous développez des fonctionnalités dans `feature` les succursales. Pour créer une `feature` branche, vous devez la `main` quitter. Les développeurs itèrent, valident et testent le code dans la `feature` branche. Lorsqu'une fonctionnalité est terminée, le développeur en fait la promotion en créant une demande de fusion auprès de`main`.

Convention de dénomination :	`feature/<story number>_<developer initials>_<descriptor>`
Exemple de convention de dénomination :	`feature/123456_MS_Implement_Feature_A`

branche bugfix

La `bugfix` branche est utilisée pour résoudre les problèmes. Ces branches sont dérivées de la `main` branche. Une fois que le correctif a été testé dans le bac à sable ou dans l'un des environnements inférieurs, il peut être promu vers des environnements supérieurs en le fusionnant `main` via une

demande de fusion. Il s'agit d'une convention de dénomination suggérée pour l'organisation et le suivi. Ce processus peut également être géré à l'aide d'une branche de fonctionnalités.

Convention de dénomination :	`bugfix/<ticket number>_<developer initials>_<descriptor>`
Exemple de convention de dénomination :	`bugfix/123456_MS_Fix_Problem_A`

branche hotfix

La `hotfix` branche est utilisée pour résoudre les problèmes critiques à fort impact avec un délai minimal entre le personnel de développement et le déploiement du code en production. Ces branches sont dérivées de la `main` branche. Une fois le correctif testé dans le bac à sable ou dans l'un des environnements inférieurs, il peut être promu vers des environnements supérieurs en le fusionnant `main` via une demande de fusion. Il s'agit d'une convention de dénomination suggérée pour l'organisation et le suivi. Ce processus peut également être géré à l'aide d'une branche de fonctionnalités.

Convention de dénomination :	`hotfix/<ticket number>_<developer initials>_<descriptor>`
Exemple de convention de dénomination :	`hotfix/123456_MS_Fix_Problem_A`

branche principale

La `main` branche représente toujours le code en cours d'exécution en production. Le code est fusionné dans la `main` branche à partir des `feature` branches à l'aide de demandes de fusion. Pour vous protéger contre la suppression et empêcher les développeurs d'y envoyer du code directement`main`, activez la protection de la `main` branche.

Convention de dénomination :	`main`

Avantages et inconvénients de la stratégie GitHub Flow

La stratégie de branchement de Github Flow convient parfaitement aux petites équipes de développement matures dotées de solides compétences en communication. Cette stratégie convient parfaitement aux équipes qui souhaitent mettre en œuvre une livraison continue, et elle est bien prise en charge par les moteurs CI/CD courants. GitHub Flow est léger, ne comporte pas trop de règles et est capable de soutenir des équipes qui évoluent rapidement. Il n'est pas adapté si vos équipes doivent suivre des processus de conformité ou de publication stricts. Les conflits de fusion sont courants dans ce modèle et se produiront probablement souvent. La résolution des conflits de fusion est une compétence essentielle, et vous devez former tous les membres de l'équipe en conséquence.

Avantages

GitHub Flow offre plusieurs avantages qui peuvent améliorer le processus de développement, rationaliser la collaboration et améliorer la qualité globale du logiciel. Voici quelques-uns des principaux avantages :

- Flexible et léger — GitHub Flow est un flux de travail léger et flexible qui aide les développeurs à collaborer sur des projets de développement de logiciels. Il permet une itération et une expérimentation rapides avec un minimum de complexité.
- Collaboration simplifiée — GitHub Flow fournit un processus clair et rationalisé pour gérer le développement des fonctionnalités. Il encourage de petits changements ciblés qui peuvent être rapidement revus et fusionnés, améliorant ainsi l'efficacité.
- Contrôle de version clair — Avec GitHub Flow, chaque modification est effectuée dans une branche distincte. Cela permet d'établir un historique de contrôle de version clair et traçable. Cela permet aux développeurs de suivre et de comprendre les modifications, de revenir en arrière si nécessaire et de maintenir une base de code fiable.
- Intégration continue fluide — GitHub Flow s'intègre aux outils d'intégration continue. La création de pull requests peut lancer des processus de test et de déploiement automatisés. Les outils CI vous aident à tester de manière approfondie les modifications avant qu'elles ne soient fusionnées dans la `main` branche, réduisant ainsi le risque d'introduction de bogues dans la base de code.
- Feedback rapide et amélioration continue — GitHub Flow encourage une boucle de feedback rapide en promouvant des révisions de code fréquentes et des discussions par le biais de pull requests. Cela facilite la détection précoce des problèmes, favorise le partage des connaissances

entre les membres de l'équipe et conduit finalement à une meilleure qualité du code et à une meilleure collaboration au sein de l'équipe de développement.

- Annulations et annulations simplifiées : dans le cas où une modification de code introduirait un bogue ou un problème inattendu, GitHub Flow simplifie le processus d'annulation ou d'annulation de la modification. En disposant d'un historique clair des validations et des branches, il est plus facile d'identifier et d'annuler les modifications problématiques, ce qui contribue à maintenir une base de code stable et fonctionnelle.
- Courbe d'apprentissage légère — GitHub Flow peut être plus facile à apprendre et à adopter que Gitflow, en particulier pour les équipes déjà familiarisées avec Git et les concepts de contrôle de version. Sa simplicité et son modèle de branchement intuitif le rendent accessible aux développeurs de différents niveaux d'expérience, réduisant ainsi la courbe d'apprentissage associée à l'adoption de nouveaux flux de travail de développement.
- Développement continu — GitHub Flow permet aux équipes d'adopter une approche de déploiement continu en permettant le déploiement immédiat de chaque modification dès son intégration dans la `main` succursale. Ce processus rationalisé élimine les retards inutiles et garantit que les dernières mises à jour et améliorations sont rapidement mises à la disposition des utilisateurs. Cela se traduit par un cycle de développement plus agile et plus réactif.

Inconvénients

Bien que GitHub Flow présente plusieurs avantages, il est important de prendre également en compte ses inconvénients potentiels :

- Adaptation limitée aux grands projets — GitHub Flow peut ne pas être aussi adapté aux projets de grande envergure comportant des bases de code complexes et de multiples branches de fonctionnalités à long terme. Dans de tels cas, un flux de travail plus structuré, tel que Gitflow, peut permettre de mieux contrôler le développement simultané et la gestion des versions.
- Absence de structure de version officielle — GitHub Flow ne définit pas explicitement un processus de publication ou ne prend pas en charge les fonctionnalités telles que le contrôle de version, les correctifs ou les branches de maintenance. Cela peut constituer une limite pour les projets nécessitant une gestion stricte des versions ou nécessitant un support et une maintenance à long terme.
- Support limité pour la planification des versions à long terme : GitHub Flow se concentre sur les branches de fonctionnalités éphémères, qui peuvent ne pas convenir aux projets nécessitant une planification des versions à long terme, tels que ceux dotés de feuilles de route strictes ou

de dépendances étendues entre les fonctionnalités. La gestion de calendriers de publication complexes peut s'avérer difficile compte tenu des contraintes de GitHub Flow.

- Risque de conflits de fusion fréquents — Comme GitHub Flow encourage les branchements et les fusions fréquents, il est possible de rencontrer des conflits de fusion, en particulier dans les projets impliquant une forte activité de développement. La résolution de ces conflits peut prendre beaucoup de temps et nécessiter des efforts supplémentaires de la part de l'équipe de développement.
- Absence de phases de flux de travail formalisées — GitHub Flow ne définit pas de phases de développement explicites, telles que les étapes alpha, bêta ou de version candidate. Cela peut rendre plus difficile la communication de l'état actuel du projet ou du niveau de stabilité des différentes branches ou versions.
- Impact des modifications radicales : étant donné que GitHub Flow encourage la fusion fréquente des modifications dans la `main` branche, le risque d'introduire des modifications majeures affectant la stabilité de la base de code est accru. Des pratiques strictes de révision du code et de test sont essentielles pour atténuer efficacement ce risque.

Stratégie de branchement Gitflow

Gitflow est un modèle de branchement qui implique l'utilisation de plusieurs branches pour faire passer le code du développement à la production. Gitflow fonctionne bien pour les équipes qui ont des cycles de publication planifiés et qui ont besoin de définir un ensemble de fonctionnalités en tant que version. Le développement est effectué dans des branches de fonctionnalités individuelles qui sont fusionnées, avec approbation, dans une branche de développement, qui est utilisée pour l'intégration. Les fonctionnalités de cette branche sont considérées comme prêtes à être produites. Lorsque toutes les fonctionnalités planifiées sont accumulées dans la branche de développement, une branche de publication est créée pour les déploiements dans des environnements supérieurs. Cette séparation permet de mieux contrôler quelles modifications sont transférées vers tel ou tel environnement nommé selon un calendrier défini. Si nécessaire, vous pouvez accélérer ce processus pour obtenir un modèle de déploiement plus rapide.

Pour plus d'informations sur la stratégie de branchement de Gitflow, consultez les ressources suivantes :

- Mettre en œuvre une stratégie de branchement Gitflow pour les DevOps environnements multi-comptes (directives prescriptives)AWS
- Le blog original de Gitflow (article de blog de Vincent Driessen)

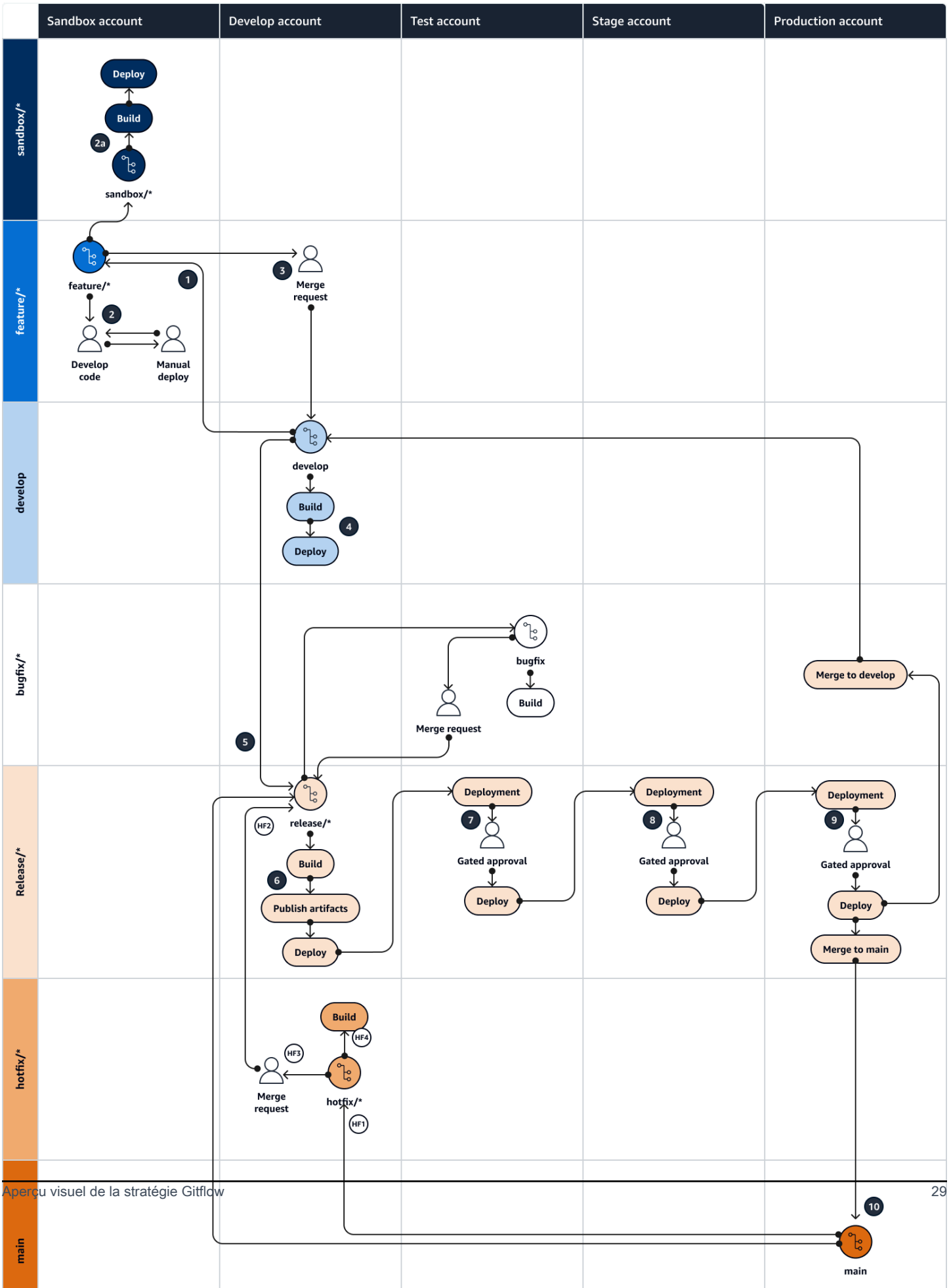
- Flux de travail Gitflow (Atlassian)

Rubriques de cette section :

- Aperçu visuel de la stratégie Gitflow
- Branches dans une stratégie Gitflow
- Avantages et inconvénients de la stratégie Gitflow

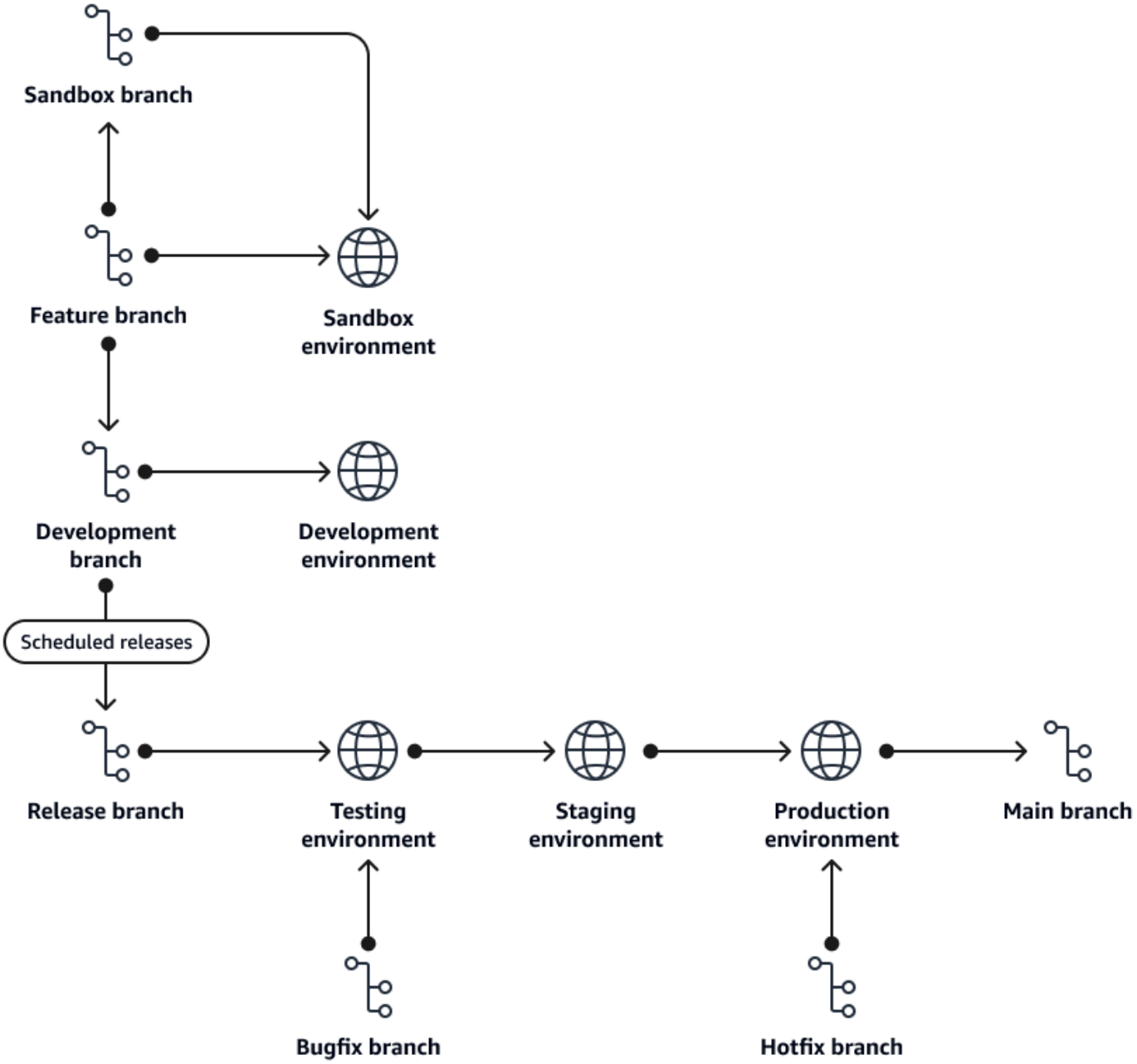
Aperçu visuel de la stratégie Gitflow

Le schéma suivant peut être utilisé comme un carré de Punnett pour comprendre la stratégie de branchement de Gitflow. Alignez les branches sur l'axe vertical avec les AWS environnements sur l'axe horizontal pour déterminer les actions à effectuer dans chaque scénario. Les chiffres encerclés vous guident dans la séquence d'actions représentée dans le diagramme. Pour plus d'informations sur les activités qui se produisent dans chaque environnement, consultez la section DevOps environnements de ce guide.


Sandbox account
Develop account
Test account
Stage account
Production account
sandbox/*
feature/*
develop
bugfix/*
Release/*
hotfix/*
main
Deploy
Build
2a
sandbox/*
feature/*
1
2
Develop code
Manual deploy
3
Merge request
develop
Build
4
Deploy
bugfix
Build
Merge request
5
Merge to develop
HF2
release/*
Build
6
Publish artifacts
Deploy
Deployment
7
Gated approval
Deploy
Deployment
8
Gated approval
Deploy
Deployment
9
Gated approval
Deploy
Merge to main
Build
HF4
HF3
Merge request
hotfix/*
HF1
10
main

Branches dans une stratégie Gitflow

Une stratégie de branchement Gitflow comporte généralement les branches suivantes.



branche de fonctionnalités

`Feature`les branches sont des branches à court terme dans lesquelles vous développez des fonctionnalités. La `feature` branche est créée en partant de la `develop` branche. Les développeurs itèrent, valident et testent le code dans la `feature` branche. Lorsque la fonctionnalité est terminée,

le développeur en fait la promotion. Il n'existe que deux voies de transfert à partir d'une branche de fonctionnalités :

- Fusionner dans la `sandbox` branche
- Créez une demande de fusion dans la `develop` succursale

Convention de dénomination :	`feature/<story number>_<developer initials>_<descriptor>`
Exemple de convention de dénomination :	`feature/123456_MS_Implement_Feature_A`

branche sandbox

La `sandbox` branche est une branche non standard à court terme pour Gitflow. Cependant, il est utile pour le développement de pipelines CI/CD. La `sandbox` branche est principalement utilisée aux fins suivantes :

- Effectuez un déploiement complet dans l'environnement sandbox en utilisant les pipelines CI/CD plutôt qu'un déploiement manuel.
- Développez et testez un pipeline avant de soumettre des demandes de fusion pour des tests complets dans un environnement inférieur, tel que le développement ou les tests.

`Sandbox`les branches sont de nature temporaire et ne sont pas destinées à durer longtemps. Ils doivent être supprimés une fois les tests spécifiques terminés.

Convention de dénomination :	`sandbox/<story number>_<developer initials>_<descriptor>`
Exemple de convention de dénomination :	`sandbox/123456_MS_Test_Pipeline_Deploy`

développer une branche

La `develop` branche est une branche durable dans laquelle les fonctionnalités sont intégrées, créées, validées et déployées dans l'environnement de développement. Toutes les `feature` branches sont fusionnées dans la `develop` branche. Les fusions avec la `develop` branche sont effectuées par le biais d'une demande de fusion qui nécessite une compilation réussie et deux approbations de développeurs. Pour empêcher la suppression, activez la protection des branches sur la `develop` branche.

Convention de dénomination :	`develop`

branche de sortie

Dans Gitflow, `release` les branches sont des branches à court terme. Ces branches sont spéciales car vous pouvez les déployer dans plusieurs environnements, en adoptant la méthodologie Build-Once, Deploy-Many. `Release`les branches peuvent cibler les environnements de test, de préparation ou de production. Une fois qu'une équipe de développement a décidé de promouvoir des fonctionnalités dans des environnements supérieurs, elle crée une nouvelle `release` branche et utilise le numéro de version incrémenté par rapport à la version précédente. Au niveau de chaque environnement, les déploiements nécessitent des approbations manuelles pour pouvoir être effectués. `Release`les branches doivent nécessiter la modification d'une demande de fusion.

Une fois que la `release` branche a été déployée en production, elle doit être fusionnée à nouveau dans les `main` branches `develop` et pour s'assurer que les corrections de bogues ou les correctifs seront intégrés dans les futurs efforts de développement.

Convention de dénomination :	`release/v{major}.{minor}`
Exemple de convention de dénomination :	`release/v1.0`

branche principale

La `main` branche est une branche à longue durée de vie qui représente toujours le code en cours d'exécution en production. Le code est automatiquement fusionné dans la `main` branche à partir d'une branche de version après un déploiement réussi depuis le pipeline de publication. Pour empêcher la suppression, activez la protection des branches sur la `main` branche.

Convention de dénomination :	`main`

branche bugfix

La `bugfix` branche est une branche à court terme qui est utilisée pour résoudre les problèmes dans les branches de publication qui n'ont pas été mises en production. Une `bugfix` branche ne doit être utilisée que pour promouvoir les corrections apportées dans `release` les branches aux environnements de test, de test ou de production. Une `bugfix` branche est toujours dérivée d'une `release` branche.

Une fois le correctif testé, il peut être promu vers la `release` branche par le biais d'une demande de fusion. Vous pouvez ensuite faire avancer la `release` branche en suivant le processus de publication standard.

Convention de dénomination :	`bugfix/<ticket>_<developer initials>_<descriptor>`
Exemple de convention de dénomination :	`bugfix/123456_MS_Fix_Problem_A`

branche hotfix

La `hotfix` succursale est une succursale à court terme utilisée pour résoudre les problèmes de production. Il ne doit être utilisé que pour promouvoir des correctifs qui doivent être accélérés pour atteindre l'environnement de production. Une `hotfix` branche est toujours issue d'une branche. `main`

Une fois le correctif testé, vous pouvez le promouvoir en production par le biais d'une demande de fusion dans la `release` branche à partir `main` de laquelle il a été créé. Pour les tests, vous pouvez ensuite faire avancer la `release` branche en suivant le processus de publication standard.

Convention de dénomination :	`hotfix/<ticket>_<developer initials>_<descriptor>`
Exemple de convention de dénomination :	`hotfix/123456_MS_Fix_Problem_A`

Avantages et inconvénients de la stratégie Gitflow

La stratégie de branchement de Gitflow convient parfaitement aux équipes plus importantes et plus distribuées qui ont des exigences strictes en matière de publication et de conformité. Gitflow contribue à un cycle de publication prévisible pour l'organisation, ce qui est souvent préféré par les grandes entreprises. Gitflow convient également aux équipes qui ont besoin de garde-fous pour terminer correctement leur cycle de vie de développement logiciel. Cela s'explique par le fait que la stratégie comporte de nombreuses opportunités d'évaluation et d'assurance qualité. Gitflow convient également aux équipes qui doivent gérer simultanément plusieurs versions de versions de production. Certains inconvénients de GitFlow sont qu'il est plus complexe que les autres modèles de branchement et qu'il nécessite un strict respect du modèle pour réussir. Gitflow ne fonctionne pas bien pour les organisations qui recherchent une livraison continue en raison de la nature rigide de la gestion des branches de publication. Les agences de lancement de Gitflow peuvent être des succursales à longue durée de vie susceptibles d'accumuler des dettes techniques si elles ne sont pas traitées correctement en temps opportun.

Avantages

Le développement basé sur Gitflow offre plusieurs avantages qui peuvent améliorer le processus de développement, rationaliser la collaboration et améliorer la qualité globale du logiciel. Voici quelques-uns des principaux avantages :

- Processus de publication prévisible — Gitflow suit un processus de publication régulier et prévisible. Il convient parfaitement aux équipes ayant des cadences de développement et de publication régulières.
- Collaboration améliorée — Gitflow encourage l'utilisation de `feature` et de `release` branches. Ces deux branches permettent aux équipes de travailler en parallèle avec un minimum de dépendance les unes par rapport aux autres.
- Bien adapté à plusieurs environnements, Gitflow utilise des `release` branches, qui peuvent avoir une durée de vie plus longue. Ces branches permettent aux équipes de cibler des versions individuelles sur une plus longue période.
- Plusieurs versions en production — Si votre équipe prend en charge plusieurs versions du logiciel en production, les `release` succursales de Gitflow prennent en charge cette exigence.
- Révisions intégrées de la qualité du code : Gitflow exige et encourage l'utilisation de révisions et d'approbations de code avant que le code ne soit promu dans un autre environnement. Ce

processus élimine les frictions entre les développeurs en imposant cette étape pour toutes les promotions de code.

- Avantages pour l'organisation — Gitflow présente également des avantages au niveau de l'organisation. Gitflow encourage l'utilisation d'un cycle de publication standard, qui aide l'organisation à comprendre et à anticiper le calendrier de publication. Comme l'entreprise sait désormais quand de nouvelles fonctionnalités peuvent être proposées, les délais sont réduits grâce à des dates de livraison fixes.

Inconvénients

Le développement basé sur Gitflow présente certains inconvénients qui peuvent avoir un impact sur le processus de développement et la dynamique de l'équipe. Voici quelques inconvénients notables :

- Complexité — Gitflow est un modèle complexe que les nouvelles équipes doivent apprendre, et vous devez respecter les règles de Gitflow pour l'utiliser avec succès.
- Déploiement continu — Gitflow ne convient pas à un modèle dans lequel de nombreux déploiements sont rapidement mis en production. Cela est dû au fait que Gitflow nécessite l'utilisation de plusieurs branches et un flux de travail strict régissant la `release` branche.
- Gestion des succursales — Gitflow utilise de nombreuses branches, dont la maintenance peut s'avérer fastidieuse. Il peut être difficile de suivre les différentes branches et de fusionner le code publié afin de maintenir les branches correctement alignées les unes avec les autres.
- Dette technique — Les versions de Gitflow étant généralement plus lentes que les autres modèles de branchement, un plus grand nombre de fonctionnalités peuvent s'accumuler au fur et à mesure de leur publication, ce qui peut entraîner une accumulation de dettes techniques.

Les équipes doivent soigneusement prendre en compte ces inconvénients lorsqu'elles décident si le développement basé sur Gitflow est la bonne approche pour leur projet.

Étapes suivantes

Ce guide explique les différences entre trois stratégies de branchement Git courantes : GitHub Flow, Gitflow et Trunk. Il décrit leurs flux de travail en détail et présente également les avantages et les inconvénients de chacun. Les étapes suivantes consistent à choisir l'un de ces flux de travail standard pour votre organisation. Pour mettre en œuvre l'une de ces stratégies de branchement, consultez les rubriques suivantes :

- Mettre en œuvre une stratégie de branchement Trunk pour les environnements multi-comptes DevOps
- Mettre en œuvre une stratégie GitHub de branchement Flow pour les environnements multi-comptes DevOps
- Mettre en œuvre une stratégie de branchement Gitflow pour les environnements multi-comptes DevOps

Si vous ne savez pas par où commencer l'utilisation de Git et des DevOps processus par votre équipe, nous vous recommandons de choisir une solution standard et de la tester. L'utilisation d'une convention de branchement standard permet à l'équipe de s'appuyer sur la documentation existante et de découvrir ce qui fonctionne le mieux pour elle.

N'ayez pas peur de modifier votre stratégie si elle ne fonctionne pas pour votre organisation ou vos équipes de développement. Les besoins et les exigences des équipes de développement peuvent évoluer au fil du temps, et il n'existe pas de solution unique et parfaite.

Ressources

Ce guide n'inclut pas de formation à Git ; toutefois, de nombreuses ressources de haute qualité sont disponibles sur Internet si vous avez besoin de cette formation. Nous vous recommandons de commencer par le site de documentation Git.

Les ressources suivantes peuvent vous aider dans votre parcours de branchement Git dans le AWS Cloud.

AWS Conseils prescriptifs

- Mettre en œuvre une stratégie de branchement Trunk pour les environnements multi-comptes DevOps
- Mettre en œuvre une stratégie GitHub de branchement Flow pour les environnements multi-comptes DevOps
- Mettre en œuvre une stratégie de branchement Gitflow pour les environnements multi-comptes DevOps

Autres AWS conseils

- AWS DevOps Orientations
- AWS Architecture de référence du pipeline de déploiement
- Présentation de DevOps
- DevOpsressources

Autres ressources

- Méthodologie d'application à douze facteurs (12factor.net)
- Git-Secrets (1) GitHub
- Gitflow
  - Le blog original de Gitflow (article de blog de Vincent Driessen)
  - Flux de travail Gitflow (Atlassian)

  - Gitflow on GitHub : Comment utiliser les flux de travail Git Flow avec GitHub Based Repos (YouTubevidéo)
  - Exemple d'initialisation de Git Flow (YouTube vidéo)
  - La branche de lancement de Gitflow du début à la fin (YouTube vidéo)
- GitHub Flux
  - GitHub Démarrage rapide de Flow (GitHub documentation)
  - Pourquoi GitHub Flow ? (Site Web GitHub de Flow)
- Tronc
  - Introduction au développement basé sur les troncs (site Web de développement basé sur les troncs)

Collaborateurs

Conception

- Mike Stephens, architecte senior des applications cloud, AWS
- Rayjan Wilson, architecte senior d'applications cloud, AWS
- Abhilash Vinod, chef d'équipe, architecte senior d'applications cloud, AWS

Révision

- Stephen DiCato, consultant principal en sécurité, AWS
- Gaurav Samudra, architecte d'applications cloud, AWS
- Steven Guggenheimer, chef d'équipe, architecte senior d'applications cloud, AWS

Rédaction technique

- Lilly AbouHarb, rédactrice technique senior, AWS

Historique du document

Le tableau suivant décrit les modifications importantes apportées à ce guide. Pour être averti des mises à jour à venir, abonnez-vous à un fil RSS.

Modification	Description	Date
Publication initiale	—	15 février 2024

AWS Glossaire des recommandations

Les termes suivants sont couramment utilisés dans les politiques, les guides et les modèles fournis par AWS les recommandations. Pour suggérer des entrées, veuillez utiliser le lien Faire un commentaire à la fin du glossaire.

Nombres

7 R

Sept politiques de migration courantes pour transférer des applications vers le cloud. Ces politiques s'appuient sur les 5 R identifiés par Gartner en 2011 et sont les suivantes :

- Refactorisation/réarchitecture : transférez une application et modifiez son architecture en tirant pleinement parti des fonctionnalités natives cloud pour améliorer l'agilité, les performances et la capacité de mise à l'échelle. Cela implique généralement le transfert du système d'exploitation et de la base de données. Exemple : migrez votre base de données Oracle sur site vers Amazon Aurora Édition compatible avec PostgreSQL.
- Replateformer (déplacer et remodeler) : transférez une application vers le cloud et introduisez un certain niveau d'optimisation pour tirer parti des fonctionnalités du cloud. Exemple : migrez votre base de données Oracle sur site vers Amazon Relational Database Service (RDSAmazon) for Oracle dans le. AWS Cloud
- Racheter (rachat) : optez pour un autre produit, généralement en passant d'une licence traditionnelle à un modèle SaaS. Exemple : migrez votre système de gestion de la relation client (CRM) vers Salesforce.com.
- Réhéberger (lift and shift) : transférez une application vers le cloud sans apporter de modifications pour tirer parti des fonctionnalités du cloud. Exemple : migrez votre base de données Oracle sur site vers Oracle sur une EC2 instance dans le AWS Cloud.
- Relocaliser (lift and shift au niveau de l'hyperviseur) : transférez l'infrastructure vers le cloud sans acheter de nouveau matériel, réécrire des applications ou modifier vos opérations existantes. Vous migrez des serveurs d'une plateforme sur site vers un service cloud pour la même plateforme. Exemple : Migrer un Microsoft Hyper-V application à AWS.
- Retenir : conservez les applications dans votre environnement source. Il peut s'agir d'applications nécessitant une refactorisation majeure, que vous souhaitez retarder, et d'applications existantes que vous souhaitez retenir, car rien ne justifie leur migration sur le plan commercial.

- Retirer : mettez hors service ou supprimez les applications dont vous n'avez plus besoin dans votre environnement source.

A

ABAC

Voir Utilisation du contrôle d'accès basé sur les attributs.

services abstraits

Consultez la section Services gérés.

ACID

Voir atomicité, cohérence, isolement, durabilité.

migration active-active

Méthode de migration de base de données dans laquelle la synchronisation des bases de données source et cible est maintenue (à l'aide d'un outil de réplication bidirectionnelle ou d'opérations d'écriture double), tandis que les deux bases de données gèrent les transactions provenant de la connexion d'applications pendant la migration. Cette méthode prend en charge la migration par petits lots contrôlés au lieu d'exiger un basculement ponctuel. Elle est plus flexible, mais demande plus de travail que la migration active-passif.

migration active-passive

Méthode de migration de base de données dans laquelle la synchronisation des bases de données source et cible est maintenue, mais seule la base de données source gère les transactions provenant de la connexion d'applications pendant que les données sont répliquées vers la base de données cible. La base de données cible n'accepte aucune transaction pendant la migration.

fonction d'agrégation

SQLFonction qui agit sur un groupe de lignes et calcule une valeur de retour unique pour le groupe. Des exemples de fonctions d'agrégation incluent `SUM` et`MAX`.

AI

Voir intelligence artificielle.

AIOps

Voir les opérations d'intelligence artificielle.

anonymisation

Processus de suppression définitive d'informations personnelles dans un ensemble de données. L'anonymisation peut contribuer à protéger la vie privée. Les données anonymisées ne sont plus considérées comme des données personnelles.

anti-modèle

Solution fréquemment utilisée pour un problème récurrent lorsque la solution est contre-productive, inefficace ou moins efficace qu'une solution alternative.

contrôle des applications

Une approche de sécurité qui permet d'utiliser uniquement des applications approuvées afin de protéger un système contre les logiciels malveillants.

portefeuille d'applications

Ensemble d'informations détaillées sur chaque application utilisée par une organisation, y compris le coût de génération et de maintenance de l'application, ainsi que sa valeur métier. Ces informations sont essentielles pour le processus de découverte et d'analyse du portefeuille et permettent d'identifier et de prioriser les applications à migrer, à moderniser et à optimiser.

intelligence artificielle (IA)

Domaine de l'informatique consacré à l'utilisation des technologies de calcul pour exécuter des fonctions cognitives généralement associées aux humains, telles que l'apprentissage, la résolution de problèmes et la reconnaissance de modèles. Pour plus d'informations, veuillez consulter Qu'est-ce que l'intelligence artificielle ?

opérations d'intelligence artificielle (AIOps)

Processus consistant à utiliser des techniques de machine learning pour résoudre les problèmes opérationnels, réduire les incidents opérationnels et les interventions humaines, mais aussi améliorer la qualité du service. Pour plus d'informations sur la façon dont AIOps elle est utilisée dans la stratégie de AWS migration, veuillez consulter le guide d'intégration des opérations.

chiffrement asymétrique

Algorithme de chiffrement qui utilise une paire de clés, une clé publique pour le chiffrement et une clé privée pour le déchiffrement. Vous pouvez partager la clé publique, car elle n'est pas utilisée pour le déchiffrement, mais l'accès à la clé privée doit être très restreint.

atomicité, cohérence, isolement, durabilité () ACID

Ensemble de propriétés logicielles garantissant la validité des données et la fiabilité opérationnelle d'une base de données, même en cas d'erreur, de panne de courant ou d'autres problèmes.

contrôle d'accès basé sur les attributs () ABAC

Pratique qui consiste à créer des autorisations détaillées en fonction des attributs de l'utilisateur, tels que le service, le poste et le nom de l'équipe. Pour plus d'informations, consultez ABAC AWS la documentation AWS Identity and Access Management (IAM).

source de données faisant autorité

Emplacement où vous stockez la version principale des données, considérée comme la source d'information la plus fiable. Vous pouvez copier les données de la source de données officielle vers d'autres emplacements à des fins de traitement ou de modification des données, par exemple en les anonymisant, en les expurgant ou en les pseudonymisant.

Zone de disponibilité

Emplacement distinct au sein d'une Région AWS qui est à l'abri des dysfonctionnements d'autres zones de disponibilité et offre une connectivité réseau peu coûteuse et de faible latence par rapport aux autres zones de disponibilité dans la même région.

AWS Cadre d'adoption du cloud (AWS CAF)

Un cadre de directives et de bonnes pratiques d' AWS pour aider les entreprises à élaborer un plan efficient et efficace pour réussir leur migration vers le Cloud. AWS CAForganise les conseils en six domaines prioritaires appelés perspectives : l'entreprise, les personnes, la gouvernance, la plateforme, la sécurité et les opérations. Les perspectives d'entreprise, de personnes et de gouvernance mettent l'accent sur les compétences et les processus métier, tandis que les perspectives relatives à la plateforme, à la sécurité et aux opérations se concentrent sur les compétences et les processus techniques. Par exemple, la perspective liée aux personnes cible les parties prenantes qui s'occupent des ressources humaines (RH), des fonctions de dotation en personnel et de la gestion des personnes. Dans cette perspective, AWS CAF fournit des conseils pour le développement du personnel, la formation et les communications afin de préparer

l'organisation à une adoption réussie du cloud. Pour plus d'informations, consultez le AWS CAFsite Web et le AWS CAFlivre blanc.

AWS Workload (AWS WQF)

Outil qui évalue les charges de travail de migration de base de données, recommande des politiques de migration et fournit des estimations de travail. AWS WQFest inclus avec AWS Schema Conversion Tool (AWS SCT). Il analyse les schémas de base de données et les objets de code, le code d'application, les dépendances et les caractéristiques de performance, et fournit des rapports d'évaluation.

B

robot

Un bot destiné à perturber ou à nuire à des individus ou à des organisations.

BCP

Consultez la section Planification de la continuité des activités.

graphique de comportement

Vue unifiée et interactive des comportements des ressources et des interactions au fil du temps. Vous pouvez utiliser un graphique de comportement avec Amazon Detective pour examiner les tentatives de connexion infructueuses, les API appels suspects et les actions similaires. Pour plus d'informations, veuillez consulter Data in a behavior graph dans la documentation Detective.

système de poids fort

Système qui stocke d'abord l'octet le plus significatif. Voir aussi endianité.

classification binaire

Processus qui prédit un résultat binaire (l'une des deux classes possibles). Par exemple, votre modèle de machine learning peut avoir besoin de prévoir des problèmes tels que « Cet e-mail est-il du spam ou non ? » ou « Ce produit est-il un livre ou une voiture ? ».

filtre de Bloom

Structure de données probabiliste et efficace en termes de mémoire qui est utilisée pour tester si un élément fait partie d'un ensemble.

déploiement bleu/vert

Stratégie de déploiement dans laquelle vous créez deux environnements distincts mais identiques. Vous exécutez la version actuelle de l'application dans un environnement (bleu) et la nouvelle version de l'application dans l'autre environnement (vert). Cette stratégie vous permet de revenir rapidement en arrière avec un impact minimal.

bot

Application logicielle qui exécute des tâches automatisées sur Internet et simule l'activité ou l'interaction humaine. Certains robots sont utiles ou bénéfiques, comme les robots d'indexation qui indexent des informations sur Internet. D'autres robots, appelés « bots malveillants », sont destinés à perturber ou à nuire à des individus ou à des organisations.

réseau d'ordinateurs zombies

Réseaux de robots infectés par des logiciels malveillants et contrôlés par une seule entité, connue sous le nom d'herder ou d'opérateur de bots. Les botnets sont le mécanisme le plus connu pour faire évoluer les bots et leur impact.

branche

Zone contenue d'un référentiel de code. La première branche créée dans un référentiel est la branche principale. Vous pouvez créer une branche à partir d'une branche existante, puis développer des fonctionnalités ou corriger des bogues dans la nouvelle branche. Une branche que vous créez pour générer une fonctionnalité est communément appelée branche de fonctionnalités. Lorsque la fonctionnalité est prête à être publiée, vous fusionnez à nouveau la branche de fonctionnalités dans la branche principale. Pour plus d'informations, consultez À propos des branches (GitHub documentation).

accès par brise-vitre

Dans des circonstances exceptionnelles et par le biais d'un processus approuvé, il s'agit d'un moyen rapide pour un utilisateur d'accéder à un accès auquel Compte AWS il n'est généralement pas autorisé. Pour plus d'informations, consultez l'indicateur Implementation break-glass procedures dans le guide Well-Architected AWS .

stratégie existante (brownfield)

L'infrastructure existante de votre environnement. Lorsque vous adoptez une stratégie existante pour une architecture système, vous concevez l'architecture en fonction des contraintes des systèmes et de l'infrastructure actuels. Si vous étendez l'infrastructure existante, vous pouvez combiner des politiques brownfield (existantes) et greenfield (inédites).

cache de tampon

Zone de mémoire dans laquelle sont stockées les données les plus fréquemment consultées.

capacité métier

Ce que fait une entreprise pour générer de la valeur (par exemple, les ventes, le service client ou le marketing). Les architectures de microservices et les décisions de développement peuvent être dictées par les capacités métier. Pour plus d'informations, veuillez consulter la section Organisation en fonction des capacités métier du livre blanc Exécution de microservices conteneurisés sur AWS.

planification de la continuité des activités (BCP)

Plan qui tient compte de l'impact potentiel d'un événement perturbateur, tel qu'une migration à grande échelle, sur les opérations, et qui permet à une entreprise de reprendre ses activités rapidement.

C

CAF

Voir le cadre d'adoption du AWS cloud.

déploiement de Canary

Diffusion lente et progressive d'une version pour les utilisateurs finaux. Lorsque vous êtes sûr, vous déployez la nouvelle version et remplacez la version actuelle dans son intégralité.

CCoE

Voir Centre d'excellence cloud.

CDC

Consultez la section Capture des données de modification.

capture de données modifiées (CDC)

Processus de suivi des modifications apportées à une source de données, telle qu'une table de base de données, et d'enregistrement des métadonnées relatives à ces modifications. Vous pouvez l'utiliser à diverses CDC fins, telles que l'audit ou la réplication des modifications dans un système cible afin de maintenir la synchronisation.

ingénierie du chaos

Introduire intentionnellement des défaillances ou des événements perturbateurs pour tester la résilience d'un système. Vous pouvez utiliser AWS Fault Injection Service (AWS FIS) pour effectuer des expériences qui stressent vos AWS charges de travail et évaluer leur réponse.

CI/CD

Voir intégration continue et livraison continue.

classification

Processus de catégorisation qui permet de générer des prédictions. Les modèles de ML pour les problèmes de classification prédisent une valeur discrète. Les valeurs discrètes se distinguent toujours les unes des autres. Par exemple, un modèle peut avoir besoin d'évaluer la présence ou non d'une voiture sur une image.

chiffrement côté client

Chiffrement des données en local, avant que le cible ne les Service AWS reçoive.

Centre d'excellence cloud (CCoE)

Une équipe multidisciplinaire qui dirige les efforts d'adoption du cloud au sein d'une organisation, notamment en développant les bonnes pratiques en matière de cloud, en mobilisant des ressources, en établissant des délais de migration et en guidant l'organisation dans le cadre de transformations à grande échelle. Pour plus d'informations, consultez les CCoEpublications sur le blog AWS Cloud Enterprise Strategy.

cloud computing

Technologie cloud généralement utilisée pour le stockage de données à distance et la gestion des appareils IoT. Le cloud computing est généralement lié à la technologie informatique de pointe.

modèle d'exploitation cloud

Dans une organisation informatique, modèle d'exploitation utilisé pour créer, faire évoluer et optimiser un ou plusieurs environnements cloud. Pour plus d'informations, consultez la section Création de votre modèle d'exploitation cloud.

étapes d'adoption du cloud

Les quatre phases que les organisations traversent généralement lorsqu'elles migrent vers AWS Cloud :

- Projet : exécution de quelques projets liés au cloud à des fins de preuve de concept et d'apprentissage
- Base : réaliser des investissements fondamentaux pour mettre à l'échelle l'adoption du cloud (par exemple, en créant une zone de destinationCCoE, en définissant un modèle opérationnel)
- Migration : migration d'applications individuelles
- Réinvention : optimisation des produits et services et innovation dans le cloud

Ces étapes ont été définies par Stephen Orban dans le billet de blog The Journey Toward Cloud-First & the Stages of Adoption sur le blog AWS Cloud Enterprise Strategy. Pour en savoir plus sur la façon dont elles sont liées à stratégie de AWS migration, veuillez consulter le guide de préparation à la migration.

CMDB

Consultez la base de données de gestion des configurations.

référentiel de code

Emplacement où le code source et d'autres ressources, comme la documentation, les exemples et les scripts, sont stockés et mis à jour par le biais de processus de contrôle de version. Les référentiels cloud courants incluent GitHub ou AWS CodeCommit. Chaque version du code est appelée branche. Dans une structure de microservice, chaque référentiel est consacré à une seule fonctionnalité. Un seul pipeline CI/CD peut utiliser plusieurs référentiels.

cache passif

Cache tampon vide, mal rempli ou contenant des données obsolètes ou non pertinentes. Cela affecte les performances, car l'instance de base de données doit lire à partir de la mémoire principale ou du disque, ce qui est plus lent que la lecture à partir du cache tampon.

données gelées

Données rarement consultées et généralement historiques. Lorsque vous interrogez ce type de données, les requêtes lentes sont généralement acceptables. Le transfert de ces données vers des niveaux ou classes de stockage moins performants et moins coûteux peut réduire les coûts.

vision par ordinateur

Domaine de l'IA qui utilise l'apprentissage automatique pour analyser et extraire des informations à partir de formats visuels tels que des images numériques et des vidéos. Par exemple, AWS Panorama propose des appareils qui ajoutent des CV aux réseaux de caméras locaux, et Amazon SageMaker fournit des algorithmes de traitement d'image pour les CV.

dérive de configuration

Pour une charge de travail, une modification de configuration par rapport à l'état attendu. Cela peut entraîner une non-conformité de la charge de travail, et cela est généralement progressif et involontaire.

base de données de gestion des configurations (CMDB)

Référentiel qui stocke et gère les informations relatives à une base de données et à son environnement informatique, y compris les composants matériels et logiciels ainsi que leurs configurations. Vous utilisez généralement les données issues d'une CMDB phase de découverte et d'analyse du portefeuille de la migration.

pack de conformité

Une collection de AWS Config règles et d'actions correctives que vous pouvez mettre en place pour personnaliser vos contrôles de conformité et de sécurité. Vous pouvez déployer un pack de conformité en tant qu'entité unique dans un Compte AWS et une région, ou au sein d'une organisation, à l'aide d'un YAML modèle. Pour de plus amples informations, veuillez consulter Conformance packs dans la AWS Config documentation.

intégration continue et livraison continue (CI/CD)

Processus d'automatisation des étapes source, de génération, de test, intermédiaire et de production du processus de publication du logiciel. CI/CD est communément décrit comme un pipeline. CI/CD peut vous aider à automatiser les processus, à améliorer la productivité, à améliorer la qualité du code et à accélérer les livraisons. Pour plus d'informations, veuillez consulter Avantages de la livraison continue. CD peut également signifier déploiement continu. Pour plus d'informations, veuillez consulter Livraison continue et déploiement continu.

CV

Voir vision par ordinateur.

D

données au repos

Données stationnaires dans votre réseau, telles que les données stockées.

classification des données

Processus permettant d'identifier et de catégoriser les données de votre réseau en fonction de leur sévérité et de leur sensibilité. Il s'agit d'un élément essentiel de toute stratégie de gestion des risques de cybersécurité, car il vous aide à déterminer les contrôles de protection et de conservation appropriés pour les données. La classification des données est une composante du pilier de sécurité du cadre AWS Well-Architected. Pour plus d'informations, veuillez consulter Classification des données.

dérive de données

Une variation significative entre les données de production et les données utilisées pour entraîner un modèle ML, ou une modification significative des données d'entrée au fil du temps. La dérive des données peut réduire la qualité, la précision et l'équité globales des prédictions des modèles ML.

données en transit

Données qui circulent activement sur votre réseau, par exemple entre les ressources du réseau.

maillage de données

Un cadre architectural qui fournit une propriété des données distribuée et décentralisée avec une gestion et une gouvernance centralisées.

minimisation des données

Le principe de collecte et de traitement des seules données strictement nécessaires. La pratique de la minimisation des données AWS Cloud peut réduire les risques liés à la confidentialité, les coûts et l'empreinte carbone de vos analyses.

périmètre de données

Ensemble de garde-fous préventifs dans votre AWS environnement qui permettent de garantir que seules les identités fiables accèdent aux ressources fiables des réseaux attendus. Pour plus d'informations, voir Création d'un périmètre de données sur AWS.

prétraitement des données

Pour transformer les données brutes en un format facile à analyser par votre modèle de ML. Le prétraitement des données peut impliquer la suppression de certaines colonnes ou lignes et le traitement des valeurs manquantes, incohérentes ou en double.

provenance des données

Le processus de suivi de l'origine et de l'historique des données tout au long de leur cycle de vie, par exemple la manière dont les données ont été générées, transmises et stockées.

personne concernée

Personne dont les données sont collectées et traitées.

entrepôt des données

Un système de gestion des données qui prend en charge les informations commerciales, telles que les analyses. Les entrepôts de données contiennent généralement de grandes quantités de données historiques et sont généralement utilisés pour les requêtes et les analyses.

langage de définition de base de données (DDL)

Instructions ou commandes permettant de créer ou de modifier la structure des tables et des objets dans une base de données.

langage de manipulation de base de données (DML)

Instructions ou commandes permettant de modifier (insérer, mettre à jour et supprimer) des informations dans une base de données.

DDL

Voir langage de définition de base de données.

ensemble profond

Sert à combiner plusieurs modèles de deep learning à des fins de prédiction. Vous pouvez utiliser des ensembles profonds pour obtenir une prévision plus précise ou pour estimer l'incertitude des prédictions.

deep learning

Un sous-champ de ML qui utilise plusieurs couches de réseaux neuronaux artificiels pour identifier le mappage entre les données d'entrée et les variables cibles d'intérêt.

defense-in-depth

Approche de la sécurité de l'information dans laquelle une série de mécanismes et de contrôles de sécurité sont judicieusement répartis sur l'ensemble d'un réseau informatique afin de protéger la confidentialité, l'intégrité et la disponibilité du réseau et des données qu'il contient. Lorsque vous adoptez cette stratégie sur AWS, vous ajoutez plusieurs contrôles à différentes couches de

la AWS Organizations structure afin de protéger les ressources. Par exemple, une defense-in-depth approche peut combiner l'authentification multifactorielle, la segmentation du réseau et le chiffrement.

administrateur délégué

Dans AWS Organizations, un service compatible peut enregistrer un compte AWS membre pour administrer les comptes de l'organisation et gérer les autorisations pour ce service. Ce compte est appelé administrateur délégué pour ce service. Pour plus d'informations et une liste des services compatibles, veuillez consulter la rubrique Services qui fonctionnent avec AWS Organizations dans la documentation AWS Organizations .

déploiement

Processus de mise à disposition d'une application, de nouvelles fonctionnalités ou de corrections de code dans l'environnement cible. Le déploiement implique la mise en œuvre de modifications dans une base de code, puis la génération et l'exécution de cette base de code dans les environnements de l'application.

environnement de développement

Voir environnement.

contrôle de détection

Contrôle de sécurité conçu pour détecter, journaliser et alerter après la survenue d'un événement. Ces contrôles constituent une deuxième ligne de défense et vous alertent en cas d'événements de sécurité qui ont contourné les contrôles préventifs en place. Pour plus d'informations, veuillez consulter la rubrique Contrôles de détection dans Implementing security controls on AWS.

cartographie de la chaîne de valeur du développement (DVSM)

Processus utilisé pour identifier et hiérarchiser les contraintes qui nuisent à la rapidité et à la qualité du cycle de vie du développement logiciel. DVSMétend le processus de cartographie de la chaîne de valeur initialement conçu pour les pratiques de production allégée. Il met l'accent sur les étapes et les équipes nécessaires pour créer et transférer de la valeur tout au long du processus de développement logiciel.

jumeau numérique

Représentation virtuelle d'un système réel, tel qu'un bâtiment, une usine, un équipement industriel ou une ligne de production. Les jumeaux numériques prennent en charge la maintenance prédictive, la surveillance à distance et l'optimisation de la production.

tableau des dimensions

Dans un schéma en étoile, table plus petite contenant les attributs de données relatifs aux données quantitatives d'une table de faits. Les attributs des tables de dimensions sont généralement des champs de texte ou des nombres discrets qui se comportent comme du texte. Ces attributs sont couramment utilisés pour la contrainte des requêtes, le filtrage et l'étiquetage des ensembles de résultats.

catastrophe

Un événement qui empêche une charge de travail ou un système d'atteindre ses objectifs commerciaux sur son site de déploiement principal. Ces événements peuvent être des catastrophes naturelles, des défaillances techniques ou le résultat d'actions humaines, telles qu'une mauvaise configuration involontaire ou une attaque de logiciel malveillant.

reprise après sinistre (DR)

La stratégie et le processus que vous utilisez pour minimiser les temps d'arrêt et les pertes de données causés par un sinistre. Pour plus d'informations, veuillez consulter Reprise après sinistre des charges de travail sur AWS : restauration dans le cloud dans le cadre AWS Well-Architected.

DML

Voir langage de manipulation de base de données.

conception axée sur le domaine

Approche visant à développer un système logiciel complexe en connectant ses composants à des domaines évolutifs, ou objectifs métier essentiels, que sert chaque composant. Ce concept a été introduit par Eric Evans dans son ouvrage Domain-Driven Design: Tackling Complexity in the Heart of Software (Boston : Addison-Wesley Professional, 2003). Pour plus d'informations sur l'utilisation du design piloté par domaine avec le modèle de figuier étrangleur, veuillez consulter Modernizing legacy Microsoft. ASP NET(ASMX) des services Web de manière incrémentielle à l'aide de conteneurs et d'Amazon API Gateway.

DR

Consultez la section Reprise après sinistre.

détection de dérive

Suivi des écarts par rapport à une configuration de référence. Par exemple, vous pouvez l'utiliser AWS CloudFormation pour détecter la dérive des ressources du système ou AWS Control Tower

pour détecter les modifications de votre zone d'atterrissage susceptibles d'affecter le respect des exigences de gouvernance.

DVSM

Voir la cartographie de la chaîne de valeur du développement.

E

EDA

Voir analyse exploratoire des données.

informatique de périphérie

Technologie qui augmente la puissance de calcul des appareils intelligents en périphérie d'un réseau IoT. Par rapport au cloud computing, l'informatique de périphérie peut réduire la latence des communications et améliorer le temps de réponse.

chiffrement

Processus informatique qui transforme des données en texte clair, lisibles par l'homme, en texte chiffré.

clé de chiffrement

Chaîne cryptographique de bits aléatoires générée par un algorithme cryptographique. La longueur des clés peut varier, et chaque clé est conçue pour être imprévisible et unique.

endianisme

Ordre selon lequel les octets sont stockés dans la mémoire de l'ordinateur. Les systèmes de poids fort stockent d'abord l'octet le plus significatif. Les systèmes de poids faible stockent d'abord l'octet le moins significatif.

point de terminaison

Voir point de terminaison de service.

service de point de terminaison

Service que vous pouvez héberger sur un cloud privé virtuel (VPC) pour le partager avec d'autres utilisateurs. Vous pouvez créer un service de point de terminaison avec AWS PrivateLink et accorder des autorisations à d'autres Comptes AWS ou à AWS Identity and Access Management (IAM) principaux. Ces comptes ou principaux peuvent se connecter à votre service de point de

terminaison de manière privée en créant des points de VPC terminaison d'interface. Pour plus d'informations, veuillez consulter Creating an endpoint service dans la documentation Amazon Virtual Private Cloud (AmazonVPC).

planification des ressources d'entreprise (ERP)

Système qui automatise et gère les principaux processus métier (tels que la comptabilité et la gestion de projet) pour une entreprise. MES

chiffrement d'enveloppe

Processus de chiffrement d'une clé de chiffrement à l'aide d'une autre clé de chiffrement. Pour plus d'informations, veuillez consulter la rubrique Envelope encryption dans la documentation AWS Key Management Service (AWS KMS).

environment

Instance d'une application en cours d'exécution. Les types d'environnement les plus courants dans le cloud computing sont les suivants :

- Environnement de développement : instance d'une application en cours d'exécution à laquelle seule l'équipe principale chargée de la maintenance de l'application peut accéder. Les environnements de développement sont utilisés pour tester les modifications avant de les promouvoir dans les environnements supérieurs. Ce type d'environnement est parfois appelé environnement de test.
- Environnements inférieurs : tous les environnements de développement d'une application, tels que ceux utilisés pour les générations et les tests initiaux.
- Environnement de production : instance d'une application en cours d'exécution à laquelle les utilisateurs finaux peuvent accéder. Dans un pipeline CI/CD, l'environnement de production est le dernier environnement de déploiement.
- Environnements supérieurs : tous les environnements accessibles aux utilisateurs autres que l'équipe de développement principale. Ils peuvent inclure un environnement de production, des environnements de préproduction et des environnements pour les tests d'acceptation par les utilisateurs.

épopée

Dans les méthodologies agiles, catégories fonctionnelles qui aident à organiser et à prioriser votre travail. Les épopées fournissent une description détaillée des exigences et des tâches d'implémentation. Par exemple, les épopées en matière de AWS CAF sécurité comprennent la gestion des identités et des accès, les contrôles de détection, la sécurité de l'infrastructure, la

protection des données et la réponse aux incidents. Pour plus d'informations sur les épopées dans la stratégie de migration AWS , veuillez consulter le guide d'implémentation du programme.

ERP

Voir Planification des ressources d'entreprise.

analyse exploratoire des données () EDA

Processus d'analyse d'un jeu de données pour comprendre ses principales caractéristiques. Vous collectez ou agrégez des données, puis vous effectuez des enquêtes initiales pour trouver des modèles, détecter des anomalies et vérifier les hypothèses. EDAest réalisée en calculant des statistiques récapitulatives et en créant des visualisations de données.

F

tableau de données

La table centrale dans un schéma en étoile. Il stocke des données quantitatives sur les opérations commerciales. Généralement, une table de faits contient deux types de colonnes : celles qui contiennent des mesures et celles qui contiennent une clé étrangère pour une table de dimensions.

échouer rapidement

Une philosophie qui utilise des tests fréquents et progressifs pour réduire le cycle de vie du développement. C'est un élément essentiel d'une approche agile.

limite d'isolation

Dans le AWS Cloud, une limite telle qu'une zone de disponibilité Région AWS, un plan de contrôle ou un plan de données qui limite l'effet d'une panne et contribue à améliorer la résilience des charges de travail. Pour de plus amples informations, veuillez consulter AWS Bordures d'isolation des pannes.

branche de fonctionnalités

Voir la succursale.

fonctionnalités

Les données d'entrée que vous utilisez pour faire une prédiction. Par exemple, dans un contexte de fabrication, les fonctionnalités peuvent être des images capturées périodiquement à partir de la ligne de fabrication.

importance des fonctionnalités

Le niveau d'importance d'une fonctionnalité pour les prédictions d'un modèle. Il s'exprime généralement sous la forme d'un score numérique qui peut être calculé à l'aide de différentes techniques, telles que la méthode Shapley Additive Explanations (SHAP) et les gradients intégrés. Pour plus d'informations, veuillez consulter Machine learning model interpretability with :AWS.

transformation de fonctionnalité

Optimiser les données pour le processus de ML, notamment en enrichissant les données avec des sources supplémentaires, en mettant à l'échelle les valeurs ou en extrayant plusieurs ensembles d'informations à partir d'un seul champ de données. Cela permet au modèle de ML de tirer parti des données. Par exemple, si vous décomposez la date « 2021-05-27 00:15:37 » en « 2021 », « mai », « jeudi » et « 15 », vous pouvez aider l'algorithme d'apprentissage à apprendre des modèles nuancés associés à différents composants de données.

FGAC

Voir le contrôle d'accès affiné.

contrôle précis des accès () FGAC

Utilisation de plusieurs conditions pour autoriser ou refuser une demande d'accès.

migration instantanée (flash-cut)

Méthode de migration de base de données qui utilise la réplication continue des données via la capture des données de modification afin de migrer les données dans les plus brefs délais, au lieu d'utiliser une approche progressive. L'objectif est de réduire au maximum les temps d'arrêt.

G

blocage géographique

Voir les restrictions géographiques.

restrictions géographiques (blocage géographique)

Dans Amazon CloudFront, option permettant d'empêcher les utilisateurs de pays spécifiques d'accéder aux distributions de contenu. Vous pouvez utiliser une liste d'autorisation ou une liste de blocage pour spécifier les pays approuvés et interdits. Pour plus d'informations, veuillez consulter

la rubrique Restriction de la distribution géographique de votre contenu dans la CloudFront documentation.

Flux de travail Gitflow

Approche dans laquelle les environnements inférieurs et supérieurs utilisent différentes branches dans un référentiel de code source. Le flux de travail Gitflow est considéré comme l'approche héritée, tandis que le flux de travail basé sur les troncs est l'approche moderne préférée.

stratégie inédite

L'absence d'infrastructures existantes dans un nouvel environnement. Lorsque vous adoptez une stratégie inédite pour une architecture système, vous pouvez sélectionner toutes les nouvelles technologies sans restriction de compatibilité avec l'infrastructure existante, également appelée brownfield. Si vous étendez l'infrastructure existante, vous pouvez combiner des politiques brownfield (existantes) et greenfield (inédites).

barrière de protection

Règle de haut niveau qui permet de régir les ressources, les politiques et la conformité au sein des unités d'organisation (OUs). Les barrières de protection préventives appliquent des politiques pour garantir l'alignement sur les normes de conformité. Elles sont mises en œuvre à l'aide de politiques de contrôle des services et de limites des IAM autorisations. Les barrières de protection de détection détectent les violations des politiques et les problèmes de conformité, et génèrent des alertes pour y remédier. Ils sont implémentés à l'aide d'Amazon AWS Config AWS Security Hub GuardDuty AWS Trusted Advisor, d'Amazon Inspector et de AWS Lambda contrôles personnalisés.

H

HA

Découvrez la haute disponibilité.

migration de base de données hétérogène

Migration de votre base de données source vers une base de données cible qui utilise un moteur de base de données différent (par exemple, Oracle vers Amazon Aurora). La migration hétérogène fait généralement partie d'un effort de réarchitecture, et la conversion du schéma peut s'avérer une tâche complexe. AWS propose AWS SCT qui facilite les conversions de schémas.

haute disponibilité (HA)

Capacité d'une charge de travail à fonctionner en continu, sans intervention, en cas de difficultés ou de catastrophes. Les systèmes HA sont conçus pour basculer automatiquement, fournir constamment des performances de haute qualité et gérer différentes charges et défaillances avec un impact minimal sur les performances.

modernisation de l'historien

Approche utilisée pour moderniser et mettre à niveau les systèmes de technologie opérationnelle (OT) afin de mieux répondre aux besoins de l'industrie manufacturière. Un historien est un type de base de données utilisé pour collecter et stocker des données provenant de diverses sources dans une usine.

migration de base de données homogène

Migration de votre base de données source vers une base de données cible qui partage le même moteur de base de données (par exemple, Microsoft SQL Server vers Amazon RDS for SQL Server). La migration homogène s'inscrit généralement dans le cadre d'un effort de réhébergement ou de replateforme. Vous pouvez utiliser les utilitaires de base de données natifs pour migrer le schéma.

données chaudes

Données fréquemment consultées, telles que les données en temps réel ou les données translationnelles récentes. Ces données nécessitent généralement un niveau ou une classe de stockage à hautes performances pour fournir des réponses rapides aux requêtes.

correctif

Solution d'urgence à un problème critique dans un environnement de production. En raison de son urgence, un correctif est généralement créé en dehors du flux de travail de DevOps publication courant.

période de soins intensifs

Immédiatement après le basculement, période pendant laquelle une équipe de migration gère et surveille les applications migrées dans le cloud afin de résoudre les problèmes éventuels. En règle générale, cette période dure de 1 à 4 jours. À la fin de la période de soins intensifs, l'équipe de migration transfère généralement la responsabilité des applications à l'équipe des opérations cloud.

I

IaC

Considérez l'infrastructure comme un code.

politique basée sur l'identité

Politique attachée à un ou plusieurs IAM principaux qui définit leurs autorisations au sein de l' AWS Cloud environnement.

application inactive

Application dont l'utilisation moyenne CPU de la mémoire se situe entre 5 et 20 % sur une période de 90 jours. Dans un projet de migration, il est courant de retirer ces applications ou de les retenir sur site.

IIoT

Voir internet industriel des objets.

infrastructure immuable

Modèle qui déploie une nouvelle infrastructure pour les charges de travail de production au lieu de mettre à jour, d'appliquer des correctifs ou de modifier l'infrastructure existante. Les infrastructures immuables sont intrinsèquement plus cohérentes, fiables et prévisibles que les infrastructures mutables. Pour plus d'informations, consultez les meilleures pratiques de déploiement à l'aide d'une infrastructure immuable dans le Cadre AWS Well-Architected.

entrant (d'entrée) VPC

Dans une architecture à AWS comptes multiples, VPC qui accepte, inspecte et achemine les connexions réseau depuis l'extérieur d'une application. L'architecture AWS de référence de sécurité recommande de configurer votre compte réseau avec les entrées, les sorties et l'inspection VPCs afin de protéger l'interface bidirectionnelle entre votre application et Internet en général.

migration incrémentielle

Stratégie de basculement dans le cadre de laquelle vous migrez votre application par petites parties au lieu d'effectuer un basculement complet unique. Par exemple, il se peut que vous ne transfériez que quelques microservices ou utilisateurs vers le nouveau système dans un

premier temps. Après avoir vérifié que tout fonctionne correctement, vous pouvez transférer progressivement des microservices ou des utilisateurs supplémentaires jusqu'à ce que vous puissiez mettre hors service votre système hérité. Cette stratégie réduit les risques associés aux migrations de grande ampleur.

Industry 4.0

Un terme introduit par Klaus Schwab en 2016 pour désigner la modernisation des processus de fabrication grâce aux avancées en matière de connectivité, de données en temps réel, d'automatisation, d'analyse et d'IA/ML.

infrastructure

Ensemble des ressources et des actifs contenus dans l'environnement d'une application.

infrastructure en tant que code (IaC)

Processus de mise en service et de gestion de l'infrastructure d'une application via un ensemble de fichiers de configuration. IaC est conçue pour vous aider à centraliser la gestion de l'infrastructure, à normaliser les ressources et à mettre à l'échelle rapidement afin que les nouveaux environnements soient reproductibles, fiables et cohérents.

internet industriel des objets (IIoT)

L'utilisation de capteurs et d'appareils connectés à Internet dans les secteurs industriels tels que la fabrication, l'énergie, l'automobile, les soins de santé, les sciences de la vie et l'agriculture. Pour plus d'informations, veuillez consulter Building an industrial Internet of Things (IIoT) digital transformation strategy.

inspection VPC

Dans une architecture à AWS comptes multiples, système centralisé VPC qui gère les inspections du trafic réseau entre VPCs (dans des identiques ou différentes Régions AWS), Internet et les réseaux sur site. L'architecture AWS de référence de sécurité recommande de configurer votre compte réseau avec les entrées, les sorties et l'inspection VPCs afin de protéger l'interface bidirectionnelle entre votre application et Internet en général.

Internet des objets (IoT)

Réseau d'objets physiques connectés dotés de capteurs ou de processeurs intégrés qui communiquent avec d'autres appareils et systèmes via Internet ou via un réseau de communication local. Pour plus d'informations, veuillez consulter la section Qu'est-ce que l'IoT ?.

interprétabilité

Caractéristique d'un modèle de machine learning qui décrit dans quelle mesure un être humain peut comprendre comment les prédictions du modèle dépendent de ses entrées. Pour plus d'informations, veuillez consulter Machine learning model interpretability with AWS.

IoT

Voir Internet des objets.

bibliothèque d'informations informatiques (ITIL)

Ensemble de bonnes pratiques pour proposer des services informatiques et les aligner sur les exigences métier. ITILconstitue la base deITSM.

Gestion des services informatiques (ITSM)

Activités associées à la conception, à la mise en œuvre, à la gestion et à la prise en charge de services informatiques d'une organisation. Pour plus d'informations sur l'intégration des opérations cloud aux ITSM outils, veuillez consulter le guide d'intégration des opérations.

ITIL

Consultez la bibliothèque d'informations informatiques.

ITSM

Voir Gestion des services informatiques.

L

contrôle d'accès basé sur les étiquettes () LBAC

Une implémentation du contrôle d'accès obligatoire (MAC) dans laquelle une valeur d'étiquette de sécurité est explicitement attribuée aux utilisateurs et aux données elles-mêmes. L'intersection entre l'étiquette de sécurité utilisateur et l'étiquette de sécurité des données détermine les lignes et les colonnes visibles par l'utilisateur.

zone de destination

Une zone de destination est un AWS environnement à comptes multiples Well-Architected évolutif et sécurisé. Il s'agit d'un point de départ à partir duquel vos entreprises peuvent rapidement lancer et déployer des charges de travail et des applications en toute confiance dans leur environnement

de sécurité et d'infrastructure. Pour plus d'informations sur les zones de destination, veuillez consulter Setting up a secure and scalable multi-account AWS environment.

migration de grande envergure

Migration de 300 serveurs ou plus.

LBAC

Voir contrôle d'accès basé sur des étiquettes.

principe de moindre privilège

Bonne pratique de sécurité qui consiste à accorder les autorisations minimales nécessaires à l'exécution d'une tâche. Pour plus d'informations, veuillez consulter Appliquer les autorisations de moindre privilège dans la IAM documentation.

lift and shift

Voir 7 Rs.

système de poids faible

Système qui stocke d'abord l'octet le moins significatif. Voir aussi endianité.

environnements inférieurs

Voir environnement.

M

machine learning (ML)

Type d'intelligence artificielle qui utilise des algorithmes et des techniques pour la reconnaissance et l'apprentissage de modèles. Le ML analyse et apprend à partir de données enregistrées, telles que les données de l'Internet des objets (IoT), pour générer un modèle statistique basé sur des modèles. Pour plus d'informations, veuillez consulter Machine Learning.

branche principale

Voir la succursale.

malware

Logiciel conçu pour compromettre la sécurité ou la confidentialité de l'ordinateur. Les logiciels malveillants peuvent perturber les systèmes informatiques, divulguer des informations sensibles

ou obtenir un accès non autorisé. Les virus, les vers, les rançongiciels, les chevaux de Troie, les logiciels espions et les enregistreurs de frappe sont des exemples de logiciels espions.

services gérés

Services AWS pour lequel AWS fonctionnent la couche d'infrastructure, le système d'exploitation et les plateformes, et vous accédez aux points de terminaison pour stocker et récupérer des données. Amazon Simple Storage Service (Amazon S3) et Amazon DynamoDB sont des exemples de services gérés. Ils sont également connus sous le nom de services abstraits.

système d'exécution de la fabrication (MES)

Un système logiciel pour le suivi, la surveillance, la documentation et le contrôle des processus de production qui convertissent les matières premières en produits finis dans l'atelier.

MAP

Voir Migration Acceleration Program.

mécanisme

Processus complet au cours duquel vous créez un outil, favorisez son adoption, puis inspectez les résultats afin de procéder aux ajustements nécessaires. Un mécanisme est un cycle qui se renforce et s'améliore lorsqu'il fonctionne. Pour plus d'informations, veuillez consulter Building with the AWS Well-Architected Framework.

compte membre

Tous les Comptes AWS autres que le compte de gestion qui font partie d'une organisation dans AWS Organizations. Un compte ne peut être membre que d'une seule organisation à la fois.

MES

Voir le système d'exécution de la fabrication.

Transport de télémétrie par file d'attente de messages () MQTT

Protocole de communication léger machine-to-machine (M2M), basé sur le modèle de publication/d'abonnement, pour les appareils IoT aux ressources limitées.

microservice

Petit service indépendant qui communique via un réseau bien défini APIs et qui est généralement détenu par de petites équipes autonomes. Par exemple, un système d'assurance peut inclure des microservices qui mappent à des capacités métier, telles que les ventes ou le marketing,

ou à des sous-domaines, tels que les achats, les réclamations ou l'analytique. Les avantages des microservices incluent l'agilité, la flexibilité de la mise à l'échelle, la facilité de déploiement, la réutilisation du code et la résilience. Pour plus d'informations, veuillez consulter Integrating microservices by using AWS serverless services.

architecture de microservices

Approche de création d'une application avec des composants indépendants qui exécutent chaque processus d'application en tant que microservice. Ces microservices communiquent via une interface bien définie à l'aide de la légèreté. APIs Chaque microservice de cette architecture peut être mis à jour, déployé et mis à l'échelle pour répondre à la demande de fonctions spécifiques d'une application. Pour de plus amples informations, veuillez consulter Implémentation de microservices sur AWS.

Migration Acceleration (MAP)

AWS Programme qui fournit un support de conseil, des formations et des services pour aider les entreprises à générer une base opérationnelle solide pour passer au cloud et pour aider à compenser le coût initial des migrations. MAPinclut une méthodologie de migration pour exécuter les migrations héritées de manière méthodique, ainsi qu'un ensemble d'outils pour automatiser et accélérer les scénarios de migration courants.

migration à grande échelle

Processus consistant à transférer la majeure partie du portefeuille d'applications vers le cloud par vagues, un plus grand nombre d'applications étant déplacées plus rapidement à chaque vague. Cette phase utilise les bonnes pratiques et les enseignements tirés des phases précédentes pour implémenter une usine de migration d'équipes, d'outils et de processus en vue de rationaliser la migration des charges de travail grâce à l'automatisation et à la livraison agile. Il s'agit de la troisième phase de la stratégie de migration AWS.

usine de migration

Équipes interfonctionnelles qui rationalisent la migration des charges de travail grâce à des approches automatisées et agiles. Les équipes de l'usine de migration comprennent généralement des responsables des opérations, des analystes métier et des propriétaires, des ingénieurs de migration, des développeurs et DevOps des professionnels travaillant dans des sprints. Entre 20 et 50 % du portefeuille d'applications d'entreprise est constitué de modèles répétés qui peuvent être optimisés par une approche d'usine. Pour plus d'informations, veuillez consulter la rubrique discussion of migration factories et le guide Cloud Migration Factory dans cet ensemble de contenus.

métadonnées de migration

Informations relatives à l'application et au serveur nécessaires pour finaliser la migration. Chaque modèle de migration nécessite un ensemble de métadonnées de migration différent. Les exemples de métadonnées de migration incluent le sous-réseau cible, le groupe de sécurité et le AWS compte.

modèle de migration

Tâche de migration reproductible qui détaille la stratégie de migration, la destination de la migration et l'application ou le service de migration utilisé. Exemple : réhéberger la migration vers Amazon EC2 avec AWS Application Migration Service.

Évaluation du portefeuille de migration (MPA)

Outil en ligne qui fournit des informations pour valider l'analyse de rentabilisation en faveur de la migration vers le. AWS Cloud MPApropose une évaluation détaillée du portefeuille (dimensionnement approprié des serveurs, tarification, TCO comparaison, analyse des coûts de migration), ainsi que la planification de la migration (analyse et collecte des données d'applications, regroupement des applications, priorisation des migrations et planification des vagues). L'MPAoutil (connexion requise) est mis gratuitement à la disposition de tous les AWS consultants et consultants APN partenaires.

Évaluation de la préparation à la migration (MRA)

Processus qui consiste à obtenir des informations sur l'état de préparation au cloud d'une entreprise, à identifier les forces et les faiblesses, ainsi qu'à élaborer un plan d'action pour combler les lacunes identifiées, à l'aide du AWS CAF. Pour plus d'informations, veuillez consulter le guide de préparation à la migration. MRAest la première phase de la stratégie de AWS migration.

stratégie de migration

Approche utilisée pour migrer une charge de travail vers AWS Cloud. Pour plus d'informations, veuillez consulter l'entrée dans ce glossaire et Mobilize your organization to accelerate large-scale migrations.

ML

Voir apprentissage automatique.

modernisation

Transformation d'une application obsolète (héritée ou monolithique) et de son infrastructure en un système agile, élastique et hautement disponible dans le cloud afin de réduire les coûts, de gagner en efficacité et de tirer parti des innovations. Pour plus d'informations, veuillez consulter Strategy for modernizing applications in the AWS Cloud.

évaluation de la préparation à la modernisation

Évaluation qui permet de déterminer si les applications d'une organisation sont prêtes à être modernisées, d'identifier les avantages, les risques et les dépendances, et qui détermine dans quelle mesure l'organisation peut prendre en charge l'état futur de ces applications. Le résultat de l'évaluation est un plan de l'architecture cible, une feuille de route détaillant les phases de développement et les étapes du processus de modernisation, ainsi qu'un plan d'action pour combler les lacunes identifiées. Pour plus d'informations, veuillez consulter Evaluating modernizing modernizing applications in the AWS Cloud.

applications monolithiques (monolithes)

Applications qui s'exécutent en tant que service unique avec des processus étroitement couplés. Les applications monolithiques ont plusieurs inconvénients. Si une fonctionnalité de l'application connaît un pic de demande, l'architecture entière doit être mise à l'échelle. L'ajout ou l'amélioration des fonctionnalités d'une application monolithique devient également plus complexe lorsque la base de code s'élargit. Pour résoudre ces problèmes, vous pouvez utiliser une architecture de microservices. Pour plus d'informations, veuillez consulter Decomposing monoliths into microservices.

MPA

Voir Évaluation du portefeuille de migration.

MQTT

Voir Message Queuing Telemetry Transport.

classification multi-classes

Processus qui permet de générer des prédictions pour plusieurs classes (prédiction d'un résultat parmi plus de deux). Par exemple, un modèle de ML peut demander « Ce produit est-il un livre, une voiture ou un téléphone ? » ou « Quelle catégorie de produits intéresse le plus ce client ? ».

infrastructure mutable

Modèle qui met à jour et modifie l'infrastructure existante pour les charges de travail de production. Pour améliorer la cohérence, la fiabilité et la prévisibilité, le AWS Well-Architected Framework recommande l'utilisation d'une infrastructure immuable comme meilleure pratique.

O

OAC

Voir Contrôle d'accès à l'origine.

OAI

Voir l'identité d'accès à l'origine.

OCM

Voir gestion du changement organisationnel.

migration hors ligne

Méthode de migration dans laquelle la charge de travail source est supprimée au cours du processus de migration. Cette méthode implique un temps d'arrêt prolongé et est généralement utilisée pour de petites charges de travail non critiques.

OI

Consultez la section Intégration des opérations.

OLA

Voir accord au niveau opérationnel.

migration en ligne

Méthode de migration dans laquelle la charge de travail source est copiée sur le système cible sans être mise hors ligne. Les applications connectées à la charge de travail peuvent continuer à fonctionner pendant la migration. Cette méthode implique un temps d'arrêt nul ou minimal et est généralement utilisée pour les charges de travail de production critiques.

OPC-États-Unis

Voir Open Process Communications - Architecture unifiée.

Communications par processus ouvert - Architecture unifiée (OPC-UA)

Un protocole de communication machine-to-machine (M2M) pour l'automatisation industrielle. OPC-UA fournit une norme d'interopérabilité avec des schémas de cryptage, d'authentification et d'autorisation des données.

accord au niveau opérationnel () OLA

Accord qui précise ce que les groupes informatiques fonctionnels s'engagent à fournir les uns aux autres, afin de prendre en charge un contrat de niveau de service (). SLA

examen de l'état de préparation opérationnelle (ORR)

Une liste de questions et de bonnes pratiques associées qui vous aident à comprendre, à évaluer, à prévenir ou à réduire l'ampleur des incidents et des défaillances possibles. Pour plus d'informations, voir Operational Readiness Reviews (ORR) dans le AWS Well-Architected Framework.

technologie opérationnelle (OT)

Systèmes matériels et logiciels qui fonctionnent avec l'environnement physique pour contrôler les opérations, les équipements et les infrastructures industriels. Dans le secteur manufacturier, l'intégration des systèmes OT et des technologies de l'information (IT) est au cœur des transformations de l'industrie 4.0.

intégration des opérations (OI)

Processus de modernisation des opérations dans le cloud, qui implique la planification de la préparation, l'automatisation et l'intégration. Pour en savoir plus, veuillez consulter le guide d'intégration des opérations.

journal de suivi d'organisation

Journal de suivi créé par AWS CloudTrail qui journalise tous les événements pour tous les Comptes AWS dans une organisation dans AWS Organizations. Ce journal de suivi est créé dans chaque Compte AWS qui fait partie de l'organisation et suit l'activité de chaque compte. Pour plus d'informations, veuillez consulter Creating a trail for an organization dans la CloudTrail documentation.

gestion du changement organisationnel (OCM)

Cadre pour gérer les transformations métier majeures et perturbatrices du point de vue des personnes, de la culture et du leadership. OCMaide les entreprises à se préparer et à effectuer la transition vers de nouveaux systèmes et de nouvelles politiques en accélérant l'adoption des

changements, en abordant les problèmes de transition et en favorisant des changements culturels et organisationnels. Dans la stratégie de AWS migration, ce cadre s'appelle accélération des personnes, en raison de la rapidité du changement requise dans les projets d'adoption du cloud. Pour plus d'informations, consultez le OCMguide.

contrôle d'accès d'origine (OAC)

Dans CloudFront, option améliorée qui permet de restreindre l'accès à votre contenu Amazon Simple Storage Service (Amazon S3). OACprend en charge tous les compartiments S3 Régions AWS, le chiffrement côté serveur avec AWS KMS (SSE-KMS) et les `DELETE` requêtes dynamiques `PUT` adressées au compartiment S3.

identité d'accès d'origine (OAI)

Dans CloudFront, option qui permet de restreindre l'accès à votre contenu Amazon S3. Lorsque vous utilisezOAI, CloudFront crée un principal auprès duquel Amazon S3 peut s'authentifier. Les principaux authentifiés peuvent accéder au contenu dans un compartiment S3 uniquement via une distribution spécifique CloudFront . Voir également OAC, qui fournit un contrôle d'accès plus précis et amélioré.

ORR

Voir l'examen de l'état de préparation opérationnelle.

DE

Voir technologie opérationnelle.

sortant (de sortie) VPC

Dans une architecture à AWS comptes multiples, VPC qui gère les connexions réseau initiées depuis une application. L'architecture AWS de référence de sécurité recommande de configurer votre compte réseau avec les entrées, les sorties et l'inspection VPCs afin de protéger l'interface bidirectionnelle entre votre application et Internet en général.

P

limite des autorisations

Politique de IAM gestion attachée IAM aux principaux pour définir les autorisations maximales que peut avoir l'utilisateur ou le rôle. Pour plus d'informations, veuillez consulter la rubrique Limites des autorisations dans la IAM documentation.

données d'identification personnelle (PII)

Informations qui, lorsqu'elles sont consultées directement ou associées à d'autres données connexes, peuvent être utilisées pour déduire raisonnablement l'identité d'une personne. PIILes exemples incluent les noms, les adresses et les coordonnées.

PII

Voir les informations personnelles identifiables.

manuel stratégique

Ensemble d'étapes prédéfinies qui capturent le travail associé aux migrations, comme la fourniture de fonctions d'opérations de base dans le cloud. Un manuel stratégique peut revêtir la forme de scripts, de runbooks automatisés ou d'un résumé des processus ou des étapes nécessaires au fonctionnement de votre environnement modernisé.

PLC

Voir contrôleur logique programmable.

PLM

Consultez la section Gestion du cycle de vie des produits.

politique

Objet qui permet de définir des autorisations (voir Politique basée sur l'identité), de spécifier les conditions d'accès (voir Politique de contrôle des services), ou de définir les autorisations maximales pour tous les comptes d'une organisation dans AWS Organizations (voir Politique de contrôle de service).

persistance polyglotte

Choix indépendant de la technologie de stockage de données d'un microservice en fonction des modèles d'accès aux données et d'autres exigences. Si vos microservices utilisent la même technologie de stockage de données, ils peuvent rencontrer des difficultés d'implémentation ou présenter des performances médiocres. Les microservices sont plus faciles à mettre en œuvre, atteignent de meilleures performances, ainsi qu'une meilleure capacité de mise à l'échelle s'ils utilisent l'entrepôt de données le mieux adapté à leurs besoins. Pour plus d'informations, veuillez consulter Enabling data persistence in microservices.

évaluation du portefeuille

Processus de découverte, d'analyse et de priorisation du portefeuille d'applications afin de planifier la migration. Pour plus d'informations, veuillez consulter Evaluating migration readiness.

predicate

Une condition de requête qui renvoie `true` ou`false`, généralement située dans une `WHERE` clause.

poussée vers le bas

Technique d'optimisation des requêtes de base de données qui filtre les données de la requête avant le transfert. Cela permet de réduire la quantité de données devant être extraites et traitées à partir de la base de données relationnelle, et d'améliorer les performances des requêtes.

contrôle préventif

Contrôle de sécurité conçu pour empêcher qu'un événement ne se produise. Ces contrôles constituent une première ligne de défense pour empêcher tout accès non autorisé ou toute modification indésirable de votre réseau. Pour plus d'informations, veuillez consulter Preventative controls dans Implementing security controls on AWS.

principal

Une entité d' AWS qui peut exécuter des actions et accéder à des ressources. Cette entité est généralement un utilisateur root pour un Compte AWS, un IAM rôle ou un utilisateur. Pour plus d'informations, consultez les termes et concepts de Principal in Roles dans la IAM documentation.

Protection de la confidentialité

Une approche de l'ingénierie des systèmes qui prend en compte la confidentialité tout au long du processus d'ingénierie.

zones hébergées privées

Conteneur qui contient des informations concernant la façon dont vous souhaitez qu'Amazon Route 53 réponde aux DNS requêtes concernant un domaine et ses sous-domaines dans un ou plusieurs d'VPCsentre eux. Pour plus d'informations, veuillez consulter Working with private hosted zones dans la documentation Route 53.

contrôle proactif

Utilisation d'un contrôle de sécurité conçu pour empêcher le déploiement de ressources non conformes. Ces contrôles analysent les ressources avant qu'elles ne soient provisionnées. Si la

ressource n'est pas conforme au contrôle, elle n'est pas provisionnée. Pour plus d'informations, consultez le guide de référence sur les contrôles dans la AWS Control Tower documentation et consultez la section Contrôles proactifs dans Implémentation des contrôles de sécurité sur AWS.

gestion du cycle de vie des produits (PLM)

Gestion des données et des processus d'un produit tout au long de son cycle de vie, depuis la conception, le développement et le lancement, en passant par la croissance et la maturité, jusqu'au déclin et au retrait.

environnement de production

Voir environnement.

contrôleur logique programmable (PLC)

Dans le secteur manufacturier, un ordinateur hautement fiable et adaptable qui surveille les machines et automatise les processus de fabrication.

pseudonymisation

Processus de remplacement des identifiants personnels dans un ensemble de données par des valeurs fictives. La pseudonymisation peut contribuer à protéger la vie privée. Les données pseudonymisées sont toujours considérées comme des données personnelles.

publier/souscrire (pub/sub)

Modèle qui permet des communications asynchrones entre les microservices afin d'améliorer l'évolutivité et la réactivité. Par exemple, dans un environnement basé sur des microservices MES, un microservice peut publier des messages d'événements sur un canal auquel d'autres microservices peuvent s'abonner. Le système peut ajouter de nouveaux microservices sans modifier le service de publication.

Q

plan de requête

Série d'étapes, telles que des instructions, utilisées pour accéder aux données d'un système de base de données SQL relationnelle.

régression du plan de requêtes

Le cas où un optimiseur de service de base de données choisit un plan moins optimal qu'avant une modification donnée de l'environnement de base de données. Cela peut être dû à des

changements en termes de statistiques, de contraintes, de paramètres d'environnement, de liaisons de paramètres de requêtes et de mises à jour du moteur de base de données.

R

RACImatrice

Voir responsable, redevable, consulté, informé (RACI).

rançongiciel

Logiciel malveillant conçu pour bloquer l'accès à un système informatique ou à des données jusqu'à ce qu'un paiement soit effectué.

RASCImatrice

Voir responsable, redevable, consulté, informé (RACI).

RCAC

Voir contrôle d'accès aux lignes et aux colonnes.

réplica en lecture

Copie d'une base de données utilisée en lecture seule. Vous pouvez acheminer les requêtes vers le réplica de lecture pour réduire la charge sur votre base de données principale.

réarchitecte

Voir 7 Rs.

objectif de point de récupération (RPO)

Durée maximale acceptable depuis le dernier point de récupération 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.

objectif de temps de récupération (RTO)

Délai maximal acceptable entre l'interruption du service et son rétablissement.

refactoriser

Voir 7 Rs.

Région

Ensemble de AWS ressources dans une zone géographique. Chaque Région AWS est isolée et indépendante des autres pour assurer la tolérance aux pannes, la stabilité et la résilience. Pour plus d'informations, voir Spécifier ce que Régions AWS votre compte peut utiliser.

régression

Technique de ML qui prédit une valeur numérique. Par exemple, pour résoudre le problème « Quel sera le prix de vente de cette maison ? », un modèle de ML pourrait utiliser un modèle de régression linéaire pour prédire le prix de vente d'une maison sur la base de faits connus à son sujet (par exemple, la superficie en mètres carrés).

réhéberger

Voir 7 Rs.

version

Dans un processus de déploiement, action visant à promouvoir les modifications apportées à un environnement de production.

déplacer

Voir 7 Rs.

replateforme

Voir 7 Rs.

rachat

Voir 7 Rs.

résilience

La capacité d'une application à résister aux perturbations ou à s'en remettre. La haute disponibilité et la reprise après sinistre sont des considérations courantes lors de la planification de la résilience dans le AWS Cloud. Pour de plus amples informations, veuillez consulter AWS Cloud Résilience.

politique basée sur les ressources

Politique attachée à une ressource, comme un compartiment Amazon S3, un point de terminaison ou une clé de chiffrement. Ce type de politique précise les principaux auxquels l'accès est autorisé, les actions prises en charge et toutes les autres conditions qui doivent être remplies.

matrice responsable, redevable, consulté et informé (RACI)

Une matrice qui définit les rôles et les responsabilités de toutes les parties impliquées dans les activités de migration et les opérations cloud. Le nom de la matrice est dérivé des types de responsabilité définis dans la matrice : responsable (R), responsable (A), consulté (C) et informé (I). Le type de support (S) est facultatif. Si vous incluez le support, la matrice est appelée RASCImatrice, et si vous l'excluez, elle est appelée RACImatrice.

contrôle réactif

Contrôle de sécurité conçu pour permettre de remédier aux événements indésirables ou aux écarts par rapport à votre référence de sécurité. Pour plus d'informations, veuillez consulter la rubrique Responsive controls dans Implementing security controls on AWS.

retain

Voir 7 Rs.

se retirer

Voir 7 Rs.

rotation

Processus où vous mettez à jour le secret périodiquement pour rendre l'accès au informations d'identification plus difficile pour un pirate informatique.

contrôle d'accès aux lignes et aux colonnes (RCAC)

L'utilisation d'SQLexpressions simples et flexibles qui ont défini des règles d'accès. RCACconsiste en des autorisations de ligne et des masques de colonnes.

RPO

Voir l'objectif du point de récupération.

RTO

Voir l'objectif en matière de temps de rétablissement.

runbook

Ensemble de procédures manuelles ou automatisées nécessaires à l'exécution d'une tâche spécifique. Elles visent généralement à rationaliser les opérations ou les procédures répétitives présentant des taux d'erreur élevés.

S

SAML2.0

Norme ouverte utilisée par de nombreux fournisseurs d'identité (IdPs). Cette fonctionnalité active l'authentification unique fédérée (SSO), permettant aux utilisateurs de se connecter à AWS Management Console ou d'appeler les AWS API opérations sans qu'il soit nécessaire de créer un identifiant IAM pour chaque membre de l'organisation. Pour plus d'informations sur la fédération SAML 2.0, veuillez consulter À propos de la fédération SAML 2.0 dans la documentation. IAM

SCADA

Voir Contrôle de supervision et acquisition de données.

SCP

Voir la politique de contrôle des services.

secret

Dans AWS Secrets Manager des informations confidentielles ou restreintes, telles qu'un mot de passe ou des informations d'identification utilisateur, que vous stockez sous forme cryptée. Il comprend la valeur secrète et ses métadonnées. La valeur secrète peut être binaire, une chaîne unique ou plusieurs chaînes. Pour plus d'informations, consultez Que contient le secret d'un Secrets Manager ? dans la documentation Secrets Manager.

contrôle de sécurité

Barrière de protection technique ou administrative qui empêche, détecte ou réduit la capacité d'un assaillant d'exploiter une vulnérabilité de sécurité. Il existe quatre principaux types de contrôles de sécurité : préventifs, détectifs, réactifs et proactifs.

renforcement de la sécurité

Processus qui consiste à réduire la surface d'attaque pour la rendre plus résistante aux attaques. Cela peut inclure des actions telles que la suppression de ressources qui ne sont plus requises, la mise en œuvre des bonnes pratiques de sécurité consistant à accorder le moindre privilège ou la désactivation de fonctionnalités inutiles dans les fichiers de configuration.

système de gestion des informations et des événements de sécurité (SIEM)

Outils et services qui associent les systèmes de gestion des informations de sécurité (SIM) et de gestion des événements de sécurité (SEM). Un SIEM système collecte, surveille et analyse les

données provenant de serveurs, de réseaux, d'appareils et d'autres sources afin de détecter les menaces et les failles de sécurité, mais aussi de générer des alertes.

automatisation des réponses de sécurité

Action prédéfinie et programmée conçue pour répondre automatiquement à un événement de sécurité ou y remédier. Ces automatisations servent de contrôles de sécurité détectifs ou réactifs qui vous aident à mettre en œuvre les meilleures pratiques en matière AWS de sécurité. Parmi les actions de réponse automatique, citons la modification d'un groupe VPC de sécurité, l'application de correctifs à une EC2 instance Amazon ou la rotation des informations d'identification.

chiffrement côté serveur

Chiffrement des données à destination, par le Service AWS qui les reçoit.

politique de contrôle des services (SCP)

Politique qui propose un contrôle centralisé des autorisations pour tous les comptes d'une organisation dans AWS Organizations. SCPsdéfinissez des barrières de protection ou des limites aux actions qu'un administrateur peut déléguer à des utilisateurs ou à des rôles. Vous pouvez utiliser SCPs comme listes d'autorisation ou de refus, pour indiquer les services ou les actions autorisés ou interdits. Pour plus d'informations, veuillez consulter la rubrique Stratégies de contrôle de service dans la AWS Organizations documentation.

point de terminaison du service

Le URL point d'entrée d'un Service AWS. Pour vous connecter par programmation au service cible, vous pouvez utiliser un point de terminaison. Pour plus d'informations, veuillez consulter la rubrique Service AWS endpoints dans Références générales AWS.

contrat de niveau de service () SLA

Accord qui précise ce qu'une équipe informatique promet de fournir à ses clients, comme le temps de disponibilité et les performances des services.

indicateur de niveau de service () SLI

Mesure d'un aspect des performances d'un service, tel que son taux d'erreur, sa disponibilité ou son débit.

objectif de niveau de service () SLO

Mesure cible qui représente l'état d'un service, tel que mesuré par un indicateur de niveau de service.

modèle de responsabilité partagée

Un modèle décrivant la responsabilité que vous partagez en matière AWS de sécurité et de conformité dans le cloud. AWS est responsable de la sécurité du cloud, alors que vous êtes responsable de la sécurité dans le cloud. Pour de plus amples informations, veuillez consulter Modèle de responsabilité partagée.

SIEM

Voir les informations de sécurité et le système de gestion des événements.

point de défaillance unique (SPOF)

Défaillance d'un seul composant critique d'une application susceptible de perturber le système.

SLA

Voir contrat de niveau de service.

SLI

Voir l'indicateur de niveau de service.

SLO

Voir l'objectif de niveau de service.

split-and-seed modèle

Modèle permettant de mettre à l'échelle et d'accélérer les projets de modernisation. Au fur et à mesure que les nouvelles fonctionnalités et les nouvelles versions de produits sont définies, l'équipe principale se divise pour créer des équipes de produit. Cela permet de mettre à l'échelle les capacités et les services de votre organisation, d'améliorer la productivité des développeurs et de favoriser une innovation rapide. Pour plus d'informations, veuillez consulter Phased approach to modernizing applications in the. AWS Cloud

SPOF

Voir point de défaillance unique.

schéma d'étoiles

Structure organisationnelle de base de données qui utilise une grande table de faits pour stocker les données transactionnelles ou mesurées et utilise une ou plusieurs tables dimensionnelles plus

petites pour stocker les attributs des données. Cette structure est conçue pour être utilisée dans un entrepôt de données ou à des fins de business intelligence.

modèle de figuier étrangleur

Approche de modernisation des systèmes monolithiques en réécrivant et en remplaçant progressivement les fonctionnalités du système jusqu'à ce que le système hérité puisse être mis hors service. Ce modèle utilise l'analogie d'un figuier de vigne qui se développe dans un arbre existant et qui finit par supplanter son hôte. Le schéma a été présenté par Martin Fowler comme un moyen de gérer les risques lors de la réécriture de systèmes monolithiques. Pour obtenir un exemple d'application de ce modèle, veuillez consulter Modernizing legacy MicrosoftASP.NET(ASMX) des services Web de manière incrémentielle à l'aide de conteneurs et d'Amazon API Gateway.

sous-réseau

Plage d'adresses IP dans votreVPC. Un sous-réseau doit se trouver dans une seule zone de disponibilité.

contrôle de supervision et acquisition de données (SCADA)

Dans le secteur manufacturier, un système qui utilise du matériel et des logiciels pour surveiller les actifs physiques et les opérations de production.

chiffrement symétrique

Algorithme de chiffrement qui utilise la même clé pour chiffrer et déchiffrer les données.

tests Synthetiques

Tester un système de manière à simuler les interactions des utilisateurs afin de détecter les problèmes potentiels ou de surveiller les performances. Vous pouvez utiliser Amazon CloudWatch Synthetics pour créer ces tests.

T

balises

Des paires clé-valeur qui jouent le rôle de métadonnées pour organiser vos AWS ressources. Les balises peuvent vous aider à gérer, identifier, organiser, rechercher et filtrer des ressources. Pour plus d'informations, veuillez consulter la rubrique Balisage de vos AWS ressources.

variable cible

La valeur que vous essayez de prédire dans le cadre du ML supervisé. Elle est également qualifiée de variable de résultat. Par exemple, dans un environnement de fabrication, la variable cible peut être un défaut du produit.

liste de tâches

Outil utilisé pour suivre les progrès dans un runbook. Liste de tâches qui contient une vue d'ensemble du runbook et une liste des tâches générales à effectuer. Pour chaque tâche générale, elle inclut le temps estimé nécessaire, le propriétaire et l'avancement.

environnement de test

Voir environnement.

entraînement

Pour fournir des données à partir desquelles votre modèle de ML peut apprendre. Les données d'entraînement doivent contenir la bonne réponse. L'algorithme d'apprentissage identifie des modèles dans les données d'entraînement, qui mettent en correspondance les attributs des données d'entrée avec la cible (la réponse que vous souhaitez prédire). Il fournit un modèle de ML qui capture ces modèles. Vous pouvez alors utiliser le modèle de ML pour obtenir des prédictions sur de nouvelles données pour lesquelles vous ne connaissez pas la cible.

passerelle de transit

Hub de transit de réseau que vous pouvez utiliser pour relier votre réseau VPCs et vos réseaux sur site. Pour plus d'informations, veuillez consulter la rubrique Qu'est-ce qu'une passerelle de transit ? dans la AWS Transit Gateway documentation.

flux de travail basé sur jonction

Approche selon laquelle les développeurs génèrent et testent des fonctionnalités localement dans une branche de fonctionnalités, puis fusionnent ces modifications dans la branche principale. La branche principale est ensuite intégrée aux environnements de développement, de préproduction et de production, de manière séquentielle.

accès sécurisé

Octroi d'autorisations à un service que vous spécifiez pour effectuer des tâches au sein de votre organisation dans AWS Organizations et dans ses comptes en votre nom. Le service de confiance crée un rôle lié au service dans chaque compte, lorsque ce rôle est nécessaire, pour effectuer des

tâches de gestion à votre place. Pour plus d'informations, consultez la section Utilisation AWS Organizations avec d'autres AWS services dans la AWS Organizations documentation.

réglage

Pour modifier certains aspects de votre processus d'entraînement afin d'améliorer la précision du modèle de ML. Par exemple, vous pouvez entraîner le modèle de ML en générant un ensemble d'étiquetage, en ajoutant des étiquettes, puis en répétant ces étapes plusieurs fois avec différents paramètres pour optimiser le modèle.

équipe de deux pizzas

Une petite DevOps équipe que vous pouvez nourrir avec deux pizzas. Une équipe de deux pizzas garantit les meilleures opportunités de collaboration possible dans le développement de logiciels.

U

incertitude

Un concept qui fait référence à des informations imprécises, incomplètes ou inconnues susceptibles de compromettre la fiabilité des modèles de ML prédictifs. Il existe deux types d'incertitude : l'incertitude épistémique est causée par des données limitées et incomplètes, alors que l'incertitude aléatoire est causée par le bruit et le caractère aléatoire inhérents aux données. Pour plus d'informations, veuillez consulter le guide Quantifying uncertainty in deep learning systems.

tasks indifférenciés

Également connu sous le nom de « levage de charges lourdes », ce travail est nécessaire pour créer et exploiter une application, mais qui n'apporte pas de valeur directe à l'utilisateur final ni d'avantage concurrentiel. Les exemples de tâches indifférenciées incluent l'approvisionnement, la maintenance et la planification des capacités.

environnements supérieurs

Voir environnement.

V

mise à vide

Opération de maintenance de base de données qui implique un nettoyage après des mises à jour incrémentielles afin de récupérer de l'espace de stockage et d'améliorer les performances.

contrôle de version

Processus et outils permettant de suivre les modifications, telles que les modifications apportées au code source dans un référentiel.

VPCpeering

Connexion entre deux VPCs qui vous permet d'acheminer le trafic à l'aide d'adresses IP privées. Pour plus d'informations, veuillez consulter la rubrique Qu'est-ce que l'VPCappairage ? dans la VPC documentation Amazon.

vulnérabilités

Défaut logiciel ou matériel qui compromet la sécurité du système.

W

cache actif

Cache tampon qui contient les données actuelles et pertinentes fréquemment consultées. L'instance de base de données peut lire à partir du cache tampon, ce qui est plus rapide que la lecture à partir de la mémoire principale ou du disque.

données chaudes

Données auxquelles l'accès est occasionnel. Lorsque vous interrogez ce type de données, des requêtes modérément lentes sont généralement acceptables.

fonction de fenêtrage

SQLFonction qui effectue un calcul sur un groupe de lignes liées d'une manière ou d'une autre à l'enregistrement en cours. Les fonctions de fenêtre sont utiles pour traiter des tâches, telles que le calcul d'une moyenne mobile ou l'accès à la valeur des lignes en fonction de la position relative de la ligne en cours.

charge de travail

Ensemble de ressources et de code qui fournit une valeur métier, par exemple une application destinée au client ou un processus de backend.

flux de travail

Groupes fonctionnels d'un projet de migration chargés d'un ensemble de tâches spécifique. Chaque flux de travail est indépendant, mais prend en charge les autres flux de travail du projet. Par exemple, le flux de travail du portefeuille est chargé de prioriser les applications, de planifier les vagues et de collecter les métadonnées de migration. Le flux de travail du portefeuille fournit ces actifs au flux de travail de migration, qui migre ensuite les serveurs et les applications.

WORM

Voir écrire une fois, lire plusieurs.

WQF

Voir le cadre AWS de qualification de la charge de travail.

écrire une fois, lire plusieurs (WORM)

Modèle de stockage qui écrit les données une seule fois et empêche leur suppression ou leur modification. Les utilisateurs autorisés peuvent lire les données autant de fois que nécessaire, mais ils ne peuvent pas les modifier. Cette infrastructure de stockage de données est considérée comme immuable.

Z

exploit zéro

Une attaque, généralement un logiciel malveillant, qui tire parti d'une vulnérabilité de type « jour zéro ».

vulnérabilité de type « jour zéro »

Une faille ou une vulnérabilité non atténuée dans un système de production. Les acteurs malveillants peuvent utiliser ce type de vulnérabilité pour attaquer le système. Les développeurs prennent souvent conscience de la vulnérabilité à la suite de l'attaque.

application zombie

Application dont l'utilisation moyenne de CPU la mémoire est inférieure à 5 %. Dans un projet de migration, il est courant de retirer ces applications.

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