Escolhendo uma estratégia de ramificação do Git para ambientes com várias contas DevOps

AWS Orientação prescritiva

AWS Orientação prescritiva: Escolhendo uma estratégia de ramificação do Git para ambientes com várias contas DevOps

Table of Contents

Escolhendo uma estratégia de ramificação do Git para ambientes com várias contas DevOps

Amazon Web Services (colaboradores)

Fevereiro de 2024 (histórico do documento)

Mudar para uma abordagem baseada em nuvem e fornecer soluções de software AWS pode ser transformador. Isso pode exigir mudanças em seu processo de ciclo de vida de desenvolvimento de software. Normalmente, vários Contas da AWS são usados durante o processo de desenvolvimento no Nuvem AWS. Escolher uma estratégia de ramificação do Git compatível para combinar com seus DevOps processos é essencial para o sucesso. Escolher a estratégia de ramificação do Git certa para sua organização ajuda você a comunicar de forma concisa DevOps os padrões e as melhores práticas entre as equipes de desenvolvimento. A ramificação do Git pode ser simples em um único ambiente, mas pode se tornar confusa quando aplicada em vários ambientes, como ambientes de sandbox, desenvolvimento, teste, preparação e produção. Ter vários ambientes aumenta a complexidade da DevOps implementação.

Este guia fornece diagramas visuais das estratégias de ramificação do Git que mostram como uma organização pode implementar um processo com várias contas. DevOps Guias visuais ajudam as equipes a entender como mesclar suas estratégias de ramificação do Git com suas práticas. DevOps Usar um modelo de ramificação padrão, como Gitflow, Flow ou Trunk, GitHub para gerenciar o repositório de código-fonte ajuda as equipes de desenvolvimento a alinhar seu trabalho. Essas equipes também podem usar recursos de treinamento padrão do Git na Internet para entender e implementar esses modelos e estratégias.

Para obter as DevOps melhores práticas AWS, consulte o DevOpsGuidance in AWS Well-Architected. Ao revisar este guia, use a devida diligência para selecionar a estratégia de ramificação certa para sua organização. Algumas estratégias podem se adequar melhor ao seu caso de uso do que outras.

Objetivos

Este guia faz parte de uma série de documentação sobre como escolher e implementar estratégias de DevOps ramificação para organizações com várias Contas da AWS. Esta série foi projetada para ajudá-lo a aplicar a estratégia que melhor atenda aos seus requisitos, metas e melhores práticas

desde o início, para agilizar sua experiência no. Nuvem AWS Este guia não contém scripts DevOps executáveis porque eles variam com base no mecanismo de integração contínua e entrega contínua (CI/CD) e nas estruturas de tecnologia que sua organização usa.

Este guia explica as diferenças entre três estratégias comuns de ramificação do Git: GitHub Flow, Gitflow e Trunk. As recomendações deste guia ajudam as equipes a identificar uma estratégia de ramificação que se alinha às metas organizacionais. Depois de revisar este guia, você poderá escolher uma estratégia de ramificação para sua organização. Depois de escolher uma estratégia, você pode usar um dos seguintes padrões para ajudá-lo a implementar essa estratégia com suas equipes de desenvolvimento:

- Implemente uma estratégia de ramificação de troncos para ambientes com várias contas DevOps
- Implemente uma estratégia GitHub de ramificação do Flow para ambientes com várias contas DevOps
- Implemente uma estratégia de ramificação do Gitflow para ambientes com várias contas DevOps

É importante observar que o que funciona para uma organização, equipe ou projeto pode não ser adequado para outras. A escolha entre as estratégias de ramificação do Git depende de vários fatores, como tamanho da equipe, requisitos do projeto e o equilíbrio desejado entre colaboração, frequência de integração e gerenciamento de lançamentos.

Usando práticas de CI/CD

AWS recomenda que você implemente a integração e a entrega contínuas (os CI/CD), which is the process of automating the software release lifecycle. It automates much or all of the manual DevOps processes that are traditionally required to get new code from development into production. A CI/CD pipeline encompasses the sandbox, development, testing, staging, and production environments. In each environment, the CI/CD pipeline provisions any infrastructure that is needed to deploy or test the code. By using CI/CD, development teams can make changes to code that are then automatically tested and deployed. CI/CD pipelines também fornecem governança e proteções para as equipes de desenvolvimento). Eles impõem consistência, padrões, melhores práticas e níveis mínimos de aceitação para aceitação e implantação de recursos. Para obter mais informações, consulte Praticando a integração contínua e a entrega contínua em AWS.

Todas as estratégias de ramificação discutidas neste guia são adequadas para CI/CD practices. The complexity of the CI/CD pipeline increases with the complexity of the branching strategy. For example, Gitflow is the most complex branching strategy discussed in this guide. CI/CD pipelines for

this strategy require more steps (such as for compliance reasons), and they must support multiple, simultaneous production releases. Using CI/CD also becomes more important as the complexity of the branching strategy increases. This is because CI/CD estabelecer barreiras e mecanismos para equipes de desenvolvimento que impeçam os desenvolvedores de contornar intencionalmente ou não o processo definido.

AWS oferece um conjunto de serviços para desenvolvedores projetados para ajudá-lo a criar pipelines de CI/CD. Por exemplo, AWS CodePipelineé um serviço de entrega contínua totalmente gerenciado que ajuda você a automatizar seus pipelines de lançamento para atualizações rápidas e confiáveis de aplicativos e infraestrutura. AWS CodeBuildcompila o código-fonte, executa testes e produz pacotes ready-to-deploy de software. Para obter mais informações, consulte Ferramentas do desenvolvedor em AWS.

Entendendo os DevOps ambientes

Para entender as estratégias de ramificação, você deve entender o propósito e as atividades que ocorrem em cada ambiente. Estabelecer vários ambientes ajuda você a separar as atividades de desenvolvimento em estágios, monitorar essas atividades e evitar o lançamento não intencional de recursos não aprovados. Você pode ter um ou mais Contas da AWS em cada ambiente.

A maioria das organizações tem vários ambientes definidos para uso. No entanto, o número de ambientes pode variar de acordo com a organização e de acordo com as políticas de desenvolvimento de software. Esta série de documentação pressupõe que você tenha os cinco ambientes comuns a seguir que abrangem seu pipeline de desenvolvimento, embora possam ser chamados por nomes diferentes:

- Sandbox — Um ambiente em que os desenvolvedores escrevem código, cometem erros e realizam trabalhos de prova de conceito.
- Desenvolvimento — Um ambiente em que os desenvolvedores integram seu código para confirmar que tudo funciona como um aplicativo único e coeso.
- Teste — Um ambiente em que equipes de controle de qualidade ou testes de aceitação ocorrem. As equipes geralmente realizam testes de desempenho ou integração nesse ambiente.
- Preparação — um ambiente de pré-produção em que você valida se o código e a infraestrutura funcionam conforme o esperado em circunstâncias equivalentes à produção. Esse ambiente está configurado para ser o mais semelhante possível ao ambiente de produção.
- Produção — Um ambiente que gerencia o tráfego de seus usuários finais e clientes.

Esta seção descreve cada ambiente em detalhes. Ele também descreve as etapas de criação, as etapas de implantação e os critérios de saída de cada ambiente, para que você possa prosseguir para o próximo. A imagem a seguir mostra esses ambientes em sequência.

Tópicos nesta seção:

- Ambiente sandbox

- Ambiente de desenvolvimento
- Ambiente de teste
- Ambiente de preparação
- Ambiente de produção

Ambiente sandbox

O ambiente sandbox é onde os desenvolvedores escrevem código, cometem erros e realizam trabalhos de prova de conceito. Você pode implantar em um ambiente sandbox a partir de uma estação de trabalho local ou por meio de um script em uma estação de trabalho local.

Acesso

Os desenvolvedores devem ter acesso total ao ambiente sandbox.

Etapas de construção

Os desenvolvedores executam manualmente a compilação em suas estações de trabalho locais quando estão prontos para implantar alterações no ambiente sandbox.

1. Use git-secrets (GitHub) para verificar informações confidenciais
2. Lint, o código-fonte
3. Crie e compile o código-fonte, se aplicável
4. Execute testes unitários
5. Realizar análise de cobertura de código
6. Executar análise estática de código
7. Crie infraestrutura como código (IaC)
8. Execute a análise de segurança do IaC
9. Extraia licenças de código aberto
10. Publique artefatos de construção

Etapas da implantação

Se você estiver usando os modelos Gitflow ou Trunk, as etapas de implantação serão iniciadas automaticamente quando uma `feature` ramificação for criada com sucesso no ambiente sandbox.

Se você estiver usando o modelo GitHub Flow, execute manualmente as seguintes etapas de implantação. A seguir estão as etapas de implantação no ambiente sandbox:

1. Baixe artefatos publicados
2. Executar o controle de versão do banco de dados
3. Execute a implantação do IaC
4. Realize testes de integração

Expectativas antes de passar para o ambiente de desenvolvimento

- Construção bem-sucedida da `feature` filial no ambiente sandbox
- Um desenvolvedor implantou e testou manualmente o recurso no ambiente sandbox

Ambiente de desenvolvimento

O ambiente de desenvolvimento é onde os desenvolvedores integram seus códigos para garantir que tudo funcione como um aplicativo coeso. No Gitflow, o ambiente de desenvolvimento contém os recursos mais recentes incluídos pela solicitação de mesclagem e está pronto para lançamento. Nas estratégias de GitHub Flow e Trunk, o ambiente de desenvolvimento é considerado um ambiente de teste, e a base de código pode ser instável e inadequada para implantação na produção.

Acesso

Atribua permissões de acordo com o princípio do menor privilégio. Privilégio mínimo é a prática recomendada de segurança para conceder as permissões mínimas necessárias para executar uma tarefa. Os desenvolvedores devem ter menos acesso ao ambiente de desenvolvimento do que ao ambiente sandbox.

Etapas de construção

A criação de uma solicitação de mesclagem para a `develop` ramificação (Gitflow) ou a `main` ramificação (Trunk ou GitHub Flow) inicia automaticamente a construção.

1. Use git-secrets (GitHub) para verificar informações confidenciais
2. Lint, o código-fonte

3. Crie e compile o código-fonte, se aplicável
4. Execute testes unitários
5. Realizar análise de cobertura de código
6. Executar análise estática de código
7. Crie IaC
8. Execute a análise de segurança do IaC
9. Extraia licenças de código aberto

Etapas da implantação

Se você estiver usando o modelo Gitflow, as etapas de implantação serão iniciadas automaticamente quando uma `develop` ramificação for criada com sucesso no ambiente de desenvolvimento. Se você estiver usando o modelo GitHub Flow ou o modelo Trunk, as etapas de implantação serão iniciadas automaticamente quando uma solicitação de mesclagem for criada na `main` ramificação. A seguir estão as etapas de implantação no ambiente de desenvolvimento:

1. Baixe os artefatos publicados nas etapas de construção
2. Executar o controle de versão do banco de dados
3. Execute a implantação do IaC
4. Execute testes de integração

Expectativas antes de passar para o ambiente de teste

- Construção e implantação bem-sucedidas da `develop` ramificação (Gitflow) ou da `main` ramificação (Trunk ou GitHub Flow) no ambiente de desenvolvimento
- O teste unitário passa em 100%
- Construção bem-sucedida do IaC
- Os artefatos de implantação foram criados com sucesso
- Um desenvolvedor realizou uma verificação manual para confirmar se o recurso está funcionando conforme o esperado

Ambiente de teste

A equipe de garantia de qualidade (QA) usa o ambiente de teste para validar os recursos. Eles aprovam as alterações depois de concluírem os testes. Quando eles aprovam, a filial passa para o próximo ambiente, a fase de preparação. No Gitflow, esse ambiente e outros acima dele só estão disponíveis para implantação a partir de `release` filiais. Uma `release` ramificação é baseada em uma `develop` ramificação que contém os recursos planejados.

Acesso

Atribua permissões de acordo com o princípio do menor privilégio. Os desenvolvedores devem ter menos acesso ao ambiente de teste do que ao ambiente de desenvolvimento. A equipe de controle de qualidade precisa de permissões suficientes para testar o recurso.

Etapas de construção

O processo de compilação nesse ambiente só é aplicável para correções de bugs ao usar a estratégia Gitflow. A criação de uma solicitação de mesclagem para a `bugfix` ramificação inicia automaticamente a construção.

1. Use git-secrets (GitHub) para verificar informações confidenciais
2. Lint, o código-fonte
3. Crie e compile o código-fonte, se aplicável
4. Execute testes unitários
5. Realizar análise de cobertura de código
6. Executar análise estática de código
7. Crie IaC
8. Execute a análise de segurança do IaC
9. Extraia licenças de código aberto

Etapas da implantação

Inicie automaticamente a implantação da `release` ramificação (Gitflow) ou da `main` ramificação (Trunk ou GitHub Flow) no ambiente de teste após a implantação no ambiente de desenvolvimento. A seguir estão as etapas de implantação no ambiente de teste:

1. Implemente a `release` ramificação (Gitflow) ou `main` ramificação (Trunk ou GitHub Flow) no ambiente de teste
2. Pausa para aprovação manual pelo pessoal designado
3. Baixe artefatos publicados
4. Executar o controle de versão do banco de dados
5. Execute a implantação do IaC
6. Execute testes de integração
7. Realize testes de desempenho
8. Aprovação de garantia de qualidade

Expectativas antes de passar para o ambiente de teste

- As equipes de desenvolvimento e controle de qualidade realizaram testes suficientes para satisfazer os requisitos da sua organização.
- A equipe de desenvolvimento resolveu todos os bugs descobertos por meio de uma `bugfix` ramificação.

Ambiente de preparação

O ambiente de preparação está configurado para ser igual ao ambiente de produção. Por exemplo, a configuração dos dados deve ser semelhante em escopo e tamanho às cargas de trabalho de produção. Use o ambiente de preparação para verificar se o código e a infraestrutura funcionam conforme o esperado. Esse ambiente também é a escolha preferida para casos de uso comercial, como visualizações prévias ou demonstrações para clientes.

Acesso

Atribua permissões de acordo com o princípio do menor privilégio. Os desenvolvedores devem ter o mesmo acesso ao ambiente de preparação que têm ao ambiente de produção.

Etapas de construção

Nenhum. Os mesmos artefatos que foram usados no ambiente de teste são reutilizados no ambiente de teste.

Etapas da implantação

Inicie automaticamente a implantação da `release` filial (Gitflow) ou da `main` filial (Trunk ou GitHub Flow) no ambiente de teste após a aprovação e a implantação no ambiente de teste. A seguir estão as etapas de implantação no ambiente de preparação:

1. Implemente a `release` ramificação (Gitflow) ou `main` ramificação (Trunk ou GitHub Flow) no ambiente de teste
2. Pausa para aprovação manual pelo pessoal designado
3. Baixe artefatos publicados
4. Executar o controle de versão do banco de dados
5. Execute a implantação do IaC
6. (Opcional) Realizar testes de integração
7. (Opcional) Execute o teste de carga
8. Obtenha a aprovação dos aprovadores necessários de desenvolvimento, controle de qualidade, produto ou negócios

Expectativas antes de passar para o ambiente de produção

- Uma versão equivalente à produção foi implantada com sucesso no ambiente de teste
- (Opcional) A integração e o teste de carga foram bem-sucedidos

Ambiente de produção

O ambiente de produção suporta o produto lançado, manipulando dados reais de clientes reais. Esse é um ambiente protegido ao qual é atribuído acesso por privilégio mínimo e o acesso elevado só deve ser permitido por meio de um processo de exceção auditado por um período limitado de tempo.

Acesso

No ambiente de produção, os desenvolvedores devem ter acesso limitado e somente para leitura no AWS Management Console. Por exemplo, os desenvolvedores devem ser capazes de acessar dados de log para day-to-day operações. Todas as versões para produção devem ser limitadas por uma etapa de aprovação antes da implantação.

Etapas de construção

Nenhum. Os mesmos artefatos que foram usados nos ambientes de teste e preparação são reutilizados no ambiente de produção.

Etapas da implantação

Inicie automaticamente a implantação da `release` filial (Gitflow) ou da `main` filial (Trunk ou GitHub Flow) no ambiente de produção após a aprovação e a implantação no ambiente de preparação. A seguir estão as etapas de implantação no ambiente de produção:

1. Implante a `release` ramificação (Gitflow) ou `main` ramificação (Trunk ou GitHub Flow) no ambiente de produção
2. Pausa para aprovação manual pelo pessoal designado
3. Baixe artefatos publicados
4. Executar o controle de versão do banco de dados
5. Execute a implantação do IaC

Melhores práticas para desenvolvimento baseado em Git

Para adotar com sucesso o desenvolvimento baseado em Git, é importante seguir um conjunto de melhores práticas que promovam a colaboração, mantenham a qualidade do código e ofereçam suporte à integração e entrega contínuas (CI/CD). Além das melhores práticas deste guia, consulte o AWS DevOps Well-Architected Guidance. A seguir estão algumas das principais práticas recomendadas para o desenvolvimento baseado em Git em: AWS

- Mantenha as mudanças pequenas e frequentes — incentive os desenvolvedores a realizar alterações ou recursos pequenos e incrementais. Isso reduz o risco de conflitos de mesclagem e facilita a identificação e a correção rápida de problemas.
- Use alternadores de recursos — Para gerenciar o lançamento de recursos incompletos ou experimentais, use alternadores de recursos ou sinalizadores de recursos. Isso ajuda você a ocultar, ativar ou desativar recursos específicos na produção sem afetar a estabilidade da ramificação principal.
- Mantenha uma suíte de testes robusta — Uma suíte de testes abrangente e bem mantida é crucial para detectar problemas precocemente e verificar se a base de código permanece estável. Invista na automação de testes e priorize a correção de quaisquer testes que falhem.
- Adote a integração contínua — Use ferramentas e práticas de integração contínua para criar, testar e integrar automaticamente as alterações de código na `develop` ramificação (Gitflow) ou `main` ramificação (Trunk ou GitHub Flow). Isso ajuda você a detectar problemas mais cedo e agiliza o processo de desenvolvimento.
- Realize análises de código — incentive as revisões de código por pares para manter a qualidade, compartilhar conhecimento e detectar possíveis problemas antes que eles sejam integrados à `main` ramificação. Use pull requests ou outras ferramentas de revisão de código para facilitar esse processo.
- Monitore e corrija compilações quebradas — Quando uma compilação falha ou os testes falham, priorize a correção do problema o mais rápido possível. Isso mantém a `develop` ramificação (Gitflow) ou `main` ramificação (Trunk ou GitHub Flow) em um estado liberável e minimiza o impacto em outros desenvolvedores.
- Comunique-se e colabore — Promova a comunicação aberta e a colaboração entre os membros da equipe. Certifique-se de que os desenvolvedores estejam cientes do trabalho contínuo e das mudanças que estão sendo feitas na base de código.

- Refatore continuamente — Refatore regularmente a base de código para melhorar sua capacidade de manutenção e reduzir a dívida técnica. Incentive os desenvolvedores a deixar o código em um estado melhor do que o encontrado.
- Use ramificações de vida curta para tarefas complexas — Para tarefas maiores ou mais complexas, use ramificações de vida curta (também conhecidas como ramificações de tarefas) para trabalhar nas alterações. No entanto, certifique-se de manter a vida útil da filial curta, normalmente menos de um dia. Mescle as alterações novamente na `develop` ramificação (Gitflow) ou `main` ramificação (Trunk ou GitHub Flow) assim que possível. Fusões e revisões menores e mais frequentes são mais fáceis de serem consumidas e processadas por uma equipe do que uma grande solicitação de mesclagem.
- Treine e apoie a equipe — Forneça treinamento e suporte para desenvolvedores que são novos no desenvolvimento baseado em Git ou que precisam de orientação para adotar suas melhores práticas.

Estratégias de ramificação do Git

Da ordem menor à mais complexa, este guia descreve detalhadamente as seguintes estratégias de ramificação baseadas em Git:

- Tronco — O desenvolvimento baseado em troncos é uma prática de desenvolvimento de software na qual todos os desenvolvedores trabalham em uma única ramificação, normalmente chamada de ramificação `trunk` ou`main`. A ideia por trás dessa abordagem é manter a base de código em um estado continuamente liberável, integrando alterações de código com frequência e confiando em testes automatizados e integração contínua.
- GitHub Fluxo — O GitHub fluxo é um fluxo de trabalho leve, baseado em ramificações, desenvolvido pela. GitHub É baseado na ideia de `feature` filiais de curta duração. Quando um recurso está completo e pronto para ser implantado, o recurso é mesclado na `main` ramificação.
- Gitflow — Com uma abordagem Gitflow, o desenvolvimento é concluído em ramificações de recursos individuais. Após a aprovação, você mescla as `feature` ramificações em uma ramificação de integração que geralmente é nomeada`develop`. Quando recursos suficientes se acumulam na `develop` ramificação, uma `release` ramificação é criada para implantar os recursos nos ambientes superiores.

Cada estratégia de ramificação tem vantagens e desvantagens. Embora todos usem os mesmos ambientes, nem todos usam as mesmas filiais ou etapas de aprovação manual. Nesta seção do guia, analise cada estratégia de ramificação em detalhes para que você esteja familiarizado com suas nuances e possa avaliar se ela se adequa ao caso de uso da sua organização.

Tópicos nesta seção:

Estratégia de ramificação de troncos

O desenvolvimento baseado em troncos é uma prática de desenvolvimento de software na qual todos os desenvolvedores trabalham em uma única ramificação, normalmente chamada de ramificação `trunk` ou`main`. A ideia por trás dessa abordagem é manter a base de código em um

estado continuamente liberável, integrando alterações de código com frequência e confiando em testes automatizados e integração contínua.

No desenvolvimento baseado em troncos, os desenvolvedores confirmam suas alterações na `main` ramificação várias vezes ao dia, visando atualizações pequenas e incrementais. Isso permite ciclos rápidos de feedback, reduz o risco de conflitos de fusão e promove a colaboração entre os membros da equipe. A prática enfatiza a importância de uma suíte de testes bem mantida, pois ela depende de testes automatizados para detectar possíveis problemas com antecedência e garantir que a base do código permaneça estável e liberável.

O desenvolvimento baseado em troncos geralmente é contrastado com o desenvolvimento baseado em recursos (também conhecido como ramificação de recursos ou desenvolvimento orientado a recursos), em que cada novo recurso ou correção de bug é desenvolvido em sua própria ramificação dedicada, separada da ramificação principal. A escolha entre desenvolvimento baseado em troncos e desenvolvimento baseado em recursos depende de fatores como tamanho da equipe, requisitos do projeto e o equilíbrio desejado entre colaboração, frequência de integração e gerenciamento de lançamentos.

Para obter mais informações sobre a estratégia de ramificação de troncos, consulte os seguintes recursos:

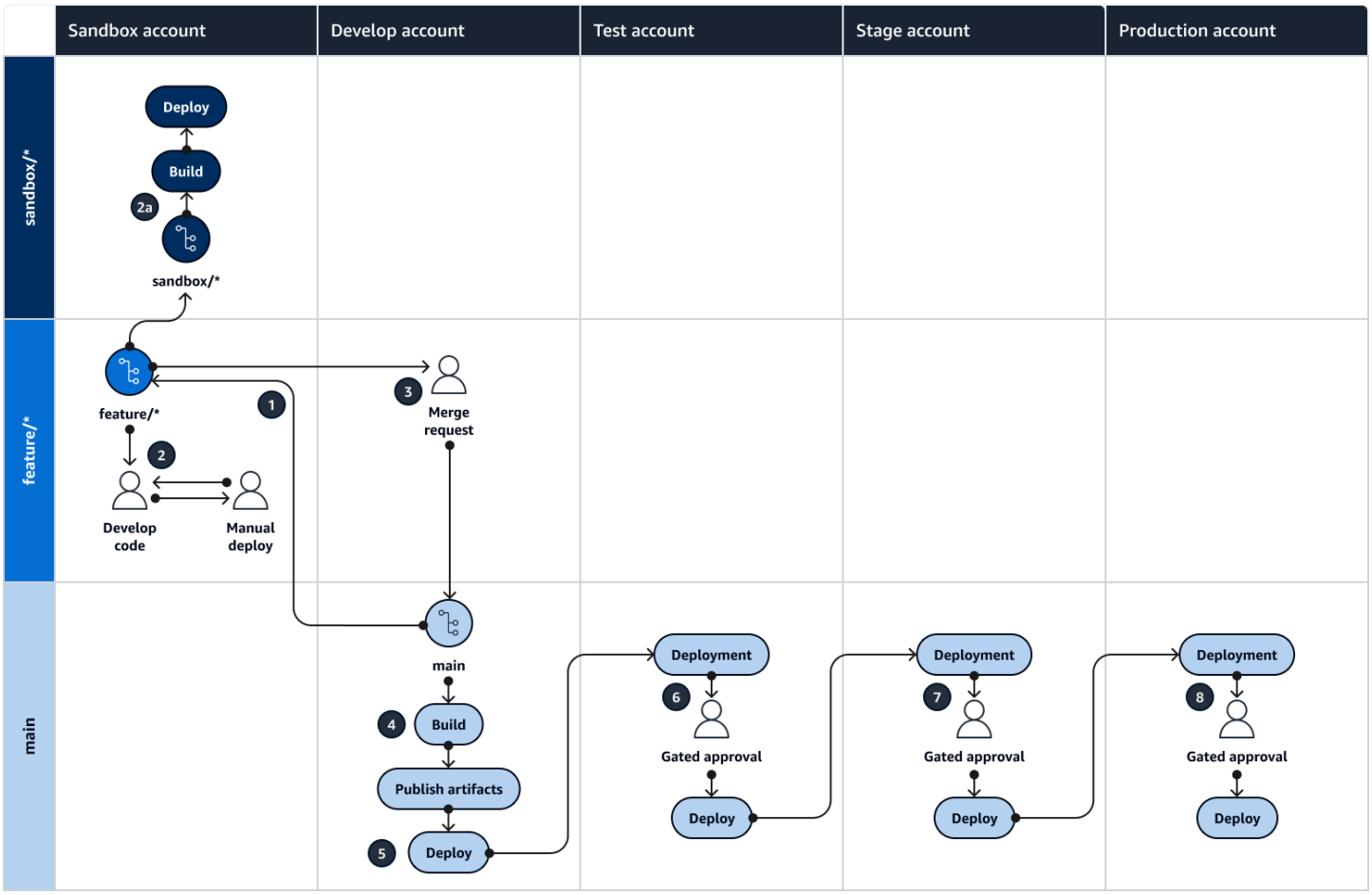
- Implemente uma estratégia de ramificação de troncos para DevOps ambientes com várias contas (orientaçãoAWS prescritiva)
- Introdução ao desenvolvimento baseado em troncos (site de desenvolvimento baseado em troncos)

Tópicos nesta seção:

Visão geral visual da estratégia Trunk

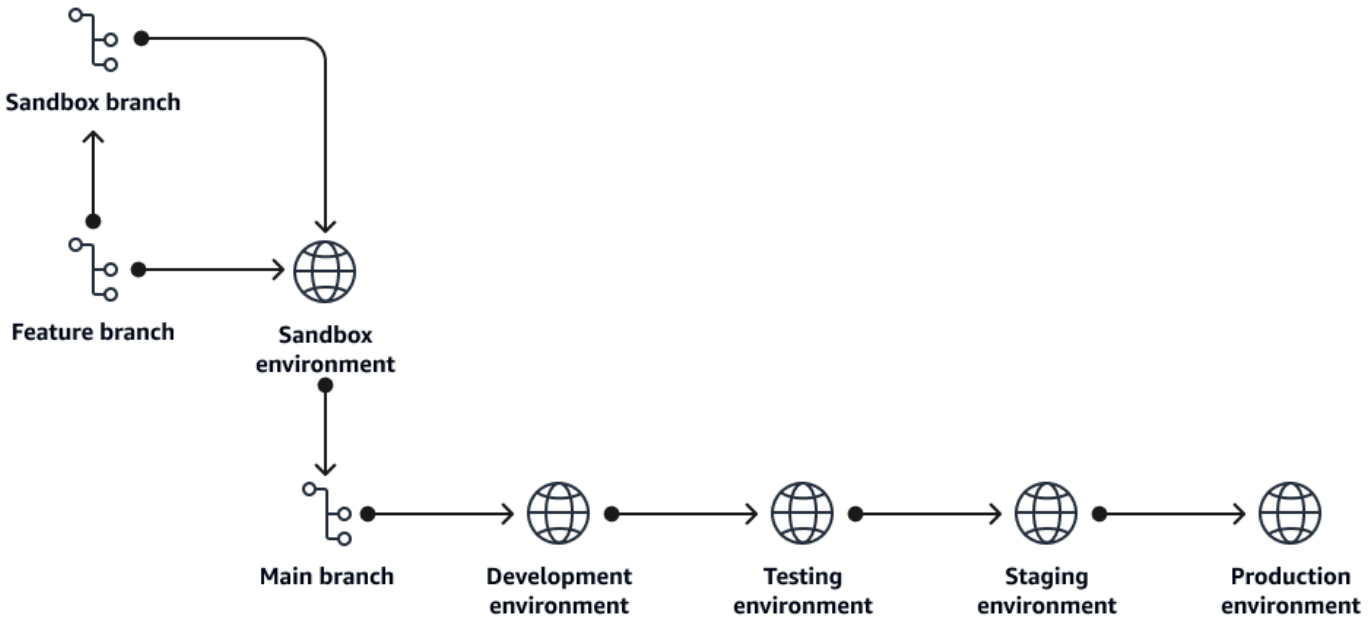
O diagrama a seguir pode ser usado como um quadrado de Punnett (Wikipedia) para entender a estratégia de ramificação do tronco. Alinhe as ramificações no eixo vertical com os AWS ambientes no eixo horizontal para determinar quais ações realizar em cada cenário. Os números circulados

guiam você pela sequência de ações representada no diagrama. Este diagrama mostra o fluxo de trabalho de desenvolvimento de uma estratégia de ramificação de troncos, desde uma `feature` ramificação no ambiente sandbox até a versão de produção da `main` filial. Para obter mais informações sobre as atividades que ocorrem em cada ambiente, consulte DevOps ambientes neste guia.



Estratégia de filiais em um tronco

Uma estratégia de ramificação de tronco geralmente tem as seguintes ramificações.



ramificação de recursos

Você desenvolve recursos ou cria um hotfix em uma `feature` ramificação. Para criar uma `feature` ramificação, você ramifica a partir da `main` ramificação. Os desenvolvedores iteram, confirmam e testam o código em uma `feature` ramificação. Quando um recurso é concluído, o desenvolvedor promove o recurso. Há apenas dois caminhos a partir de uma `feature` ramificação:

- Fusão na filial `sandbox`
- Crie uma solicitação de mesclagem na filial `main`

Convenção de nomenclatura:	`feature/<story number>_<developer initials>_<descriptor>`
Exemplo de convenção de nomenclatura:	`feature/123456_MS_Implement_Feature_A`

filial de sandbox

Essa ramificação é uma ramificação de tronco não padrão, mas é útil para o desenvolvimento de pipeline de CI/CD. A `sandbox` filial é usada principalmente para as seguintes finalidades:

- Execute uma implantação completa no ambiente sandbox usando os pipelines de CI/CD
- Desenvolva e teste um pipeline antes de enviar solicitações de mesclagem para testes completos em um ambiente inferior, como desenvolvimento ou teste.

`Sandbox`as filiais são de natureza temporária e devem ter vida curta. Eles devem ser excluídos após a conclusão do teste específico.

Convenção de nomenclatura:	`sandbox/<story number>_<developer initials>_<descriptor>`
Exemplo de convenção de nomenclatura:	`sandbox/123456_MS_Test_Pipeline_Deploy`

ramificação principal

A `main` ramificação sempre representa o código que está sendo executado na produção. O código é ramificado`main`, desenvolvido e, em seguida, mesclado novamente com o. `main` As implantações do `main` podem ser direcionadas a qualquer ambiente. Para se proteger contra exclusão, ative a proteção de ramificação para a `main` ramificação.

Convenção de nomenclatura:	`main`

ramificação de hotfix

Não há `hotfix` ramificação dedicada em um fluxo de trabalho baseado em troncos. Os hotfixes usam `feature` ramificações.

Vantagens e desvantagens da estratégia Trunk

A estratégia de ramificação do Trunk é adequada para equipes de desenvolvimento menores e maduras que tenham fortes habilidades de comunicação. Também funciona bem se você tiver lançamentos contínuos e contínuos de recursos para o aplicativo. Não é adequado se você tiver equipes de desenvolvimento grandes ou fragmentadas ou se tiver lançamentos de recursos expansivos e programados. Conflitos de mesclagem ocorrerão neste modelo, portanto, esteja ciente

de que a resolução de conflitos de mesclagem é uma habilidade fundamental. Todos os membros da equipe devem ser treinados adequadamente.

Vantagens

O desenvolvimento baseado em troncos oferece várias vantagens que podem melhorar o processo de desenvolvimento, agilizar a colaboração e aprimorar a qualidade geral do software. A seguir estão alguns dos principais benefícios:

- Ciclos de feedback mais rápidos — Com o desenvolvimento baseado em troncos, os desenvolvedores integram suas alterações de código com frequência, muitas vezes várias vezes ao dia. Isso permite um feedback mais rápido sobre possíveis problemas e ajuda os desenvolvedores a identificar e corrigir problemas mais rapidamente do que fariam em um modelo de desenvolvimento baseado em recursos.
- Conflitos de mesclagem reduzidos — No desenvolvimento baseado em troncos, o risco de conflitos de mesclagem grandes e complicados é minimizado porque as mudanças são integradas continuamente. Isso ajuda a manter uma base de código mais limpa e reduz o tempo gasto na resolução de conflitos. Resolver conflitos pode ser demorado e propenso a erros no desenvolvimento baseado em recursos.
- Colaboração aprimorada — O desenvolvimento baseado em troncos incentiva os desenvolvedores a trabalharem juntos na mesma filial, promovendo melhor comunicação e colaboração dentro da equipe. Isso pode levar a uma solução mais rápida de problemas e a uma dinâmica de equipe mais coesa.
- Revisões de código mais fáceis — Como as alterações de código são menores e mais frequentes no desenvolvimento baseado em troncos, pode ser mais fácil realizar análises de código completas. Mudanças menores geralmente são mais fáceis de entender e revisar, levando a uma identificação mais eficaz de possíveis problemas e melhorias.
- Integração e entrega contínuas — O desenvolvimento baseado em troncos apóia os princípios de integração e entrega contínuas (CI/CD). Ao manter a base de código em um estado liberável e integrar as mudanças com frequência, as equipes podem adotar com mais facilidade as práticas de CI/CD, o que leva a ciclos de implantação mais rápidos e a uma melhor qualidade do software.
- Qualidade de código aprimorada — Com integrações, testes e análises de código frequentes, o desenvolvimento baseado em troncos pode contribuir para uma melhor qualidade geral do código. Os desenvolvedores podem capturar e corrigir problemas mais rapidamente, reduzindo a probabilidade de acumulação de dívidas técnicas ao longo do tempo.

- Estratégia simplificada de ramificação — O desenvolvimento baseado em troncos simplifica a estratégia de ramificação ao reduzir o número de ramificações de longa duração. Isso pode facilitar o gerenciamento e a manutenção da base de código, especialmente para grandes projetos ou equipes.

Desvantagens

O desenvolvimento baseado em troncos tem algumas desvantagens, que podem afetar o processo de desenvolvimento e a dinâmica da equipe. A seguir estão algumas desvantagens notáveis:

- Isolamento limitado — Como todos os desenvolvedores trabalham na mesma ramificação, suas alterações são imediatamente visíveis para todos na equipe. Isso pode levar a interferências ou conflitos, causando efeitos colaterais não intencionais ou interrompendo a construção. Por outro lado, o desenvolvimento baseado em recursos isola melhor as mudanças para que os desenvolvedores possam trabalhar de forma mais independente.
- Maior pressão sobre os testes — O desenvolvimento baseado em troncos depende da integração contínua e dos testes automatizados para detectar problemas rapidamente. No entanto, essa abordagem pode colocar muita pressão na infraestrutura de testes e exigir uma suíte de testes bem mantida. Se os testes não forem abrangentes ou confiáveis, isso pode levar a problemas não detectados na filial principal.
- Menos controle sobre os lançamentos — O desenvolvimento baseado em troncos visa manter a base de código em um estado continuamente liberável. Embora isso possa ser vantajoso, nem sempre é adequado para projetos com cronogramas de lançamento rígidos ou aqueles que exigem que recursos específicos sejam lançados juntos. O desenvolvimento baseado em recursos fornece mais controle sobre quando e como os recursos são lançados.
- Rotatividade de código — Com os desenvolvedores integrando constantemente as mudanças na ramificação principal, o desenvolvimento baseado em troncos pode levar a uma maior rotatividade de código. Isso pode dificultar que os desenvolvedores acompanhem o estado atual da base de código e pode causar confusão ao tentar entender o efeito das mudanças recentes.
- Requer uma forte cultura de equipe — O desenvolvimento baseado em troncos exige um alto nível de disciplina, comunicação e colaboração entre os membros da equipe. Isso pode ser difícil de manter, especialmente em equipes maiores ou quando se trabalha com desenvolvedores menos experientes com essa abordagem.
- Desafios de escalabilidade — À medida que o tamanho da equipe de desenvolvimento cresce, o número de alterações de código integradas à ramificação principal pode aumentar rapidamente.

Isso pode levar a quebras de compilação e falhas de teste mais frequentes, dificultando a manutenção da base de código em um estado liberável.

GitHub Estratégia de ramificação de fluxo

GitHub O Flow é um fluxo de trabalho leve e baseado em ramificações, desenvolvido pela. GitHub GitHubO fluxo é baseado na ideia de ramificações de recursos de curta duração que são mescladas à ramificação principal quando o recurso está concluído e pronto para ser implantado. Os princípios fundamentais do GitHub Flow são:

- A ramificação é leve — os desenvolvedores podem criar ramificações de recursos para seu trabalho com apenas alguns cliques, melhorando a capacidade de colaborar e experimentar sem afetar a ramificação principal.
- Implantação contínua — As alterações são implantadas assim que são incorporadas à filial principal, o que permite feedback e iteração rápidos.
- Solicitações de mesclagem — Os desenvolvedores usam solicitações de mesclagem para iniciar um processo de discussão e revisão antes de mesclar suas alterações na ramificação principal.

Para obter mais informações sobre GitHub o Flow, consulte os seguintes recursos:

- Implemente uma estratégia GitHub de ramificação do Flow para DevOps ambientes com várias contas (orientaçãoAWS prescritiva)
- GitHub Flow Quickstart (GitHub documentação)
- Por que GitHub Flow? (Site GitHub do Flow)

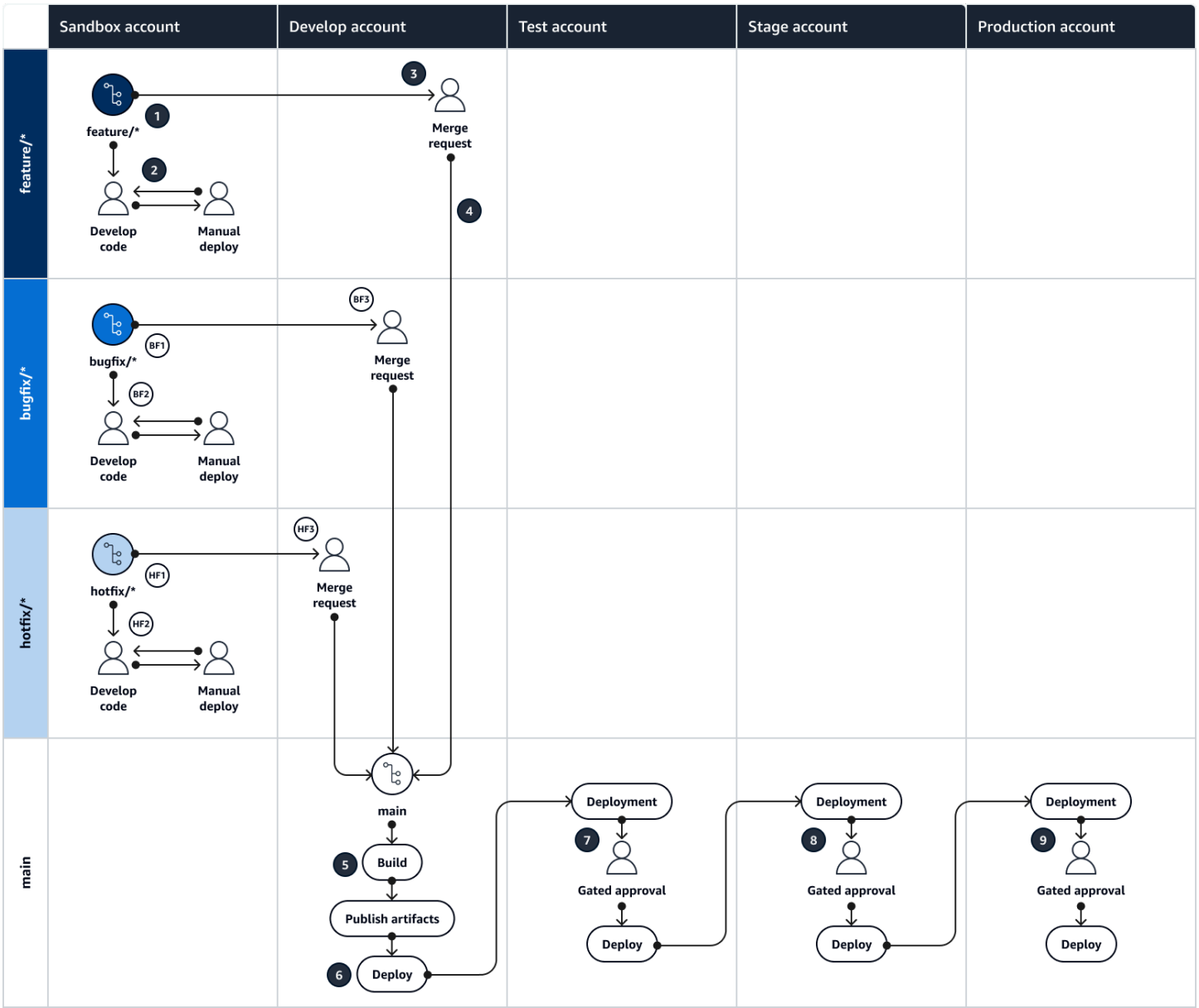
Tópicos nesta seção:

- Visão geral visual da estratégia GitHub Flow
- Filiais em uma estratégia GitHub de fluxo
- Vantagens e desvantagens da estratégia GitHub Flow

Visão geral visual da estratégia GitHub Flow

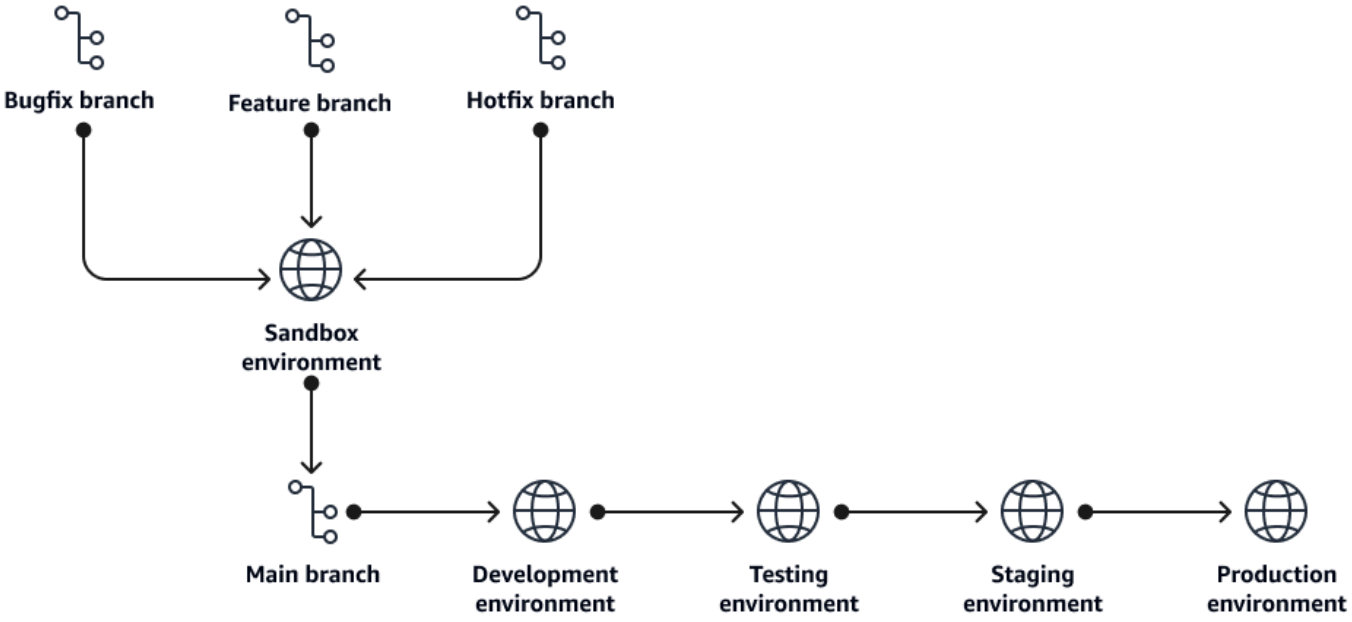
O diagrama a seguir pode ser usado como um quadrado de Punnett para entender a estratégia de ramificação do GitHub Flow. Alinhe as ramificações no eixo vertical com os AWS ambientes no eixo

horizontal para determinar quais ações realizar em cada cenário. Os números circulados guiam você pela sequência de ações representada no diagrama. Este diagrama mostra o fluxo de trabalho de desenvolvimento de uma estratégia de ramificação do GitHub Flow, desde uma ramificação de recursos no ambiente sandbox até a versão de produção da ramificação principal. Para obter mais informações sobre as atividades que ocorrem em cada ambiente, consulte DevOps ambientes neste guia.



Filiais em uma estratégia GitHub de fluxo

Uma estratégia GitHub de ramificação do Flow geralmente tem as seguintes ramificações.



ramificação de recursos

Você desenvolve recursos em `feature` filiais. Para criar uma `feature` ramificação, você ramifica a partir da `main` ramificação. Os desenvolvedores iteram, confirmam e testam o código na `feature` ramificação. Quando um recurso é concluído, o desenvolvedor promove o recurso criando uma solicitação de mesclagem para`main`.

Convenção de nomenclatura:	`feature/<story number>_<developer initials>_<descriptor>`
Exemplo de convenção de nomenclatura:	`feature/123456_MS_Implement_Feature_A`

ramificação de correção de bugs

A `bugfix` ramificação é usada para corrigir problemas. Esses galhos são ramificados do `main` galho. Depois que a correção de bug é testada no sandbox ou em qualquer um dos ambientes inferiores, ela pode ser promovida para ambientes superiores mesclando-a por `main` meio de uma solicitação de mesclagem. Essa é uma convenção de nomenclatura sugerida para organização e rastreamento. Esse processo também pode ser gerenciado usando uma ramificação de recursos.

Convenção de nomenclatura:	`bugfix/<ticket number>_<developer initials>_<descriptor>`
Exemplo de convenção de nomenclatura:	`bugfix/123456_MS_Fix_Problem_A`

ramificação de hotfix

A `hotfix` ramificação é usada para resolver problemas críticos de alto impacto com o mínimo de atraso entre a equipe de desenvolvimento e o código implantado na produção. Esses galhos são ramificados do `main` galho. Depois que o hotfix é testado no sandbox ou em qualquer um dos ambientes inferiores, ele pode ser promovido para ambientes superiores mesclando-o por `main` meio de uma solicitação de mesclagem. Essa é uma convenção de nomenclatura sugerida para organização e rastreamento. Esse processo também pode ser gerenciado usando uma ramificação de recursos.

Convenção de nomenclatura:	`hotfix/<ticket number>_<developer initials>_<descriptor>`
Exemplo de convenção de nomenclatura:	`hotfix/123456_MS_Fix_Problem_A`

ramificação principal

A `main` ramificação sempre representa o código que está sendo executado na produção. O código é mesclado na `main` ramificação a partir das `feature` ramificações usando solicitações de mesclagem. Para se proteger contra exclusão e impedir que os desenvolvedores enviem código diretamente para a ramificação`main`, ative a proteção de ramificação para a `main` ramificação.

Convenção de nomenclatura:	`main`

Vantagens e desvantagens da estratégia GitHub Flow

A estratégia de ramificação do Github Flow é adequada para equipes de desenvolvimento menores e maduras que tenham fortes habilidades de comunicação. Essa estratégia é adequada para equipes que desejam implementar a entrega contínua e é bem apoiada por mecanismos comuns de CI/CD.

GitHub O Flow é leve, não tem muitas regras e é capaz de apoiar equipes em rápida evolução. Não é adequado que suas equipes tenham processos rígidos de conformidade ou liberação a serem seguidos. Conflitos de mesclagem são comuns nesse modelo e provavelmente ocorrerão com frequência. A resolução de conflitos de fusão é uma habilidade fundamental, e você deve treinar todos os membros da equipe adequadamente.

Vantagens

GitHub O Flow oferece várias vantagens que podem melhorar o processo de desenvolvimento, agilizar a colaboração e aprimorar a qualidade geral do software. A seguir estão alguns dos principais benefícios:

- Flexível e leve — O GitHub Flow é um fluxo de trabalho leve e flexível que ajuda os desenvolvedores a colaborar em projetos de desenvolvimento de software. Ele permite iteração e experimentação rápidas com o mínimo de complexidade.
- Colaboração simplificada — O GitHub Flow fornece um processo claro e simplificado para gerenciar o desenvolvimento de recursos. Ele incentiva mudanças pequenas e focadas que podem ser rapidamente revisadas e mescladas, melhorando a eficiência.
- Controle de versão claro — Com GitHub o Flow, cada alteração é feita em uma ramificação separada. Isso estabelece um histórico de controle de versão claro e rastreável. Isso ajuda os desenvolvedores a rastrear e entender as mudanças, revertê-las, se necessário, e manter uma base de código confiável.
- Integração contínua perfeita — O GitHub Flow se integra às ferramentas de integração contínua. A criação de pull requests pode iniciar processos automatizados de teste e implantação. As ferramentas de CI ajudam você a testar minuciosamente as alterações antes que elas sejam incorporadas à `main` ramificação, reduzindo o risco de introduzir bugs na base de código.
- Feedback rápido e melhoria contínua — O GitHub Flow incentiva um ciclo de feedback rápido, promovendo revisões frequentes de código e discussões por meio de pull requests. Isso facilita a detecção precoce de problemas, promove o compartilhamento de conhecimento entre os membros da equipe e, por fim, leva a uma maior qualidade de código e a uma melhor colaboração dentro da equipe de desenvolvimento.
- Reversões e reversões simplificadas — Caso uma alteração no código introduza um bug ou problema inesperado, o GitHub Flow simplifica o processo de reverter ou reverter a alteração. Com um histórico claro de commits e ramificações, é mais fácil identificar e reverter alterações problemáticas, ajudando a manter uma base de código estável e funcional.

- Curva de aprendizado leve — O GitHub Flow pode ser mais fácil de aprender e adotar do que o Gitflow, especialmente para equipes já familiarizadas com o Git e os conceitos de controle de versão. Sua simplicidade e seu modelo intuitivo de ramificação o tornam acessível para desenvolvedores com diferentes níveis de experiência, reduzindo a curva de aprendizado associada à adoção de novos fluxos de trabalho de desenvolvimento.
- Desenvolvimento contínuo — O GitHub Flow capacita as equipes a adotar uma abordagem de implantação contínua, permitindo a implantação imediata de todas as alterações assim que elas são incorporadas à `main` filial. Esse processo simplificado elimina atrasos desnecessários e garante que as atualizações e melhorias mais recentes sejam disponibilizadas rapidamente aos usuários. Isso resulta em um ciclo de desenvolvimento mais ágil e responsivo.

Desvantagens

Embora GitHub o Flow ofereça várias vantagens, é importante considerar também suas possíveis desvantagens:

- Adequação limitada para grandes projetos — O GitHub Flow pode não ser tão adequado para projetos de grande escala com bases de código complexas e várias ramificações de recursos de longo prazo. Nesses casos, um fluxo de trabalho mais estruturado, como o Gitflow, pode fornecer melhor controle sobre o desenvolvimento simultâneo e o gerenciamento de lançamentos.
- Falta de estrutura formal de lançamento — o GitHub Flow não define explicitamente um processo de lançamento ou recursos de suporte, como controle de versão, hotfixes ou ramificações de manutenção. Isso pode ser uma limitação para projetos que exigem gerenciamento rigoroso de lançamentos ou precisam de suporte e manutenção de longo prazo.
- Suporte limitado para planejamento de lançamentos de longo prazo — O GitHub Flow se concentra em ramificações de recursos de curta duração, que podem não se alinhar bem a projetos que exigem planejamento de lançamentos de longo prazo, como aqueles com roteiros rígidos ou dependências extensivas de recursos. Gerenciar cronogramas de lançamento complexos pode ser um desafio dentro das restrições do Flow. GitHub
- Potencial para conflitos de mesclagem frequentes — Como o GitHub Flow incentiva ramificações e fusões frequentes, existe a possibilidade de encontrar conflitos de mesclagem, especialmente em projetos com muitas atividades de desenvolvimento. Resolver esses conflitos pode ser demorado e exigir um esforço adicional da equipe de desenvolvimento.
- Falta de fases formalizadas do fluxo de trabalho — O GitHub fluxo não define fases explícitas para o desenvolvimento, como estágios alfa, beta ou candidatos a lançamento. Isso pode dificultar a

comunicação do estado atual do projeto ou do nível de estabilidade de diferentes ramificações ou versões.

- Impacto de alterações significativas — Como o GitHub Flow incentiva a mesclagem frequente de alterações na `main` ramificação, há um risco maior de introduzir alterações significativas que afetem a estabilidade da base de código. Práticas rígidas de revisão e teste de código são cruciais para mitigar esse risco de forma eficaz.

Estratégia de ramificação do Gitflow

O Gitflow é um modelo de ramificação que envolve o uso de várias ramificações para mover o código do desenvolvimento para a produção. O Gitflow funciona bem para equipes que têm ciclos de lançamento programados e precisam definir uma coleção de recursos como um lançamento. O desenvolvimento é concluído em ramificações de recursos individuais que são mescladas, com aprovação, em uma ramificação de desenvolvimento, que é usada para integração. Os recursos desse ramo são considerados prontos para produção. Quando todos os recursos planejados tiverem sido acumulados na ramificação de desenvolvimento, uma ramificação de lançamento será criada para implantações em ambientes superiores. Essa separação melhora o controle sobre quais alterações estão sendo transferidas para qual ambiente nomeado em um cronograma definido. Se necessário, você pode acelerar esse processo para um modelo de implantação mais rápido.

Para obter mais informações sobre a estratégia de ramificação do Gitflow, consulte os seguintes recursos:

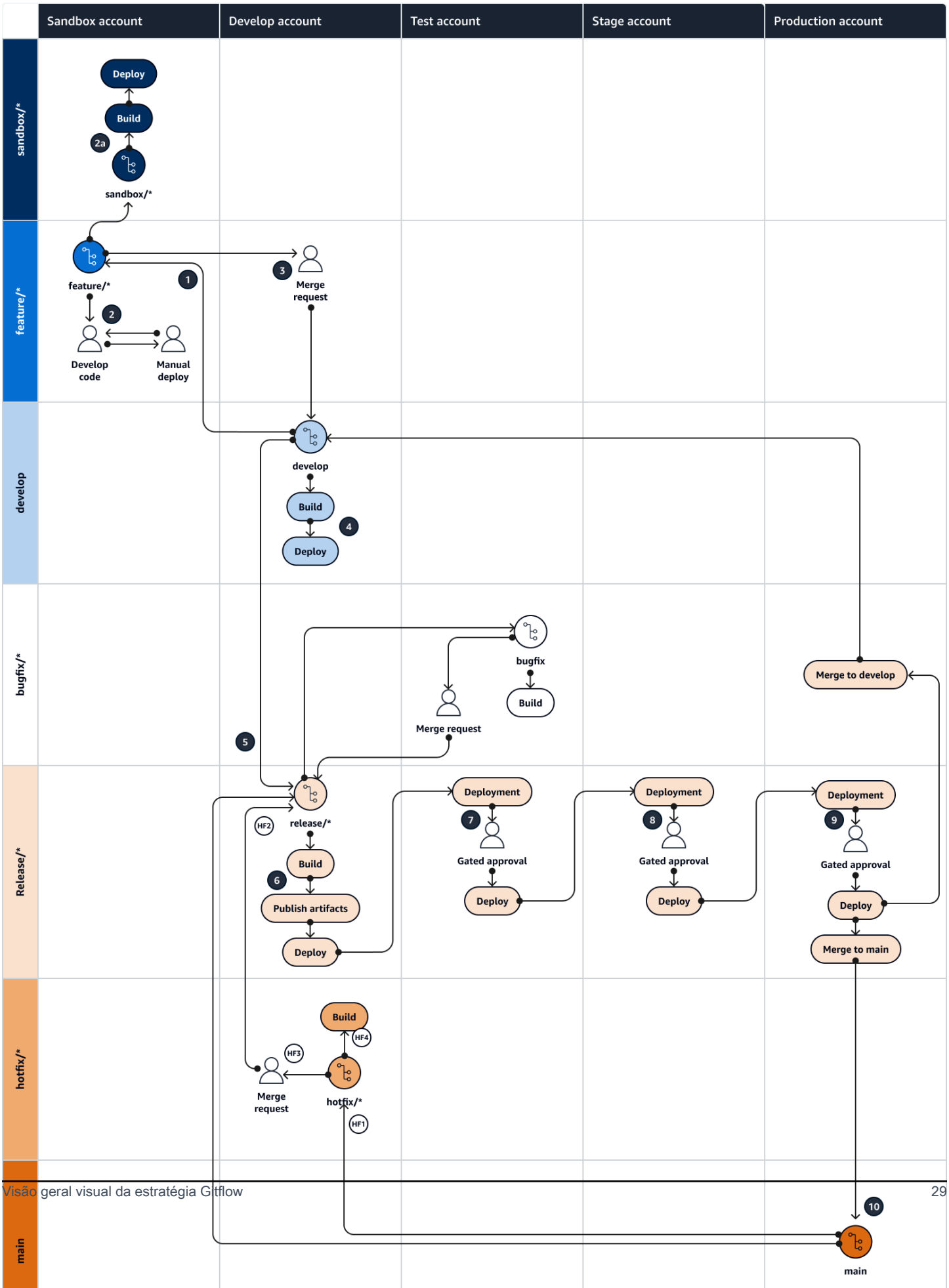
- Implemente uma estratégia de ramificação do Gitflow para DevOps ambientes com várias contas (orientação prescritiva)AWS
- O blog original do Gitflow (postagem no blog de Vincent Driessen)
- Fluxo de trabalho do Gitflow (Atlassian)

Tópicos nesta seção:

- Visão geral visual da estratégia Gitflow
- Filiais em uma estratégia Gitflow
- Vantagens e desvantagens da estratégia Gitflow

Visão geral visual da estratégia Gitflow

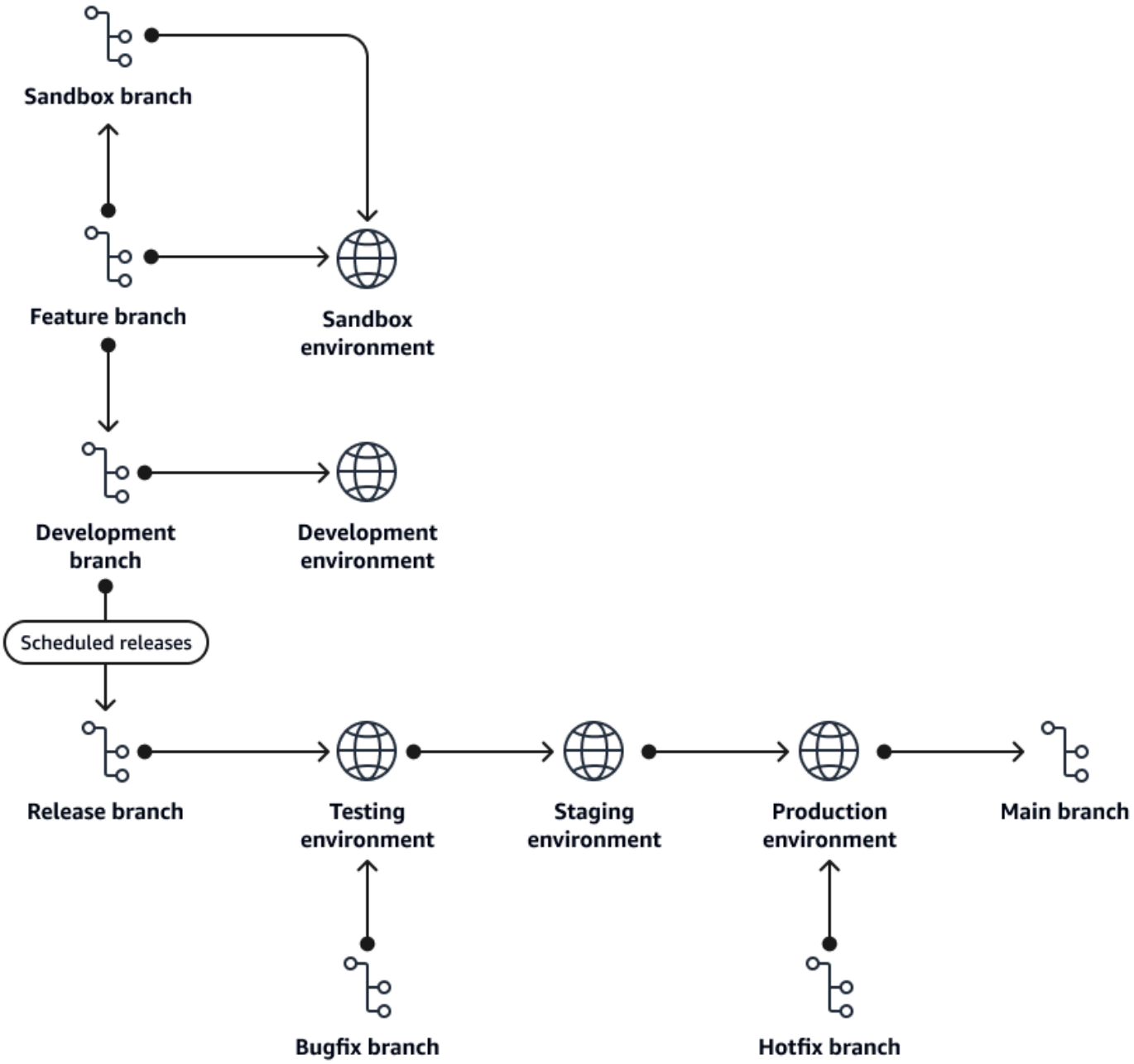
O diagrama a seguir pode ser usado como um quadrado de Punnett para entender a estratégia de ramificação do Gitflow. Alinhe as ramificações no eixo vertical com os AWS ambientes no eixo horizontal para determinar quais ações realizar em cada cenário. Os números circulados guiam você pela sequência de ações representada no diagrama. Para obter mais informações sobre as atividades que ocorrem em cada ambiente, consulte DevOps ambientes neste guia.



Visão geral visual da estratégia Gitflow

Filiais em uma estratégia Gitflow

Uma estratégia de ramificação do Gitflow geralmente tem as seguintes ramificações.



ramificação de recursos

`Feature`filiais são filiais de curto prazo nas quais você desenvolve recursos. A `feature` ramificação é criada pela ramificação da `develop` ramificação. Os desenvolvedores iteram, confirmam e testam

o código na `feature` ramificação. Quando o recurso é concluído, o desenvolvedor promove o recurso. Há apenas dois caminhos a partir de uma ramificação de recursos:

- Fusão na filial `sandbox`
- Crie uma solicitação de mesclagem na filial `develop`

Convenção de nomenclatura:	`feature/<story number>_<developer initials>_<descriptor>`
Exemplo de convenção de nomenclatura:	`feature/123456_MS_Implement_Feature_A`

filial de sandbox

A `sandbox` filial é uma ramificação não padrão de curto prazo do Gitflow. No entanto, é útil para o desenvolvimento de pipelines de CI/CD. A `sandbox` filial é usada principalmente para as seguintes finalidades:

- Execute uma implantação completa no ambiente sandbox usando os pipelines de CI/CD em vez de uma implantação manual.
- Desenvolva e teste um pipeline antes de enviar solicitações de mesclagem para testes completos em um ambiente inferior, como desenvolvimento ou teste.

`Sandbox`os galhos são de natureza temporária e não devem ter vida longa. Eles devem ser excluídos após a conclusão do teste específico.

Convenção de nomenclatura:	`sandbox/<story number>_<developer initials>_<descriptor>`
Exemplo de convenção de nomenclatura:	`sandbox/123456_MS_Test_Pipeline_Deploy`

desenvolver filial

A `develop` filial é uma filial de longa duração em que os recursos são integrados, criados, validados e implantados no ambiente de desenvolvimento. Todas as `feature` filiais são mescladas na `develop` ramificação. As fusões na `develop` ramificação são concluídas por meio de uma solicitação de mesclagem que requer uma compilação bem-sucedida e duas aprovações do desenvolvedor. Para evitar a exclusão, ative a proteção da ramificação na `develop` ramificação.

Convenção de nomenclatura:	`develop`

ramificação de lançamento

No Gitflow, `release` filiais são filiais de curto prazo. Essas ramificações são especiais porque você pode implantá-las em vários ambientes, adotando a metodologia de criar uma vez e implantar muitos. `Release`as filiais podem ter como alvo os ambientes de teste, preparação ou produção. Depois que uma equipe de desenvolvimento decidiu promover recursos em ambientes superiores, eles criam uma nova `release` ramificação e usam incrementar o número da versão da versão anterior. Nos portões de cada ambiente, as implantações exigem aprovações manuais para prosseguir. `Release`as filiais devem exigir que uma solicitação de mesclagem seja alterada.

Depois que a `release` ramificação for implantada na produção, ela deverá ser mesclada novamente às `main` ramificações `develop` e para garantir que quaisquer correções de bugs ou hotfixes sejam incorporados novamente aos esforços futuros de desenvolvimento.

Convenção de nomenclatura:	`release/v{major}.{minor}`
Exemplo de convenção de nomenclatura:	`release/v1.0`

ramificação principal

A `main` ramificação é uma ramificação de longa duração que sempre representa o código que está sendo executado na produção. O código é mesclado automaticamente na `main` ramificação de uma ramificação de lançamento após uma implantação bem-sucedida do pipeline de lançamento. Para evitar a exclusão, ative a proteção da ramificação na `main` ramificação.

Convenção de nomenclatura:	`main`

ramificação de correção de bugs

A `bugfix` ramificação é uma ramificação de curto prazo usada para corrigir problemas em ramificações de lançamento que não foram lançadas para produção. Uma `bugfix` ramificação só deve ser usada para promover correções em `release` ramificações para os ambientes de teste, preparação ou produção. Um `bugfix` galho é sempre ramificado de um `release` galho.

Depois que a correção de bug for testada, ela poderá ser promovida para a `release` ramificação por meio de uma solicitação de mesclagem. Em seguida, você pode empurrar a `release` ramificação para frente seguindo o processo de liberação padrão.

Convenção de nomenclatura:	`bugfix/<ticket>_<developer initials>_<descriptor>`
Exemplo de convenção de nomenclatura:	`bugfix/123456_MS_Fix_Problem_A`

ramificação de hotfix

A `hotfix` filial é uma filial de curto prazo usada para corrigir problemas na produção. Ele só pode ser usado para promover correções que devem ser aceleradas para chegar ao ambiente de produção. Um `hotfix` galho é sempre ramificado de`main`.

Depois que o hotfix for testado, você poderá promovê-lo para produção por meio de uma solicitação de mesclagem na `release` ramificação a partir da qual foi criada. `main` Para testar, você pode então empurrar a `release` ramificação para frente seguindo o processo de liberação padrão.

Convenção de nomenclatura:	`hotfix/<ticket>_<developer initials>_<descriptor>`
Exemplo de convenção de nomenclatura:	`hotfix/123456_MS_Fix_Problem_A`

Vantagens e desvantagens da estratégia Gitflow

A estratégia de ramificação do Gitflow é adequada para equipes maiores e mais distribuídas que têm requisitos rígidos de liberação e conformidade. O Gitflow contribui para um ciclo de lançamento previsível para a organização, e isso geralmente é preferido por organizações maiores. O Gitflow também é adequado para equipes que precisam de barreiras para concluir adequadamente o ciclo de vida de desenvolvimento de software. Isso ocorre porque há várias oportunidades de análises e garantia de qualidade incorporadas à estratégia. O Gitflow também é adequado para equipes que precisam manter várias versões dos lançamentos de produção simultaneamente. Algumas desvantagens do GitFlow são que ele é mais complexo do que outros modelos de ramificação e requer adesão estrita ao padrão para ser concluído com sucesso. O Gitflow não funciona bem para organizações que buscam a entrega contínua devido à natureza rígida do gerenciamento de ramificações de lançamento. As ramificações de lançamento do Gitflow podem ser ramificações de longa duração que podem acumular dívidas técnicas se não forem tratadas adequadamente em tempo hábil.

Vantagens

O desenvolvimento baseado em Gitflow oferece várias vantagens que podem melhorar o processo de desenvolvimento, agilizar a colaboração e aprimorar a qualidade geral do software. A seguir estão alguns dos principais benefícios:

- Processo de lançamento previsível — O Gitflow segue um processo de lançamento regular e previsível. É adequado para equipes com cadências regulares de desenvolvimento e lançamento.
- Colaboração aprimorada — O Gitflow incentiva o uso de `feature` e ramificações. `release` Essas duas ramificações ajudam as equipes a trabalhar em paralelo com dependências mínimas umas das outras.
- Adequado para vários ambientes — o Gitflow usa `release` ramificações, que podem ser ramificações de vida mais longa. Essas filiais permitem que as equipes direcionem lançamentos individuais por um longo período de tempo.
- Várias versões em produção — Se sua equipe oferece suporte a várias versões do software em produção, as `release` filiais do Gitflow atendem a esse requisito.
- Análises de qualidade de código integradas — O Gitflow exige e incentiva o uso de análises e aprovações de código antes que o código seja promovido para outro ambiente. Esse processo elimina o atrito entre os desenvolvedores ao exigir essa etapa para todas as promoções de código.

- Benefícios da organização — O Gitflow também tem vantagens em nível organizacional. O Gitflow incentiva o uso de um ciclo de lançamento padrão, que ajuda a organização a entender e antecipar o cronograma de lançamentos. Como a empresa agora entende quando novos recursos podem ser fornecidos, há menos atrito com os cronogramas porque há datas de entrega definidas.

Desvantagens

O desenvolvimento baseado em Gitflow tem algumas desvantagens que podem afetar o processo de desenvolvimento e a dinâmica da equipe. A seguir estão algumas desvantagens notáveis:

- Complexidade — O Gitflow é um padrão complexo para novas equipes aprenderem, e você deve seguir as regras do Gitflow para usá-lo com sucesso.
- Implantação contínua — O Gitflow não se encaixa em um modelo em que muitas implantações são lançadas para produção de forma rápida. Isso ocorre porque o Gitflow requer o uso de várias ramificações e um fluxo de trabalho rígido que rege a `release` ramificação.
- Gerenciamento de filiais — O Gitflow usa muitas filiais, o que pode ser difícil de manter. Pode ser difícil rastrear as várias ramificações e mesclar o código lançado para manter as ramificações devidamente alinhadas umas com as outras.
- Débito técnico — Como os lançamentos do Gitflow geralmente são mais lentos do que os outros modelos ramificados, mais recursos podem se acumular para o lançamento, o que pode causar o acúmulo de dívidas técnicas.

As equipes devem considerar cuidadosamente essas desvantagens ao decidir se o desenvolvimento baseado em Gitflow é a abordagem correta para o projeto.

Próximas etapas

Este guia explica as diferenças entre três estratégias comuns de ramificação do Git: GitHub Flow, Gitflow e Trunk. Ele descreve seus fluxos de trabalho em detalhes e também fornece as vantagens e desvantagens de cada um. As próximas etapas são escolher um desses fluxos de trabalho padrão para sua organização. Para implementar uma dessas estratégias de ramificação, veja o seguinte:

- Implemente uma estratégia de ramificação de troncos para ambientes com várias contas DevOps
- Implemente uma estratégia GitHub de ramificação do Flow para ambientes com várias contas DevOps
- Implemente uma estratégia de ramificação do Gitflow para ambientes com várias contas DevOps

Se você não sabe por onde começar a jornada de sua equipe para usar o Git DevOps e os processos, recomendamos escolher uma solução padrão e testá-la. Usar uma convenção de ramificação padrão ajuda a equipe a se basear na documentação existente e a aprender o que funciona melhor para ela.

Não tenha medo de mudar sua estratégia se ela não estiver funcionando para sua organização ou equipes de desenvolvimento. As necessidades e exigências das equipes de desenvolvimento podem mudar com o tempo, e não há uma solução única e perfeita.

Recursos

Este guia não inclui treinamento para Git; no entanto, há muitos recursos de alta qualidade disponíveis na Internet se você precisar desse treinamento. Recomendamos que você comece com o site de documentação do Git.

Os recursos a seguir podem ajudá-lo em sua jornada de ramificação do Git no. Nuvem AWS

AWS Orientação prescritiva

- Implemente uma estratégia de ramificação de troncos para ambientes com várias contas DevOps
- Implemente uma estratégia GitHub de ramificação do Flow para ambientes com várias contas DevOps
- Implemente uma estratégia de ramificação do Gitflow para ambientes com várias contas DevOps

Outra AWS orientação

- AWS DevOps Orientação
- AWS Arquitetura de referência do pipeline de implantação
- O que é DevOps?
- DevOpsrecursos

Outros recursos

- Metodologia de aplicativo de doze fatores (12factor.net)
- Segredos do Git (2) GitHub
- Gitflow
  - O blog original do Gitflow (postagem no blog de Vincent Driessen)
  - Fluxo de trabalho do Gitflow (Atlassian)
  - Gitflow ativado GitHub: Como usar fluxos de trabalho do Git Flow com GitHub repositórios baseados (vídeo) YouTube
  - Exemplo de Git Flow Init (vídeo) YouTube

  - A ramificação de lançamento do Gitflow do início ao fim (vídeo) YouTube
- GitHub Fluxo
  - GitHub Flow Quickstart (GitHub documentação)
  - Por que GitHub Flow? (Site GitHub do Flow)
- Tronco
  - Introdução ao desenvolvimento baseado em troncos (site de desenvolvimento baseado em troncos)

Colaboradores

Autoria

- Mike Stephens, arquiteto sênior de aplicativos em nuvem, AWS
- Rayjan Wilson, arquiteto sênior de aplicativos em nuvem, AWS
- Abhilash Vinod, líder de equipe, arquiteto sênior de aplicativos em nuvem, AWS

Analisando

- Stephen DiCato, consultor sênior de segurança, AWS
- Gaurav Samudra, arquiteto de aplicativos em nuvem, AWS
- Steven Guggenheimer, líder de equipe, arquiteto sênior de aplicativos em nuvem, AWS

Redação técnica

- Lilly AbouHarb, redatora técnica sênior, AWS

Histórico do documento

A tabela a seguir descreve alterações significativas feitas neste guia. Se desejar receber notificações sobre futuras atualizações, inscreva-se em um feed RSS.

Alteração	Descrição	Data
Publicação inicial	—	15 de fevereiro de 2024

AWS Glossário de orientação prescritiva

A seguir estão os termos comumente usados em estratégias, guias e padrões fornecidos pela Orientação AWS Prescritiva. Para sugerir entradas, use o link Fornecer feedback no final do glossário.

Números

7 Rs

Sete estratégias comuns de migração para mover aplicações para a nuvem. Essas estratégias baseiam-se nos 5 Rs identificados pela Gartner em 2011 e consistem em:

- Refatorar/rearquitetar: mova uma aplicação e modifique sua arquitetura aproveitando ao máximo os recursos nativos de nuvem para melhorar a agilidade, a performance e a escalabilidade. Isso normalmente envolve a portabilidade do sistema operacional e do banco de dados. Exemplo: migre seu banco de dados Oracle local para a edição compatível com o Amazon Aurora PostgreSQL.
- Redefinir a plataforma (mover e redefinir [mover e redefinir (lift-and-reshape]): mova uma aplicação para a nuvem e introduza algum nível de otimização a fim de aproveitar os recursos da nuvem. Exemplo: Migre seu banco de dados Oracle local para o Amazon Relational Database Service (Amazon RDS) for Oracle no. Nuvem AWS
- Recomprar (drop and shop): mude para um produto diferente, normalmente migrando de uma licença tradicional para um modelo SaaS. Exemplo: migre seu sistema de gerenciamento de relacionamento com o cliente (CRM) para a Salesforce.com.
- Redefinir a hospedagem (mover sem alterações [lift-and-shift])mover uma aplicação para a nuvem sem fazer nenhuma alteração a fim de aproveitar os recursos da nuvem. Exemplo: Migre seu banco de dados Oracle local para o Oracle em uma EC2 instância no. Nuvem AWS
- Realocar (mover o hipervisor sem alterações [hypervisor-level lift-and-shift]): mover a infraestrutura para a nuvem sem comprar novo hardware, reescrever aplicações ou modificar suas operações existentes. Você migra servidores de uma plataforma local para um serviço em nuvem para a mesma plataforma. Exemplo: Migrar um Microsoft Hyper-V aplicativo para o. AWS
- Reter (revisitar): mantenha as aplicações em seu ambiente de origem. Isso pode incluir aplicações que exigem grande refatoração, e você deseja adiar esse trabalho para um

momento posterior, e aplicações antigas que você deseja manter porque não há justificativa comercial para migrá-las.

- Retirar: desative ou remova aplicações que não são mais necessárias em seu ambiente de origem.

A

ABAC

Consulte controle de acesso baseado em atributos.

serviços abstratos

Veja os serviços gerenciados.

ACID

Veja atomicidade, consistência, isolamento, durabilidade.

migração ativa-ativa

Um método de migração de banco de dados no qual os bancos de dados de origem e de destino são mantidos em sincronia (por meio de uma ferramenta de replicação bidirecional ou operações de gravação dupla), e ambos os bancos de dados lidam com transações de aplicações conectadas durante a migração. Esse método oferece suporte à migração em lotes pequenos e controlados, em vez de exigir uma substituição única. É mais flexível, mas exige mais trabalho do que a migração ativa-passiva.

migração ativa-passiva

Um método de migração de banco de dados no qual os bancos de dados de origem e de destino são mantidos em sincronia, mas somente o banco de dados de origem manipula as transações das aplicações conectadas enquanto os dados são replicados no banco de dados de destino. O banco de dados de destino não aceita nenhuma transação durante a migração.

função agregada

Uma função SQL que opera em um grupo de linhas e calcula um único valor de retorno para o grupo. Exemplos de funções agregadas incluem `SUM` e. `MAX`

AI

Veja a inteligência artificial.

AIOps

Veja as operações de inteligência artificial.

anonimização

O processo de excluir permanentemente informações pessoais em um conjunto de dados. A anonimização pode ajudar a proteger a privacidade pessoal. Dados anônimos não são mais considerados dados pessoais.

antipadrões

Uma solução frequentemente usada para um problema recorrente em que a solução é contraproducente, ineficaz ou menos eficaz do que uma alternativa.

controle de aplicativos

Uma abordagem de segurança que permite o uso somente de aplicativos aprovados para ajudar a proteger um sistema contra malware.

portfólio de aplicações

Uma coleção de informações detalhadas sobre cada aplicação usada por uma organização, incluindo o custo para criar e manter a aplicação e seu valor comercial. Essas informações são fundamentais para o processo de descoberta e análise de portfólio e ajudam a identificar e priorizar as aplicações a serem migradas, modernizadas e otimizadas.

inteligência artificial (IA)

O campo da ciência da computação que se dedica ao uso de tecnologias de computação para desempenhar funções cognitivas normalmente associadas aos humanos, como aprender, resolver problemas e reconhecer padrões. Para obter mais informações, consulte O que é inteligência artificial?

operações de inteligência artificial (AIOps)

O processo de usar técnicas de machine learning para resolver problemas operacionais, reduzir incidentes operacionais e intervenção humana e aumentar a qualidade do serviço. Para obter mais informações sobre como AIOps é usado na estratégia de AWS migração, consulte o guia de integração de operações.

criptografia assimétrica

Um algoritmo de criptografia que usa um par de chaves, uma chave pública para criptografia e uma chave privada para descriptografia. É possível compartilhar a chave pública porque ela não é usada na descriptografia, mas o acesso à chave privada deve ser altamente restrito.

atomicidade, consistência, isolamento, durabilidade (ACID)

Um conjunto de propriedades de software que garantem a validade dos dados e a confiabilidade operacional de um banco de dados, mesmo no caso de erros, falhas de energia ou outros problemas.

controle de acesso por atributo (ABAC)

A prática de criar permissões minuciosas com base nos atributos do usuário, como departamento, cargo e nome da equipe. Para obter mais informações, consulte ABAC AWS na documentação AWS Identity and Access Management (IAM).

fonte de dados autorizada

Um local onde você armazena a versão principal dos dados, que é considerada a fonte de informações mais confiável. Você pode copiar dados da fonte de dados autorizada para outros locais com o objetivo de processar ou modificar os dados, como anonimizá-los, redigi-los ou pseudonimizá-los.

Zona de disponibilidade

Um local distinto dentro de um Região da AWS que está isolado de falhas em outras zonas de disponibilidade e fornece conectividade de rede barata e de baixa latência a outras zonas de disponibilidade na mesma região.

AWS Estrutura de adoção da nuvem (AWS CAF)

Uma estrutura de diretrizes e melhores práticas AWS para ajudar as organizações a desenvolver um plano eficiente e eficaz para migrar com sucesso para a nuvem. AWS O CAF organiza a orientação em seis áreas de foco chamadas perspectivas: negócios, pessoas, governança, plataforma, segurança e operações. As perspectivas de negócios, pessoas e governança têm como foco habilidades e processos de negócios; as perspectivas de plataforma, segurança e operações concentram-se em habilidades e processos técnicos. Por exemplo, a perspectiva das pessoas tem como alvo as partes interessadas que lidam com recursos humanos (RH), funções de pessoal e gerenciamento de pessoal. Nessa perspectiva, o AWS CAF fornece orientação para desenvolvimento, treinamento e comunicação de pessoas para ajudar a preparar a organização

para a adoção bem-sucedida da nuvem. Para obter mais informações, consulte o site da AWS CAF e o whitepaper da AWS CAF.

AWS Estrutura de qualificação da carga de trabalho (AWS WQF)

Uma ferramenta que avalia as cargas de trabalho de migração do banco de dados, recomenda estratégias de migração e fornece estimativas de trabalho. AWS O WQF está incluído com AWS Schema Conversion Tool ()AWS SCT. Ela analisa esquemas de banco de dados e objetos de código, código de aplicações, dependências e características de performance, além de fornecer relatórios de avaliação.

B

bot ruim

Um bot destinado a perturbar ou causar danos a indivíduos ou organizações.

BCP

Veja o planejamento de continuidade de negócios.

gráfico de comportamento

Uma visualização unificada e interativa do comportamento e das interações de recursos ao longo do tempo. É possível usar um gráfico de comportamento com o Amazon Detective para examinar tentativas de login malsucedidas, chamadas de API suspeitas e ações similares. Para obter mais informações, consulte Dados em um gráfico de comportamento na documentação do Detective.

sistema big-endian

Um sistema que armazena o byte mais significativo antes. Veja também endianness.

classificação binária

Um processo que prevê um resultado binário (uma de duas classes possíveis). Por exemplo, seu modelo de ML pode precisar prever problemas como “Este e-mail é ou não é spam?” ou “Este produto é um livro ou um carro?”

filtro de bloom

Uma estrutura de dados probabilística e eficiente em termos de memória que é usada para testar se um elemento é membro de um conjunto.

blue/green deployment (implantação azul/verde)

Uma estratégia de implantação em que você cria dois ambientes separados, mas idênticos. Você executa a versão atual do aplicativo em um ambiente (azul) e a nova versão do aplicativo no outro ambiente (verde). Essa estratégia ajuda você a reverter rapidamente com o mínimo de impacto.

bot

Um aplicativo de software que executa tarefas automatizadas pela Internet e simula a atividade ou interação humana. Alguns bots são úteis ou benéficos, como rastreadores da Web que indexam informações na Internet. Alguns outros bots, conhecidos como bots ruins, têm como objetivo perturbar ou causar danos a indivíduos ou organizações.

botnet

Redes de bots infectadas por malware e sob o controle de uma única parte, conhecidas como pastor de bots ou operador de bots. As redes de bots são o mecanismo mais conhecido para escalar bots e seu impacto.

ramo

Uma área contida de um repositório de código. A primeira ramificação criada em um repositório é a ramificação principal. Você pode criar uma nova ramificação a partir de uma ramificação existente e, em seguida, desenvolver recursos ou corrigir bugs na nova ramificação. Uma ramificação que você cria para gerar um recurso é comumente chamada de ramificação de recurso. Quando o recurso estiver pronto para lançamento, você mesclará a ramificação do recurso de volta com a ramificação principal. Para obter mais informações, consulte Sobre filiais (GitHub documentação).

acesso em vidro quebrado

Em circunstâncias excepcionais e por meio de um processo aprovado, um meio rápido para um usuário obter acesso a um Conta da AWS que ele normalmente não tem permissão para acessar. Para obter mais informações, consulte o indicador Implementar procedimentos de quebra de vidro na orientação do Well-Architected AWS .

estratégia brownfield

A infraestrutura existente em seu ambiente. Ao adotar uma estratégia brownfield para uma arquitetura de sistema, você desenvolve a arquitetura de acordo com as restrições dos sistemas e da infraestrutura atuais. Se estiver expandindo a infraestrutura existente, poderá combinar as estratégias brownfield e greenfield.

cache do buffer

A área da memória em que os dados acessados com mais frequência são armazenados.

capacidade de negócios

O que uma empresa faz para gerar valor (por exemplo, vendas, atendimento ao cliente ou marketing). As arquiteturas de microsserviços e as decisões de desenvolvimento podem ser orientadas por recursos de negócios. Para obter mais informações, consulte a seção Organizados de acordo com as capacidades de negócios do whitepaper Executar microsserviços conteinerizados na AWS.

planejamento de continuidade de negócios (BCP)

Um plano que aborda o impacto potencial de um evento disruptivo, como uma migração em grande escala, nas operações e permite que uma empresa retome as operações rapidamente.

C

CAF

Consulte Estrutura de adoção da AWS nuvem.

implantação canária

O lançamento lento e incremental de uma versão para usuários finais. Quando estiver confiante, você implanta a nova versão e substituirá a versão atual em sua totalidade.

CCoE

Veja o Centro de Excelência em Nuvem.

CDC

Veja a captura de dados de alterações.

captura de dados de alterações (CDC)

O processo de rastrear alterações em uma fonte de dados, como uma tabela de banco de dados, e registrar metadados sobre a alteração. É possível usar o CDC para várias finalidades, como auditar ou replicar alterações em um sistema de destino para manter a sincronização.

engenharia do caos

Introduzir intencionalmente falhas ou eventos disruptivos para testar a resiliência de um sistema. Você pode usar AWS Fault Injection Service (AWS FIS) para realizar experimentos que estressam suas AWS cargas de trabalho e avaliar sua resposta.

CI/CD

Veja a integração e a entrega contínuas.

classificação

Um processo de categorização que ajuda a gerar previsões. Os modelos de ML para problemas de classificação predizem um valor discreto. Os valores discretos são sempre diferentes uns dos outros. Por exemplo, um modelo pode precisar avaliar se há ou não um carro em uma imagem.

criptografia no lado do cliente

Criptografia de dados localmente, antes que o alvo os AWS service (Serviço da AWS) receba.

Centro de excelência em nuvem (CCoE)

Uma equipe multidisciplinar que impulsiona os esforços de adoção da nuvem em toda a organização, incluindo o desenvolvimento de práticas recomendas de nuvem, a mobilização de recursos, o estabelecimento de cronogramas de migração e a liderança da organização em transformações em grande escala. Para obter mais informações, consulte as publicações CCo E no Blog de Estratégia Nuvem AWS Empresarial.

computação em nuvem

A tecnologia de nuvem normalmente usada para armazenamento de dados remoto e gerenciamento de dispositivos de IoT. A computação em nuvem geralmente está conectada à tecnologia de computação de ponta.

modelo operacional em nuvem

Em uma organização de TI, o modelo operacional usado para criar, amadurecer e otimizar um ou mais ambientes de nuvem. Para obter mais informações, consulte Criar seu modelo operacional de nuvem.

estágios de adoção da nuvem

As quatro fases pelas quais as organizações normalmente passam quando migram para o Nuvem AWS:

- Projeto: executar alguns projetos relacionados à nuvem para fins de prova de conceito e aprendizado
- Fundação — Fazer investimentos fundamentais para escalar sua adoção da nuvem (por exemplo, criar uma landing zone, definir um CCo E, estabelecer um modelo de operações)
- Migração: migrar aplicações individuais
- Reinvenção: otimizar produtos e serviços e inovar na nuvem

Esses estágios foram definidos por Stephen Orban na postagem do blog The Journey Toward Cloud-First & the Stages of Adoption no blog de estratégia Nuvem AWS empresarial. Para obter informações sobre como eles se relacionam com a estratégia de AWS migração, consulte o guia de preparação para migração.

CMDB

Consulte o banco de dados de gerenciamento de configuração.

repositório de código

Um local onde o código-fonte e outros ativos, como documentação, amostras e scripts, são armazenados e atualizados por meio de processos de controle de versão. Os repositórios de nuvem comuns incluem GitHub ouBitbucket Cloud. Cada versão do código é chamada de ramificação. Em uma estrutura de microsserviços, cada repositório é dedicado a uma única peça de funcionalidade. Um único pipeline de CI/CD pode usar vários repositórios.

cache frio

Um cache de buffer que está vazio, não está bem preenchido ou contém dados obsoletos ou irrelevantes. Isso afeta a performance porque a instância do banco de dados deve ler da memória principal ou do disco, um processo que é mais lento do que a leitura do cache do buffer.

dados frios

Dados que raramente são acessados e geralmente são históricos. Ao consultar esse tipo de dados, consultas lentas geralmente são aceitáveis. Mover esses dados para níveis ou classes de armazenamento de baixo desempenho e menos caros pode reduzir os custos.

visão computacional (CV)

Um campo da IA que usa aprendizado de máquina para analisar e extrair informações de formatos visuais, como imagens e vídeos digitais. Por exemplo, a Amazon SageMaker AI fornece algoritmos de processamento de imagem para CV.

desvio de configuração

Para uma carga de trabalho, uma alteração de configuração em relação ao estado esperado. Isso pode fazer com que a carga de trabalho se torne incompatível e, normalmente, é gradual e não intencional.

banco de dados de gerenciamento de configuração (CMDB)

Um repositório que armazena e gerencia informações sobre um banco de dados e seu ambiente de TI, incluindo componentes de hardware e software e suas configurações. Normalmente, os dados de um CMDB são usados no estágio de descoberta e análise do portfólio da migração.

pacote de conformidade

Um conjunto de AWS Config regras e ações de remediação que você pode montar para personalizar suas verificações de conformidade e segurança. Você pode implantar um pacote de conformidade como uma entidade única em uma Conta da AWS região ou em uma organização usando um modelo YAML. Para obter mais informações, consulte Pacotes de conformidade na documentação. AWS Config

integração contínua e entrega contínua (CI/CD)

O processo de automatizar os estágios de origem, criação, teste, preparação e produção do processo de lançamento do software. CI/CD is commonly described as a pipeline. CI/CDpode ajudá-lo a automatizar processos, melhorar a produtividade, melhorar a qualidade do código e entregar com mais rapidez. Para obter mais informações, consulte Benefícios da entrega contínua. CD também pode significar implantação contínua. Para obter mais informações, consulte Entrega contínua versus implantação contínua.

CV

Veja visão computacional.

D

dados em repouso

Dados estacionários em sua rede, por exemplo, dados que estão em um armazenamento.

classificação de dados

Um processo para identificar e categorizar os dados em sua rede com base em criticalidade e confidencialidade. É um componente crítico de qualquer estratégia de gerenciamento de riscos de

segurança cibernética, pois ajuda a determinar os controles adequados de proteção e retenção para os dados. A classificação de dados é um componente do pilar de segurança no AWS Well-Architected Framework. Para obter mais informações, consulte Classificação de dados.

desvio de dados

Uma variação significativa entre os dados de produção e os dados usados para treinar um modelo de ML ou uma alteração significativa nos dados de entrada ao longo do tempo. O desvio de dados pode reduzir a qualidade geral, a precisão e a imparcialidade das previsões do modelo de ML.

dados em trânsito

Dados que estão se movendo ativamente pela sua rede, como entre os recursos da rede.

malha de dados

Uma estrutura arquitetônica que fornece propriedade de dados distribuída e descentralizada com gerenciamento e governança centralizados.

minimização de dados

O princípio de coletar e processar apenas os dados estritamente necessários. Praticar a minimização de dados no Nuvem AWS pode reduzir os riscos de privacidade, os custos e a pegada de carbono de sua análise.

perímetro de dados

Um conjunto de proteções preventivas em seu AWS ambiente que ajudam a garantir que somente identidades confiáveis acessem recursos confiáveis das redes esperadas. Para obter mais informações, consulte Construindo um perímetro de dados em. AWS

pré-processamento de dados

A transformação de dados brutos em um formato que seja facilmente analisado por seu modelo de ML. O pré-processamento de dados pode significar a remoção de determinadas colunas ou linhas e o tratamento de valores ausentes, inconsistentes ou duplicados.

proveniência dos dados

O processo de rastrear a origem e o histórico dos dados ao longo de seu ciclo de vida, por exemplo, como os dados foram gerados, transmitidos e armazenados.

titular dos dados

Um indivíduo cujos dados estão sendo coletados e processados.

data warehouse

Um sistema de gerenciamento de dados que oferece suporte à inteligência comercial, como análises. Os data warehouses geralmente contêm grandes quantidades de dados históricos e geralmente são usados para consultas e análises.

linguagem de definição de dados (DDL)

Instruções ou comandos para criar ou modificar a estrutura de tabelas e objetos em um banco de dados.

linguagem de manipulação de dados (DML)

Instruções ou comandos para modificar (inserir, atualizar e excluir) informações em um banco de dados.

DDL

Consulte a linguagem de definição de banco de dados.

deep ensemble

A combinação de vários modelos de aprendizado profundo para gerar previsões. Os deep ensembles podem ser usados para produzir uma previsão mais precisa ou para estimar a incerteza nas previsões.

Aprendizado profundo

Um subcampo do ML que usa várias camadas de redes neurais artificiais para identificar o mapeamento entre os dados de entrada e as variáveis-alvo de interesse.

defense-in-depth

Uma abordagem de segurança da informação na qual uma série de mecanismos e controles de segurança são cuidadosamente distribuídos por toda a rede de computadores para proteger a confidencialidade, a integridade e a disponibilidade da rede e dos dados nela contidos. Ao adotar essa estratégia AWS, você adiciona vários controles em diferentes camadas da AWS Organizations estrutura para ajudar a proteger os recursos. Por exemplo, uma defense-in-depth abordagem pode combinar autenticação multifatorial, segmentação de rede e criptografia.

administrador delegado

Em AWS Organizations, um serviço compatível pode registrar uma conta de AWS membro para administrar as contas da organização e gerenciar as permissões desse serviço. Essa conta

é chamada de administrador delegado para esse serviço Para obter mais informações e uma lista de serviços compatíveis, consulte Serviços que funcionam com o AWS Organizations na documentação do AWS Organizations .

implantação

O processo de criar uma aplicação, novos recursos ou correções de código disponíveis no ambiente de destino. A implantação envolve a implementação de mudanças em uma base de código e, em seguida, a criação e execução dessa base de código nos ambientes da aplicação

ambiente de desenvolvimento

Veja o ambiente.

controle detectivo

Um controle de segurança projetado para detectar, registrar e alertar após a ocorrência de um evento. Esses controles são uma segunda linha de defesa, alertando você sobre eventos de segurança que contornaram os controles preventivos em vigor. Para obter mais informações, consulte Controles detectivos em Como implementar controles de segurança na AWS.

mapeamento do fluxo de valor de desenvolvimento (DVSM)

Um processo usado para identificar e priorizar restrições que afetam negativamente a velocidade e a qualidade em um ciclo de vida de desenvolvimento de software. O DVSM estende o processo de mapeamento do fluxo de valor originalmente projetado para práticas de manufatura enxuta. Ele se concentra nas etapas e equipes necessárias para criar e movimentar valor por meio do processo de desenvolvimento de software.

gêmeo digital

Uma representação virtual de um sistema real, como um prédio, fábrica, equipamento industrial ou linha de produção. Os gêmeos digitais oferecem suporte à manutenção preditiva, ao monitoramento remoto e à otimização da produção.

tabela de dimensões

Em um esquema em estrela, uma tabela menor que contém atributos de dados sobre dados quantitativos em uma tabela de fatos. Os atributos da tabela de dimensões geralmente são campos de texto ou números discretos que se comportam como texto. Esses atributos são comumente usados para restringir consultas, filtrar e rotular conjuntos de resultados.

desastre

Um evento que impede que uma workload ou sistema cumpra seus objetivos de negócios em seu local principal de implantação. Esses eventos podem ser desastres naturais, falhas técnicas ou o resultado de ações humanas, como configuração incorreta não intencional ou ataque de malware.

Recuperação de desastres (RD)

A estratégia e o processo que você usa para minimizar o tempo de inatividade e a perda de dados causados por um desastre. Para obter mais informações, consulte Recuperação de desastres de cargas de trabalho em AWS: Recuperação na nuvem no AWS Well-Architected Framework.

DML

Veja a linguagem de manipulação de banco de dados.

design orientado por domínio

Uma abordagem ao desenvolvimento de um sistema de software complexo conectando seus componentes aos domínios em evolução, ou principais metas de negócios, atendidos por cada componente. Esse conceito foi introduzido por Eric Evans em seu livro, Design orientado por domínio: lidando com a complexidade no coração do software (Boston: Addison-Wesley Professional, 2003). Para obter informações sobre como usar o design orientado por domínio com o padrão strangler fig, consulte Modernizar incrementalmente os serviços web herdados do Microsoft ASP.NET (ASMX) usando contêineres e o Amazon API Gateway.

DR

Veja a recuperação de desastres.

detecção de deriva

Rastreando desvios de uma configuração básica. Por exemplo, você pode usar AWS CloudFormation para detectar desvios nos recursos do sistema ou AWS Control Tower para detectar mudanças em seu landing zone que possam afetar a conformidade com os requisitos de governança.

DVSM

Veja o mapeamento do fluxo de valor do desenvolvimento.

E

EDA

Veja a análise exploratória de dados.

EDI

Veja intercâmbio eletrônico de dados.

computação de borda

A tecnologia que aumenta o poder computacional de dispositivos inteligentes nas bordas de uma rede de IoT. Quando comparada à computação em nuvem, a computação de ponta pode reduzir a latência da comunicação e melhorar o tempo de resposta.

intercâmbio eletrônico de dados (EDI)

A troca automatizada de documentos comerciais entre organizações. Para obter mais informações, consulte O que é intercâmbio eletrônico de dados.

Criptografia

Um processo de computação que transforma dados de texto simples, legíveis por humanos, em texto cifrado.

chave de criptografia

Uma sequência criptográfica de bits aleatórios que é gerada por um algoritmo de criptografia. As chaves podem variar em tamanho, e cada chave foi projetada para ser imprevisível e exclusiva.

endianismo

A ordem na qual os bytes são armazenados na memória do computador. Os sistemas big-endian armazenam o byte mais significativo antes. Os sistemas little-endian armazenam o byte menos significativo antes.

endpoint

Veja o endpoint do serviço.

serviço de endpoint

Um serviço que pode ser hospedado em uma nuvem privada virtual (VPC) para ser compartilhado com outros usuários. Você pode criar um serviço de endpoint com AWS PrivateLink e conceder permissões a outros diretores Contas da AWS ou a AWS Identity and Access Management (IAM).

Essas contas ou entidades principais podem se conectar ao serviço de endpoint de maneira privada criando endpoints da VPC de interface. Para obter mais informações, consulte Criar um serviço de endpoint na documentação do Amazon Virtual Private Cloud (Amazon VPC).

planejamento de recursos corporativos (ERP)

Um sistema que automatiza e gerencia os principais processos de negócios (como contabilidade, MES e gerenciamento de projetos) para uma empresa.

criptografia envelopada

O processo de criptografar uma chave de criptografia com outra chave de criptografia. Para obter mais informações, consulte Criptografia de envelope na documentação AWS Key Management Service (AWS KMS).

ambiente

Uma instância de uma aplicação em execução. Estes são tipos comuns de ambientes na computação em nuvem:

- ambiente de desenvolvimento: uma instância de uma aplicação em execução que está disponível somente para a equipe principal responsável pela manutenção da aplicação. Ambientes de desenvolvimento são usados para testar mudanças antes de promovê-las para ambientes superiores. Esse tipo de ambiente às vezes é chamado de ambiente de teste.
- ambientes inferiores: todos os ambientes de desenvolvimento para uma aplicação, como aqueles usados para compilações e testes iniciais.
- ambiente de produção: uma instância de uma aplicação em execução que os usuários finais podem acessar. Em um pipeline de CI/CD, o ambiente de produção é o último ambiente de implantação.
- ambientes superiores: todos os ambientes que podem ser acessados por usuários que não sejam a equipe principal de desenvolvimento. Isso pode incluir um ambiente de produção, ambientes de pré-produção e ambientes para testes de aceitação do usuário.

epic

Em metodologias ágeis, categorias funcionais que ajudam a organizar e priorizar seu trabalho. Os epics fornecem uma descrição de alto nível dos requisitos e das tarefas de implementação. Por exemplo, os épicos de segurança AWS da CAF incluem gerenciamento de identidade e acesso, controles de detetive, segurança de infraestrutura, proteção de dados e resposta a incidentes. Para obter mais informações sobre epics na estratégia de migração da AWS , consulte o guia de implementação do programa.

ERP

Veja o planejamento de recursos corporativos.

análise exploratória de dados (EDA)

O processo de analisar um conjunto de dados para entender suas principais características. Você coleta ou agrega dados e, em seguida, realiza investigações iniciais para encontrar padrões, detectar anomalias e verificar suposições. O EDA é realizado por meio do cálculo de estatísticas resumidas e da criação de visualizações de dados.

F

tabela de fatos

A tabela central em um esquema em estrela. Ele armazena dados quantitativos sobre as operações comerciais. Normalmente, uma tabela de fatos contém dois tipos de colunas: aquelas que contêm medidas e aquelas que contêm uma chave externa para uma tabela de dimensões.

falham rapidamente

Uma filosofia que usa testes frequentes e incrementais para reduzir o ciclo de vida do desenvolvimento. É uma parte essencial de uma abordagem ágil.

limite de isolamento de falhas

No Nuvem AWS, um limite, como uma zona de disponibilidade, Região da AWS um plano de controle ou um plano de dados, que limita o efeito de uma falha e ajuda a melhorar a resiliência das cargas de trabalho. Para obter mais informações, consulte Limites de isolamento de AWS falhas.

ramificação de recursos

Veja a filial.

recursos

Os dados de entrada usados para fazer uma previsão. Por exemplo, em um contexto de manufatura, os recursos podem ser imagens capturadas periodicamente na linha de fabricação.

importância do recurso

O quanto um recurso é importante para as previsões de um modelo. Isso geralmente é expresso como uma pontuação numérica que pode ser calculada por meio de várias técnicas, como

Shapley Additive Explanations (SHAP) e gradientes integrados. Para obter mais informações, consulte Interpretabilidade do modelo de aprendizado de máquina com AWS.

transformação de recursos

O processo de otimizar dados para o processo de ML, incluindo enriquecer dados com fontes adicionais, escalar valores ou extrair vários conjuntos de informações de um único campo de dados. Isso permite que o modelo de ML se beneficie dos dados. Por exemplo, se a data “2021-05-27 00:15:37” for dividida em “2021”, “maio”, “quinta” e “15”, isso poderá ajudar o algoritmo de aprendizado a aprender padrões diferenciados associados a diferentes componentes de dados.

solicitação rápida

Fornecer a um LLM um pequeno número de exemplos que demonstram a tarefa e o resultado desejado antes de solicitar que ele execute uma tarefa semelhante. Essa técnica é uma aplicação do aprendizado contextual, em que os modelos aprendem com exemplos (fotos) incorporados aos prompts. Solicitações rápidas podem ser eficazes para tarefas que exigem formatação, raciocínio ou conhecimento de domínio específicos. Veja também a solicitação zero-shot.

FGAC

Veja o controle de acesso refinado.

Controle de acesso refinado (FGAC)

O uso de várias condições para permitir ou negar uma solicitação de acesso.

migração flash-cut

Um método de migração de banco de dados que usa replicação contínua de dados por meio da captura de dados alterados para migrar dados no menor tempo possível, em vez de usar uma abordagem em fases. O objetivo é reduzir ao mínimo o tempo de inatividade.

FM

Veja o modelo da fundação.

modelo de fundação (FM)

Uma grande rede neural de aprendizado profundo que vem treinando em grandes conjuntos de dados generalizados e não rotulados. FMs são capazes de realizar uma ampla variedade de tarefas gerais, como entender a linguagem, gerar texto e imagens e conversar em linguagem natural. Para obter mais informações, consulte O que são modelos básicos.

G

IA generativa

Um subconjunto de modelos de IA que foram treinados em grandes quantidades de dados e que podem usar uma simples solicitação de texto para criar novos conteúdos e artefatos, como imagens, vídeos, texto e áudio. Para obter mais informações, consulte O que é IA generativa.

bloqueio geográfico

Veja as restrições geográficas.

restrições geográficas (bloqueio geográfico)

Na Amazon CloudFront, uma opção para impedir que usuários em países específicos acessem distribuições de conteúdo. É possível usar uma lista de permissões ou uma lista de bloqueios para especificar países aprovados e banidos. Para obter mais informações, consulte Restringir a distribuição geográfica do seu conteúdo na CloudFront documentação.

Fluxo de trabalho do GitFlow

Uma abordagem na qual ambientes inferiores e superiores usam ramificações diferentes em um repositório de código-fonte. O fluxo de trabalho do Gitflow é considerado legado, e o fluxo de trabalho baseado em troncos é a abordagem moderna e preferida.

imagem dourada

Um instantâneo de um sistema ou software usado como modelo para implantar novas instâncias desse sistema ou software. Por exemplo, na manufatura, uma imagem dourada pode ser usada para provisionar software em vários dispositivos e ajudar a melhorar a velocidade, a escalabilidade e a produtividade nas operações de fabricação de dispositivos.

estratégia greenfield

A ausência de infraestrutura existente em um novo ambiente. Ao adotar uma estratégia greenfield para uma arquitetura de sistema, é possível selecionar todas as novas tecnologias sem a restrição da compatibilidade com a infraestrutura existente, também conhecida como brownfield. Se estiver expandindo a infraestrutura existente, poderá combinar as estratégias brownfield e greenfield.

barreira de proteção

Uma regra de alto nível que ajuda a governar recursos, políticas e conformidade em todas as unidades organizacionais ()OUs. Barreiras de proteção preventivas impõem políticas para

garantir o alinhamento a padrões de conformidade. Elas são implementadas usando políticas de controle de serviço e limites de permissões do IAM. Barreiras de proteção detectivas detectam violações de políticas e problemas de conformidade e geram alertas para remediação. Eles são implementados usando AWS Config, AWS Security Hub, Amazon GuardDuty AWS Trusted Advisor, Amazon Inspector e verificações personalizadas AWS Lambda .

H

HA

Veja a alta disponibilidade.

migração heterogênea de bancos de dados

Migrar seu banco de dados de origem para um banco de dados de destino que usa um mecanismo de banco de dados diferente (por exemplo, Oracle para Amazon Aurora). A migração heterogênea geralmente faz parte de um esforço de redefinição da arquitetura, e converter o esquema pode ser uma tarefa complexa. O AWS fornece o AWS SCT para ajudar nas conversões de esquemas.

alta disponibilidade (HA)

A capacidade de uma workload operar continuamente, sem intervenção, em caso de desafios ou desastres. Os sistemas AH são projetados para realizar o failover automático, oferecer consistentemente desempenho de alta qualidade e lidar com diferentes cargas e falhas com impacto mínimo no desempenho.

modernização de historiador

Uma abordagem usada para modernizar e atualizar os sistemas de tecnologia operacional (OT) para melhor atender às necessidades do setor de manufatura. Um historiador é um tipo de banco de dados usado para coletar e armazenar dados de várias fontes em uma fábrica.

dados de retenção

Uma parte dos dados históricos rotulados que são retidos de um conjunto de dados usado para treinar um modelo de aprendizado de máquina. Você pode usar dados de retenção para avaliar o desempenho do modelo comparando as previsões do modelo com os dados de retenção.

migração homogênea de bancos de dados

Migrar seu banco de dados de origem para um banco de dados de destino que compartilha o mesmo mecanismo de banco de dados (por exemplo, Microsoft SQL Server para Amazon RDS para SQL Server). A migração homogênea geralmente faz parte de um esforço de redefinição da hospedagem ou da plataforma. É possível usar utilitários de banco de dados nativos para migrar o esquema.

dados quentes

Dados acessados com frequência, como dados em tempo real ou dados translacionais recentes. Esses dados normalmente exigem uma camada ou classe de armazenamento de alto desempenho para fornecer respostas rápidas às consultas.

hotfix

Uma correção urgente para um problema crítico em um ambiente de produção. Devido à sua urgência, um hotfix geralmente é feito fora do fluxo de trabalho típico de uma DevOps versão.

período de hipercuidados

Imediatamente após a substituição, o período em que uma equipe de migração gerencia e monitora as aplicações migradas na nuvem para resolver quaisquer problemas. Normalmente, a duração desse período é de 1 a 4 dias. No final do período de hipercuidados, a equipe de migração normalmente transfere a responsabilidade pelas aplicações para a equipe de operações de nuvem.

eu

IaC

Veja a infraestrutura como código.

Política baseada em identidade

Uma política anexada a um ou mais diretores do IAM que define suas permissões no Nuvem AWS ambiente.

aplicação ociosa

Uma aplicação que tem um uso médio de CPU e memória entre 5 e 20% em um período de 90 dias. Em um projeto de migração, é comum retirar essas aplicações ou retê-las on-premises.

IIoT

Veja a Internet das Coisas industrial.

infraestrutura imutável

Um modelo que implanta uma nova infraestrutura para cargas de trabalho de produção em vez de atualizar, corrigir ou modificar a infraestrutura existente. Infraestruturas imutáveis são inerentemente mais consistentes, confiáveis e previsíveis do que infraestruturas mutáveis. Para obter mais informações, consulte as melhores práticas de implantação usando infraestrutura imutável no Well-Architected AWS Framework.

VPC de entrada (admissão)

Em uma arquitetura de AWS várias contas, uma VPC que aceita, inspeciona e roteia conexões de rede de fora de um aplicativo. A Arquitetura de Referência de AWS Segurança recomenda configurar sua conta de rede com entrada, saída e inspeção VPCs para proteger a interface bidirecional entre seu aplicativo e a Internet em geral.

migração incremental

Uma estratégia de substituição na qual você migra a aplicação em pequenas partes, em vez de realizar uma única substituição completa. Por exemplo, é possível mover inicialmente apenas alguns microsserviços ou usuários para o novo sistema. Depois de verificar se tudo está funcionando corretamente, mova os microsserviços ou usuários adicionais de forma incremental até poder descomissionar seu sistema herdado. Essa estratégia reduz os riscos associados a migrações de grande porte.

Indústria 4.0

Um termo que foi introduzido por Klaus Schwab em 2016 para se referir à modernização dos processos de fabricação por meio de avanços em conectividade, dados em tempo real, automação, análise e IA/ML.

infraestrutura

Todos os recursos e ativos contidos no ambiente de uma aplicação.

Infraestrutura como código (IaC)

O processo de provisionamento e gerenciamento da infraestrutura de uma aplicação por meio de um conjunto de arquivos de configuração. A IaC foi projetada para ajudar você a centralizar o gerenciamento da infraestrutura, padronizar recursos e escalar rapidamente para que novos ambientes sejam reproduzíveis, confiáveis e consistentes.

Internet industrial das coisas (IIoT)

O uso de sensores e dispositivos conectados à Internet nos setores industriais, como manufatura, energia, automotivo, saúde, ciências biológicas e agricultura. Para obter mais informações, consulte Criando uma estratégia de transformação digital industrial da Internet das Coisas (IIoT).

VPC de inspeção

Em uma arquitetura de AWS várias contas, uma VPC centralizada que gerencia as inspeções do tráfego de rede entre VPCs (na mesma ou em diferentes Regiões da AWS) a Internet e as redes locais. A Arquitetura de Referência de AWS Segurança recomenda configurar sua conta de rede com entrada, saída e inspeção VPCs para proteger a interface bidirecional entre seu aplicativo e a Internet em geral.

Internet das Coisas (IoT)

A rede de objetos físicos conectados com sensores ou processadores incorporados que se comunicam com outros dispositivos e sistemas pela Internet ou por uma rede de comunicação local. Para obter mais informações, consulte O que é IoT?

interpretabilidade

Uma característica de um modelo de machine learning que descreve o grau em que um ser humano pode entender como as previsões do modelo dependem de suas entradas. Para obter mais informações, consulte Interpretabilidade do modelo de aprendizado de máquina com AWS.

IoT

Consulte Internet das Coisas.

Biblioteca de informações de TI (ITIL)

Um conjunto de práticas recomendadas para fornecer serviços de TI e alinhar esses serviços a requisitos de negócios. A ITIL fornece a base para o ITSM.

Gerenciamento de serviços de TI (ITSM)

Atividades associadas a design, implementação, gerenciamento e suporte de serviços de TI para uma organização. Para obter informações sobre a integração de operações em nuvem com ferramentas de ITSM, consulte o guia de integração de operações.

ITIL

Consulte a biblioteca de informações de TI.

ITSM

Veja o gerenciamento de serviços de TI.

L

controle de acesso baseado em etiqueta (LBAC)

Uma implementação do controle de acesso obrigatório (MAC) em que os usuários e os dados em si recebem explicitamente um valor de etiqueta de segurança. A interseção entre a etiqueta de segurança do usuário e a etiqueta de segurança dos dados determina quais linhas e colunas podem ser vistas pelo usuário.

zona de pouso

Uma landing zone é um AWS ambiente bem arquitetado, com várias contas, escalável e seguro. Um ponto a partir do qual suas organizações podem iniciar e implantar rapidamente workloads e aplicações com confiança em seu ambiente de segurança e infraestrutura. Para obter mais informações sobre zonas de pouso, consulte Configurar um ambiente da AWS com várias contas seguro e escalável.

modelo de linguagem grande (LLM)

Um modelo de IA de aprendizado profundo que é pré-treinado em uma grande quantidade de dados. Um LLM pode realizar várias tarefas, como responder perguntas, resumir documentos, traduzir texto para outros idiomas e completar frases. Para obter mais informações, consulte O que são LLMs.

migração de grande porte

Uma migração de 300 servidores ou mais.

LBAC

Veja controle de acesso baseado em etiquetas.

privilégio mínimo

A prática recomendada de segurança de conceder as permissões mínimas necessárias para executar uma tarefa. Para obter mais informações, consulte Aplicar permissões de privilégios mínimos na documentação do IAM.

mover sem alterações (lift-and-shift)

Veja 7 Rs.

sistema little-endian

Um sistema que armazena o byte menos significativo antes. Veja também endianness.

LLM

Veja um modelo de linguagem grande.

ambientes inferiores

Veja o ambiente.

M

machine learning (ML)

Um tipo de inteligência artificial que usa algoritmos e técnicas para reconhecimento e aprendizado de padrões. O ML analisa e aprende com dados gravados, por exemplo, dados da Internet das Coisas (IoT), para gerar um modelo estatístico baseado em padrões. Para obter mais informações, consulte Machine learning.

ramificação principal

Veja a filial.

malware

Software projetado para comprometer a segurança ou a privacidade do computador. O malware pode interromper os sistemas do computador, vazar informações confidenciais ou obter acesso não autorizado. Exemplos de malware incluem vírus, worms, ransomware, cavalos de Tróia, spyware e keyloggers.

serviços gerenciados

Serviços da AWS para o qual AWS opera a camada de infraestrutura, o sistema operacional e as plataformas, e você acessa os endpoints para armazenar e recuperar dados. O Amazon Simple Storage Service (Amazon S3) e o Amazon DynamoDB são exemplos de serviços gerenciados. Eles também são conhecidos como serviços abstratos.

sistema de execução de manufatura (MES)

Um sistema de software para rastrear, monitorar, documentar e controlar processos de produção que convertem matérias-primas em produtos acabados no chão de fábrica.

MAP

Consulte Migration Acceleration Program.

mecanismo

Um processo completo no qual você cria uma ferramenta, impulsiona a adoção da ferramenta e, em seguida, inspeciona os resultados para fazer ajustes. Um mecanismo é um ciclo que se reforça e se aprimora à medida que opera. Para obter mais informações, consulte Construindo mecanismos no AWS Well-Architected Framework.

conta de membro

Todos, Contas da AWS exceto a conta de gerenciamento, que fazem parte de uma organização em AWS Organizations. Uma conta só pode ser membro de uma organização de cada vez.

MES

Veja o sistema de execução de manufatura.

Transporte de telemetria de enfileiramento de mensagens (MQTT)

Um protocolo de comunicação leve machine-to-machine (M2M), baseado no padrão de publicação/assinatura, para dispositivos de IoT com recursos limitados.

microsserviço

Um serviço pequeno e independente que se comunica de forma bem definida APIs e normalmente é de propriedade de equipes pequenas e independentes. Por exemplo, um sistema de seguradora pode incluir microsserviços que mapeiam as capacidades comerciais, como vendas ou marketing, ou subdomínios, como compras, reclamações ou análises. Os benefícios dos microsserviços incluem agilidade, escalabilidade flexível, fácil implantação, código reutilizável e resiliência. Para obter mais informações, consulte Integração de microsserviços usando serviços sem AWS servidor.

arquitetura de microsserviços

Uma abordagem à criação de aplicações com componentes independentes que executam cada processo de aplicação como um microsserviço. Esses microsserviços se comunicam por meio

de uma interface bem definida usando leveza. APIs Cada microsserviço nessa arquitetura pode ser atualizado, implantado e escalado para atender à demanda por funções específicas de uma aplicação. Para obter mais informações, consulte Implementação de microsserviços em. AWS

Programa de Aceleração da Migração (MAP)

Um AWS programa que fornece suporte de consultoria, treinamento e serviços para ajudar as organizações a criar uma base operacional sólida para migrar para a nuvem e ajudar a compensar o custo inicial das migrações. O MAP inclui uma metodologia de migração para executar migrações legadas de forma metódica e um conjunto de ferramentas para automatizar e acelerar cenários comuns de migração.

migração em escala

O processo de mover a maior parte do portfólio de aplicações para a nuvem em ondas, com mais aplicações sendo movidas em um ritmo mais rápido a cada onda. Essa fase usa as práticas recomendadas e lições aprendidas nas fases anteriores para implementar uma fábrica de migração de equipes, ferramentas e processos para agilizar a migração de workloads por meio de automação e entrega ágeis. Esta é a terceira fase da estratégia de migração para a AWS.

fábrica de migração

Equipes multifuncionais que simplificam a migração de workloads por meio de abordagens automatizadas e ágeis. As equipes da fábrica de migração geralmente incluem operações, analistas e proprietários de negócios, engenheiros de migração, desenvolvedores e DevOps profissionais que trabalham em sprints. Entre 20 e 50% de um portfólio de aplicações corporativas consiste em padrões repetidos que podem ser otimizados por meio de uma abordagem de fábrica. Para obter mais informações, consulte discussão sobre fábricas de migração e o guia do Cloud Migration Factory neste conjunto de conteúdo.

metadados de migração

As informações sobre a aplicação e o servidor necessárias para concluir a migração. Cada padrão de migração exige um conjunto de metadados de migração diferente. Exemplos de metadados de migração incluem a sub-rede, o grupo de segurança e AWS a conta de destino.

padrão de migração

Uma tarefa de migração repetível que detalha a estratégia de migração, o destino da migração e a aplicação ou o serviço de migração usado. Exemplo: rehospede a migração para a Amazon EC2 com o AWS Application Migration Service.

Avaliação de Portfólio para Migração (MPA)

Uma ferramenta on-line que fornece informações para validar o caso de negócios para migrar para o. Nuvem AWS O MPA fornece avaliação detalhada do portfólio (dimensionamento correto do servidor, preços, comparações de TCO, análise de custos de migração), bem como planejamento de migração (análise e coleta de dados de aplicações, agrupamento de aplicações, priorização de migração e planejamento de ondas). A ferramenta MPA (requer login) está disponível gratuitamente para todos os AWS consultores e consultores parceiros da APN.

Avaliação de Preparação para Migração (MRA)

O processo de obter insights sobre o status de prontidão de uma organização para a nuvem, identificar pontos fortes e fracos e criar um plano de ação para fechar as lacunas identificadas, usando o CAF. AWS Para mais informações, consulte o guia de preparação para migração. A MRA é a primeira fase da estratégia de migração para a AWS.

estratégia de migração

A abordagem usada para migrar uma carga de trabalho para o. Nuvem AWS Para obter mais informações, consulte a entrada de 7 Rs neste glossário e consulte Mobilize sua organização para acelerar migrações em grande escala.

ML

Veja o aprendizado de máquina.

modernização

Transformar uma aplicação desatualizada (herdada ou monolítica) e sua infraestrutura em um sistema ágil, elástico e altamente disponível na nuvem para reduzir custos, ganhar eficiência e aproveitar as inovações. Para obter mais informações, consulte Estratégia para modernizar aplicativos no Nuvem AWS.

avaliação de preparação para modernização

Uma avaliação que ajuda a determinar a preparação para modernização das aplicações de uma organização. Ela identifica benefícios, riscos e dependências e determina o quão bem a organização pode acomodar o estado futuro dessas aplicações. O resultado da avaliação é um esquema da arquitetura de destino, um roteiro que detalha as fases de desenvolvimento e os marcos do processo de modernização e um plano de ação para abordar as lacunas identificadas. Para obter mais informações, consulte Avaliação da prontidão para modernização de aplicativos no. Nuvem AWS

aplicações monolíticas (monólitos)

Aplicações que são executadas como um único serviço com processos fortemente acoplados. As aplicações monolíticas apresentam várias desvantagens. Se um recurso da aplicação apresentar um aumento na demanda, toda a arquitetura deverá ser escalada. Adicionar ou melhorar os recursos de uma aplicação monolítica também se torna mais complexo quando a base de código cresce. Para resolver esses problemas, é possível criar uma arquitetura de microsserviços. Para obter mais informações, consulte Decompor monólitos em microsserviços.

MAPA

Consulte Avaliação do portfólio de migração.

MQTT

Consulte Transporte de telemetria de enfileiramento de mensagens.

classificação multiclasse

Um processo que ajuda a gerar previsões para várias classes (prevendo um ou mais de dois resultados). Por exemplo, um modelo de ML pode perguntar “Este produto é um livro, um carro ou um telefone?” ou “Qual categoria de produtos é mais interessante para este cliente?”

infraestrutura mutável

Um modelo que atualiza e modifica a infraestrutura existente para cargas de trabalho de produção. Para melhorar a consistência, confiabilidade e previsibilidade, o AWS Well-Architected Framework recomenda o uso de infraestrutura imutável como uma prática recomendada.

O

OAC

Veja o controle de acesso de origem.

CARVALHO

Veja a identidade de acesso de origem.

OCM

Veja o gerenciamento de mudanças organizacionais.

migração offline

Um método de migração no qual a workload de origem é desativada durante o processo de migração. Esse método envolve tempo de inatividade prolongado e geralmente é usado para workloads pequenas e não críticas.

OI

Veja a integração de operações.

OLA

Veja o contrato em nível operacional.

migração online

Um método de migração no qual a workload de origem é copiada para o sistema de destino sem ser colocada offline. As aplicações conectadas à workload podem continuar funcionando durante a migração. Esse método envolve um tempo de inatividade nulo ou mínimo e normalmente é usado para workloads essenciais para a produção.

OPC-UA

Consulte Comunicação de processo aberto — Arquitetura unificada.

Comunicação de processo aberto — Arquitetura unificada (OPC-UA)

Um protocolo de comunicação machine-to-machine (M2M) para automação industrial. O OPC-UA fornece um padrão de interoperabilidade com esquemas de criptografia, autenticação e autorização de dados.

acordo de nível operacional (OLA)

Um acordo que esclarece o que os grupos funcionais de TI prometem oferecer uns aos outros para apoiar um acordo de serviço (SLA).

análise de prontidão operacional (ORR)

Uma lista de verificação de perguntas e melhores práticas associadas que ajudam você a entender, avaliar, prevenir ou reduzir o escopo de incidentes e possíveis falhas. Para obter mais informações, consulte Operational Readiness Reviews (ORR) no Well-Architected AWS Framework.

tecnologia operacional (OT)

Sistemas de hardware e software que funcionam com o ambiente físico para controlar operações, equipamentos e infraestrutura industriais. Na manufatura, a integração dos sistemas OT e de tecnologia da informação (TI) é o foco principal das transformações da Indústria 4.0.

integração de operações (OI)

O processo de modernização das operações na nuvem, que envolve planejamento de preparação, automação e integração. Para obter mais informações, consulte o guia de integração de operações.

trilha organizacional

Uma trilha criada por ela AWS CloudTrail registra todos os eventos de todos Contas da AWS em uma organização em AWS Organizations. Essa trilha é criada em cada Conta da AWS que faz parte da organização e monitora a atividade em cada conta. Para obter mais informações, consulte Criação de uma trilha para uma organização na CloudTrail documentação.

gerenciamento de alterações organizacionais (OCM)

Uma estrutura para gerenciar grandes transformações de negócios disruptivas de uma perspectiva de pessoas, cultura e liderança. O OCM ajuda as organizações a se prepararem e fazerem a transição para novos sistemas e estratégias, acelerando a adoção de alterações, abordando questões de transição e promovendo mudanças culturais e organizacionais. Na estratégia de AWS migração, essa estrutura é chamada de aceleração de pessoas, devido à velocidade de mudança exigida nos projetos de adoção da nuvem. Para obter mais informações, consulte o guia do OCM.

controle de acesso de origem (OAC)

Em CloudFront, uma opção aprimorada para restringir o acesso para proteger seu conteúdo do Amazon Simple Storage Service (Amazon S3). O OAC oferece suporte a todos os buckets S3 Regiões da AWS, criptografia do lado do servidor com AWS KMS (SSE-KMS) e solicitações dinâmicas ao bucket S3. `PUT` `DELETE`

Identidade do acesso de origem (OAI)

Em CloudFront, uma opção para restringir o acesso para proteger seu conteúdo do Amazon S3. Quando você usa o OAI, CloudFront cria um principal com o qual o Amazon S3 pode se autenticar. Os diretores autenticados podem acessar o conteúdo em um bucket do S3 somente por meio de uma distribuição específica. CloudFront Veja também OAC, que fornece um controle de acesso mais granular e aprimorado.

ORR

Veja a análise de prontidão operacional.

OT

Veja a tecnologia operacional.

VPC de saída (egresso)

Em uma arquitetura de AWS várias contas, uma VPC que gerencia conexões de rede que são iniciadas de dentro de um aplicativo. A Arquitetura de Referência de AWS Segurança recomenda configurar sua conta de rede com entrada, saída e inspeção VPCs para proteger a interface bidirecional entre seu aplicativo e a Internet em geral.

P

limite de permissões

Uma política de gerenciamento do IAM anexada a entidades principais do IAM para definir as permissões máximas que o usuário ou perfil podem ter. Para obter mais informações, consulte Limites de permissões na documentação do IAM.

Informações de identificação pessoal (PII)

Informações que, quando visualizadas diretamente ou combinadas com outros dados relacionados, podem ser usadas para inferir razoavelmente a identidade de um indivíduo. Exemplos de PII incluem nomes, endereços e informações de contato.

PII

Veja as informações de identificação pessoal.

manual

Um conjunto de etapas predefinidas que capturam o trabalho associado às migrações, como a entrega das principais funções operacionais na nuvem. Um manual pode assumir a forma de scripts, runbooks automatizados ou um resumo dos processos ou etapas necessários para operar seu ambiente modernizado.

PLC

Consulte controlador lógico programável.

AMEIXA

Veja o gerenciamento do ciclo de vida do produto.

política

Um objeto que pode definir permissões (consulte a política baseada em identidade), especificar as condições de acesso (consulte a política baseada em recursos) ou definir as permissões máximas para todas as contas em uma organização em AWS Organizations (consulte a política de controle de serviços).

persistência poliglota

Escolher de forma independente a tecnologia de armazenamento de dados de um microsserviço com base em padrões de acesso a dados e outros requisitos. Se seus microsserviços tiverem a mesma tecnologia de armazenamento de dados, eles poderão enfrentar desafios de implementação ou apresentar baixa performance. Os microsserviços serão implementados com mais facilidade e alcançarão performance e escalabilidade melhores se usarem o armazenamento de dados mais bem adaptado às suas necessidades. Para obter mais informações, consulte Habilitar a persistência de dados em microsserviços.

avaliação do portfólio

Um processo de descobrir, analisar e priorizar o portfólio de aplicações para planejar a migração. Para obter mais informações, consulte Avaliar a preparação para a migração.

predicado

Uma condição de consulta que retorna `true` ou`false`, normalmente localizada em uma `WHERE` cláusula.

pressão de predicados

Uma técnica de otimização de consulta de banco de dados que filtra os dados na consulta antes da transferência. Isso reduz a quantidade de dados que devem ser recuperados e processados do banco de dados relacional e melhora o desempenho das consultas.

controle preventivo

Um controle de segurança projetado para evitar que um evento ocorra. Esses controles são a primeira linha de defesa para ajudar a evitar acesso não autorizado ou alterações indesejadas em sua rede. Para obter mais informações, consulte Controles preventivos em Como implementar controles de segurança na AWS.

principal (entidade principal)

Uma entidade AWS que pode realizar ações e acessar recursos. Essa entidade geralmente é um usuário raiz para um Conta da AWS, uma função do IAM ou um usuário. Para obter mais informações, consulte Entidade principal em Termos e conceitos de perfis na documentação do IAM.

privacidade por design

Uma abordagem de engenharia de sistema que leva em consideração a privacidade em todo o processo de desenvolvimento.

zonas hospedadas privadas

Um contêiner que contém informações sobre como você deseja que o Amazon Route 53 responda às consultas de DNS para um domínio e seus subdomínios em um ou mais. VPCs Para obter mais informações, consulte Como trabalhar com zonas hospedadas privadas na documentação do Route 53.

controle proativo

Um controle de segurança projetado para impedir a implantação de recursos não compatíveis. Esses controles examinam os recursos antes de serem provisionados. Se o recurso não estiver em conformidade com o controle, ele não será provisionado. Para obter mais informações, consulte o guia de referência de controles na AWS Control Tower documentação e consulte Controles proativos em Implementação de controles de segurança em AWS.

gerenciamento do ciclo de vida do produto (PLM)

O gerenciamento de dados e processos de um produto em todo o seu ciclo de vida, desde o design, desenvolvimento e lançamento, passando pelo crescimento e maturidade, até o declínio e a remoção.

ambiente de produção

Veja o ambiente.

controlador lógico programável (PLC)

Na fabricação, um computador altamente confiável e adaptável que monitora as máquinas e automatiza os processos de fabricação.

encadeamento imediato

Usando a saída de um prompt do LLM como entrada para o próximo prompt para gerar respostas melhores. Essa técnica é usada para dividir uma tarefa complexa em subtarefas ou para refinar ou expandir iterativamente uma resposta preliminar. Isso ajuda a melhorar a precisão e a relevância das respostas de um modelo e permite resultados mais granulares e personalizados.

pseudonimização

O processo de substituir identificadores pessoais em um conjunto de dados por valores de espaço reservado. A pseudonimização pode ajudar a proteger a privacidade pessoal. Os dados pseudonimizados ainda são considerados dados pessoais.

publish/subscribe (pub/sub)

Um padrão que permite comunicações assíncronas entre microsserviços para melhorar a escalabilidade e a capacidade de resposta. Por exemplo, em um MES baseado em microsserviços, um microsserviço pode publicar mensagens de eventos em um canal no qual outros microsserviços possam se inscrever. O sistema pode adicionar novos microsserviços sem alterar o serviço de publicação.

Q

plano de consulta

Uma série de etapas, como instruções, usadas para acessar os dados em um sistema de banco de dados relacional SQL.

regressão de planos de consultas

Quando um otimizador de serviço de banco de dados escolhe um plano menos adequado do que escolhia antes de uma determinada alteração no ambiente de banco de dados ocorrer. Isso pode ser causado por alterações em estatísticas, restrições, configurações do ambiente, associações de parâmetros de consulta e atualizações do mecanismo de banco de dados.

R

Matriz RACI

Veja responsável, responsável, consultado, informado (RACI).

RAG

Consulte Geração Aumentada de Recuperação.

ransomware

Um software mal-intencionado desenvolvido para bloquear o acesso a um sistema ou dados de computador até que um pagamento seja feito.

Matriz RASCI

Veja responsável, responsável, consultado, informado (RACI).

RCAC

Veja o controle de acesso por linha e coluna.

réplica de leitura

Uma cópia de um banco de dados usada somente para leitura. É possível encaminhar consultas para a réplica de leitura e reduzir a carga no banco de dados principal.

rearquiteta

Veja 7 Rs.

objetivo de ponto de recuperação (RPO).

O máximo período de tempo aceitável desde o último ponto de recuperação de dados. Isso determina o que é considerado uma perda aceitável de dados entre o último ponto de recuperação e a interrupção do serviço.

objetivo de tempo de recuperação (RTO)

O máximo atraso aceitável entre a interrupção e a restauração do serviço.

refatorar

Veja 7 Rs.

Região

Uma coleção de AWS recursos em uma área geográfica. Cada um Região da AWS é isolado e independente dos outros para fornecer tolerância a falhas, estabilidade e resiliência. Para obter mais informações, consulte Especificar o que Regiões da AWS sua conta pode usar.

regressão

Uma técnica de ML que prevê um valor numérico. Por exemplo, para resolver o problema de “Por qual preço esta casa será vendida?” um modelo de ML pode usar um modelo de regressão linear para prever o preço de venda de uma casa com base em fatos conhecidos sobre a casa (por exemplo, a metragem quadrada).

redefinir a hospedagem

Veja 7 Rs.

versão

Em um processo de implantação, o ato de promover mudanças em um ambiente de produção.

realocar

Veja 7 Rs.

redefinir a plataforma

Veja 7 Rs.

recomprar

Veja 7 Rs.

resiliência

A capacidade de um aplicativo de resistir ou se recuperar de interrupções. Alta disponibilidade e recuperação de desastres são considerações comuns ao planejar a resiliência no. Nuvem AWS Para obter mais informações, consulte Nuvem AWS Resiliência.

política baseada em recurso

Uma política associada a um recurso, como um bucket do Amazon S3, um endpoint ou uma chave de criptografia. Esse tipo de política especifica quais entidades principais têm acesso permitido, ações válidas e quaisquer outras condições que devem ser atendidas.

matriz responsável, accountable, consultada, informada (RACI)

Uma matriz que define as funções e responsabilidades de todas as partes envolvidas nas atividades de migração e nas operações de nuvem. O nome da matriz é derivado dos tipos de responsabilidade definidos na matriz: responsável (R), responsabilizável (A), consultado (C) e informado (I). O tipo de suporte (S) é opcional. Se você incluir suporte, a matriz será chamada de matriz RASCI e, se excluir, será chamada de matriz RACI.

controle responsivo

Um controle de segurança desenvolvido para conduzir a remediação de eventos adversos ou desvios em relação à linha de base de segurança. Para obter mais informações, consulte Controles responsivos em Como implementar controles de segurança na AWS.

reter

Veja 7 Rs.

aposentar-se

Veja 7 Rs.

Geração Aumentada de Recuperação (RAG)

Uma tecnologia de IA generativa na qual um LLM faz referência a uma fonte de dados autorizada que está fora de suas fontes de dados de treinamento antes de gerar uma resposta. Por exemplo, um modelo RAG pode realizar uma pesquisa semântica na base de conhecimento ou nos dados personalizados de uma organização. Para obter mais informações, consulte O que é RAG.

alternância

O processo de atualizar periodicamente um segredo para dificultar o acesso das credenciais por um invasor.

controle de acesso por linha e coluna (RCAC)

O uso de expressões SQL básicas e flexíveis que tenham regras de acesso definidas. O RCAC consiste em permissões de linha e máscaras de coluna.

RPO

Veja o objetivo do ponto de recuperação.

RTO

Veja o objetivo do tempo de recuperação.

runbook

Um conjunto de procedimentos manuais ou automatizados necessários para realizar uma tarefa específica. Eles são normalmente criados para agilizar operações ou procedimentos repetitivos com altas taxas de erro.

S

SAML 2.0

Um padrão aberto que muitos provedores de identidade (IdPs) usam. Esse recurso permite o login único federado (SSO), para que os usuários possam fazer login AWS Management Console ou chamar as operações da AWS API sem que você precise criar um usuário no IAM para todos em sua organização. Para obter mais informações sobre a federação baseada em SAML 2.0, consulte Sobre a federação baseada em SAML 2.0 na documentação do IAM.

SCADA

Veja controle de supervisão e aquisição de dados.

SCP

Veja a política de controle de serviços.

secret

Em AWS Secrets Manager, informações confidenciais ou restritas, como uma senha ou credenciais de usuário, que você armazena de forma criptografada. Ele consiste no valor secreto e em seus metadados. O valor secreto pode ser binário, uma única string ou várias strings. Para obter mais informações, consulte O que há em um segredo do Secrets Manager? na documentação do Secrets Manager.

segurança por design

Uma abordagem de engenharia de sistemas que leva em conta a segurança em todo o processo de desenvolvimento.

controle de segurança

Uma barreira de proteção técnica ou administrativa que impede, detecta ou reduz a capacidade de uma ameaça explorar uma vulnerabilidade de segurança. Existem quatro tipos principais de controles de segurança: preventivos, detectivos, responsivos e proativos.

fortalecimento da segurança

O processo de reduzir a superfície de ataque para torná-la mais resistente a ataques. Isso pode incluir ações como remover recursos que não são mais necessários, implementar a prática recomendada de segurança de conceder privilégios mínimos ou desativar recursos desnecessários em arquivos de configuração.

sistema de gerenciamento de eventos e informações de segurança (SIEM)

Ferramentas e serviços que combinam sistemas de gerenciamento de informações de segurança (SIM) e gerenciamento de eventos de segurança (SEM). Um sistema SIEM coleta, monitora e analisa dados de servidores, redes, dispositivos e outras fontes para detectar ameaças e violações de segurança e gerar alertas.

automação de resposta de segurança

Uma ação predefinida e programada projetada para responder ou remediar automaticamente um evento de segurança. Essas automações servem como controles de segurança responsivos ou detectivos que ajudam você a implementar as melhores práticas AWS de segurança. Exemplos de ações de resposta automatizada incluem a modificação de um grupo de segurança da VPC, a correção de uma instância EC2 da Amazon ou a rotação de credenciais.

Criptografia do lado do servidor

Criptografia dos dados em seu destino, por AWS service (Serviço da AWS) quem os recebe.

política de controle de serviços (SCP)

Uma política que fornece controle centralizado sobre as permissões de todas as contas em uma organização em AWS Organizations. SCPs defina barreiras ou estabeleça limites nas ações que um administrador pode delegar a usuários ou funções. Você pode usar SCPs como listas de permissão ou listas de negação para especificar quais serviços ou ações são permitidos ou proibidos. Para obter mais informações, consulte Políticas de controle de serviço na AWS Organizations documentação.

service endpoint (endpoint de serviço)

O URL do ponto de entrada para um AWS service (Serviço da AWS). Você pode usar o endpoint para se conectar programaticamente ao serviço de destino. Para obter mais informações, consulte Endpoints do AWS service (Serviço da AWS) na Referência geral da AWS.

acordo de serviço (SLA)

Um acordo que esclarece o que uma equipe de TI promete fornecer aos clientes, como tempo de atividade e performance do serviço.

indicador de nível de serviço (SLI)

Uma medida de um aspecto de desempenho de um serviço, como taxa de erro, disponibilidade ou taxa de transferência.

objetivo de nível de serviço (SLO)

Uma métrica alvo que representa a integridade de um serviço, conforme medida por um indicador de nível de serviço.

modelo de responsabilidade compartilhada

Um modelo que descreve a responsabilidade com a qual você compartilha AWS pela segurança e conformidade na nuvem. AWS é responsável pela segurança da nuvem, enquanto você é responsável pela segurança na nuvem. Para obter mais informações, consulte o Modelo de responsabilidade compartilhada.

SIEM

Veja informações de segurança e sistema de gerenciamento de eventos.

ponto único de falha (SPOF)

Uma falha em um único componente crítico de um aplicativo que pode interromper o sistema.

SLA

Veja o contrato de nível de serviço.

ESGUIO

Veja o indicador de nível de serviço.

SLO

Veja o objetivo do nível de serviço.

split-and-seed modelo

Um padrão para escalar e acelerar projetos de modernização. À medida que novos recursos e lançamentos de produtos são definidos, a equipe principal se divide para criar novas equipes de produtos. Isso ajuda a escalar os recursos e os serviços da sua organização, melhora a produtividade do desenvolvedor e possibilita inovações rápidas. Para obter mais informações, consulte Abordagem em fases para modernizar aplicativos no. Nuvem AWS

CUSPE

Veja um único ponto de falha.

esquema de estrelas

Uma estrutura organizacional de banco de dados que usa uma grande tabela de fatos para armazenar dados transacionais ou medidos e usa uma ou mais tabelas dimensionais menores

para armazenar atributos de dados. Essa estrutura foi projetada para uso em um data warehouse ou para fins de inteligência comercial.

padrão strangler fig

Uma abordagem à modernização de sistemas monolíticos que consiste em reescrever e substituir incrementalmente a funcionalidade do sistema até que o sistema herdado possa ser desativado. Esse padrão usa a analogia de uma videira que cresce e se torna uma árvore estabelecida e, eventualmente, supera e substitui sua hospedeira. O padrão foi apresentado por Martin Fowler como forma de gerenciar riscos ao reescrever sistemas monolíticos. Para ver um exemplo de como aplicar esse padrão, consulte Modernizar incrementalmente os serviços Web herdados do Microsoft ASP.NET (ASMX) usando contêineres e o Amazon API Gateway.

sub-rede

Um intervalo de endereços IP na VPC. Cada sub-rede fica alocada em uma única zona de disponibilidade.

controle de supervisão e aquisição de dados (SCADA)

Na manufatura, um sistema que usa hardware e software para monitorar ativos físicos e operações de produção.

symmetric encryption (criptografia simétrica)

Um algoritmo de criptografia que usa a mesma chave para criptografar e descriptografar dados.

testes sintéticos

Testar um sistema de forma que simule as interações do usuário para detectar possíveis problemas ou monitorar o desempenho. Você pode usar o Amazon CloudWatch Synthetics para criar esses testes.

prompt do sistema

Uma técnica para fornecer contexto, instruções ou diretrizes a um LLM para direcionar seu comportamento. Os prompts do sistema ajudam a definir o contexto e estabelecer regras para interações com os usuários.

T

tags

Pares de valores-chave que atuam como metadados para organizar seus recursos. AWS As tags podem ajudar você a gerenciar, identificar, organizar, pesquisar e filtrar recursos. Para obter mais informações, consulte Marcar seus recursos do AWS.

variável-alvo

O valor que você está tentando prever no ML supervisionado. Ela também é conhecida como variável de resultado. Por exemplo, em uma configuração de fabricação, a variável-alvo pode ser um defeito do produto.

lista de tarefas

Uma ferramenta usada para monitorar o progresso por meio de um runbook. Uma lista de tarefas contém uma visão geral do runbook e uma lista de tarefas gerais a serem concluídas. Para cada tarefa geral, ela inclui o tempo estimado necessário, o proprietário e o progresso.

ambiente de teste

Veja o ambiente.

treinamento

O processo de fornecer dados para que seu modelo de ML aprenda. Os dados de treinamento devem conter a resposta correta. O algoritmo de aprendizado descobre padrões nos dados de treinamento que mapeiam os atributos dos dados de entrada no destino (a resposta que você deseja prever). Ele gera um modelo de ML que captura esses padrões. Você pode usar o modelo de ML para obter previsões de novos dados cujo destino você não conhece.

gateway de trânsito

Um hub de trânsito de rede que você pode usar para interconectar sua rede com VPCs a rede local. Para obter mais informações, consulte O que é um gateway de trânsito na AWS Transit Gateway documentação.

fluxo de trabalho baseado em troncos

Uma abordagem na qual os desenvolvedores criam e testam recursos localmente em uma ramificação de recursos e, em seguida, mesclam essas alterações na ramificação principal. A

ramificação principal é então criada para os ambientes de desenvolvimento, pré-produção e produção, sequencialmente.

Acesso confiável

Conceder permissões a um serviço que você especifica para realizar tarefas em sua organização AWS Organizations e em suas contas em seu nome. O serviço confiável cria um perfil vinculado ao serviço em cada conta, quando esse perfil é necessário, para realizar tarefas de gerenciamento para você. Para obter mais informações, consulte Usando AWS Organizations com outros AWS serviços na AWS Organizations documentação.

tuning (ajustar)

Alterar aspectos do processo de treinamento para melhorar a precisão do modelo de ML. Por exemplo, você pode treinar o modelo de ML gerando um conjunto de rótulos, adicionando rótulos e repetindo essas etapas várias vezes em configurações diferentes para otimizar o modelo.

equipe de duas pizzas

Uma pequena DevOps equipe que você pode alimentar com duas pizzas. Uma equipe de duas pizzas garante a melhor oportunidade possível de colaboração no desenvolvimento de software.

U

incerteza

Um conceito que se refere a informações imprecisas, incompletas ou desconhecidas que podem minar a confiabilidade dos modelos preditivos de ML. Há dois tipos de incertezas: a incerteza epistêmica é causada por dados limitados e incompletos, enquanto a incerteza aleatória é causada pelo ruído e pela aleatoriedade inerentes aos dados. Para obter mais informações, consulte o guia Como quantificar a incerteza em sistemas de aprendizado profundo.

tarefas indiferenciadas

Também conhecido como trabalho pesado, trabalho necessário para criar e operar um aplicativo, mas que não fornece valor direto ao usuário final nem oferece vantagem competitiva. Exemplos de tarefas indiferenciadas incluem aquisição, manutenção e planejamento de capacidade.

ambientes superiores

Veja o ambiente.

V

aspiração

Uma operação de manutenção de banco de dados que envolve limpeza após atualizações incrementais para recuperar armazenamento e melhorar a performance.

controle de versões

Processos e ferramentas que rastreiam mudanças, como alterações no código-fonte em um repositório.

emparelhamento da VPC

Uma conexão entre duas VPCs que permite rotear o tráfego usando endereços IP privados. Para ter mais informações, consulte O que é emparelhamento de VPC? na documentação da Amazon VPC.

Vulnerabilidade

Uma falha de software ou hardware que compromete a segurança do sistema.

W

cache quente

Um cache de buffer que contém dados atuais e relevantes que são acessados com frequência. A instância do banco de dados pode ler do cache do buffer, o que é mais rápido do que ler da memória principal ou do disco.

dados mornos

Dados acessados raramente. Ao consultar esse tipo de dados, consultas moderadamente lentas geralmente são aceitáveis.

função de janela

Uma função SQL que executa um cálculo em um grupo de linhas que se relacionam de alguma forma com o registro atual. As funções de janela são úteis para processar tarefas, como calcular uma média móvel ou acessar o valor das linhas com base na posição relativa da linha atual.

workload

Uma coleção de códigos e recursos que geram valor empresarial, como uma aplicação voltada para o cliente ou um processo de back-end.

workstreams

Grupos funcionais em um projeto de migração que são responsáveis por um conjunto específico de tarefas. Cada workstream é independente, mas oferece suporte aos outros workstreams do projeto. Por exemplo, o workstream de portfólio é responsável por priorizar aplicações, planejar ondas e coletar metadados de migração. O workstream de portfólio entrega esses ativos ao workstream de migração, que então migra os servidores e as aplicações.

MINHOCA

Veja escrever uma vez, ler muitas.

WQF

Consulte Estrutura de qualificação AWS da carga de trabalho.

escreva uma vez, leia muitas (WORM)

Um modelo de armazenamento que grava dados uma única vez e evita que os dados sejam excluídos ou modificados. Os usuários autorizados podem ler os dados quantas vezes forem necessárias, mas não podem alterá-los. Essa infraestrutura de armazenamento de dados é considerada imutável.

Z

exploração de dia zero

Um ataque, geralmente malware, que tira proveito de uma vulnerabilidade de dia zero.

vulnerabilidade de dia zero

Uma falha ou vulnerabilidade não mitigada em um sistema de produção. Os agentes de ameaças podem usar esse tipo de vulnerabilidade para atacar o sistema. Os desenvolvedores frequentemente ficam cientes da vulnerabilidade como resultado do ataque.

aviso zero-shot

Fornecer a um LLM instruções para realizar uma tarefa, mas sem exemplos (fotos) que possam ajudar a orientá-la. O LLM deve usar seu conhecimento pré-treinado para lidar com a tarefa. A

eficácia da solicitação zero depende da complexidade da tarefa e da qualidade da solicitação. Veja também a solicitação de algumas fotos.

aplicação zumbi

Uma aplicação que tem um uso médio de CPU e memória inferior a 5%. Em um projeto de migração, é comum retirar essas aplicações.

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.