AWSOrientação prescritiva

Usando registros de decisão arquitetônica para simplificar a tomada de decisões técnicas para um projeto de desenvolvimento de software

AWSOrientação prescritiva: Usando registros de decisão arquitetônica para simplificar a tomada de decisões técnicas para um projeto de desenvolvimento de software

Table of Contents

Usando registros de decisão arquitetônica para simplificar a tomada de decisões técnicas para um projeto de desenvolvimento de software

Darius Kunce e Dominik Goby, Arquitetos de Aplicação,AWSServiços profissionais

Março de 2022

Este guia apresenta o processo de registros de decisão arquitetônica (ADR) para projetos de engenharia de software. Os ADRs dão suporte ao alinhamento da equipe, documentam orientações estratégicas para um projeto ou produto e reduzem os esforços recorrentes e demorados de tomada de decisão.

Durante o desenvolvimento de projetos e produtos, as equipes de engenharia de software precisam tomar decisões arquitetônicas para atingir seus objetivos. Essas decisões podem ser técnicas, como decidir usar o padrão de segregação de responsabilidade de consulta de comando (CQRS) ou relacionadas ao processo, como decidir usar o GitFlow fluxo de trabalho para gerenciar o código fonte. Tomar essas decisões é um processo demorado e difícil. As equipes devem justificar, documentar e comunicar essas decisões às partes interessadas relevantes.

Três antipadrões principais geralmente surgem ao tomar decisões arquitetônicas:

- Nenhuma decisão é tomada, por medo de fazer a escolha errada.
- Uma decisão é tomada sem qualquer justificativa, e as pessoas não entendem por que foi tomada. Isso resulta no mesmo tópico sendo discutido várias vezes.
- A decisão não é capturada em um repositório de decisões arquitetônicas, então os membros da equipe esquecem ou não sabem que a decisão foi tomada.

Esses antipadrões são particularmente importantes para enfrentar durante o processo de desenvolvimento de um produto ou projeto.

Capturar a decisão, o contexto e as considerações que levaram à decisão na forma de um ADR permite que as partes interessadas atuais e future coletem informações sobre as decisões tomadas e o processo de pensamento por trás de cada decisão. Isso reduz o tempo de desenvolvimento de software e fornece melhor documentação para future equipes.

Resultados comerciais direcionados

ADRs visam três resultados comerciais:

- Eles alinham os membros atuais e future da equipe.
- Eles definem uma direção estratégica para o projeto ou produto.

- Eles evitam antipadrões de decisão definindo um processo para documentar e comunicar adequadamente as decisões arquitetônicas.

ADRs capturam o contexto da decisão de informar as future partes interessadas. Uma coleção de ADRs fornece uma experiência de entrega e documentação de referência. Os membros da equipe ou do projeto usam a coleção ADR para projetos de acompanhamento e planejamento de recursos do produto. Ser capaz de fazer referência a ADRs reduz o tempo necessário durante o desenvolvimento, revisões e decisões arquitetônicas. Os ADRs também permitem que outras equipes aprendam e obtenham insights sobre considerações feitas por outras equipes de desenvolvimento de projetos e produtos.

Processo ADR

Um registro de decisão arquitetônico (ADR) é um documento que descreve uma escolha que a equipe faz sobre um aspecto significativo da arquitetura de software que está planejando construir. Cada ADR descreve a decisão arquitetônica, seu contexto e suas consequências. Os ADRs têm estados e, portanto, seguem um ciclo de vida. Para ver um exemplo de ADR, consulte oApêndice (p. 12).

O processo ADR produz uma coleção de registros de decisão arquitetônica. Essa coleção cria o log de decisões. O log de decisão fornece o contexto do projeto, bem como informações detalhadas de implementação e design. Os membros do projeto deslizam as manchetes de cada ADR para obter uma visão geral do contexto do projeto. Eles lêem os ADRs para mergulhar profundamente nas implementações de projetos e nas escolhas de design.

Quando a equipe aceita um ADR, ele se torna imutável. Se novos insights exigirem uma decisão diferente, a equipe propõe um novo ADR. Quando a equipe aceita o novo ADR, ele substitui o ADR anterior.

Escopo do processo ADR

Os membros do projeto devem criar um ADR para cada decisão arquitetonicamente significativa que afeta o projeto ou produto de software, incluindo o seguinte (Richards e Ford 2020 (p. 11)):

- Estrutura (por exemplo, padrões como microsserviços)
- Requisitos não funcionais (segurança, alta disponibilidade e tolerância a falhas)
- Dependências (acoplamento de componentes)
- Interfaces (APIs e contratos publicados)
- Técnicas de construção (bibliotecas, estruturas, ferramentas e processos)

Requisitos funcionais e não funcionais são as entradas mais comuns para o processo ADR.

Conteúdo do ADR

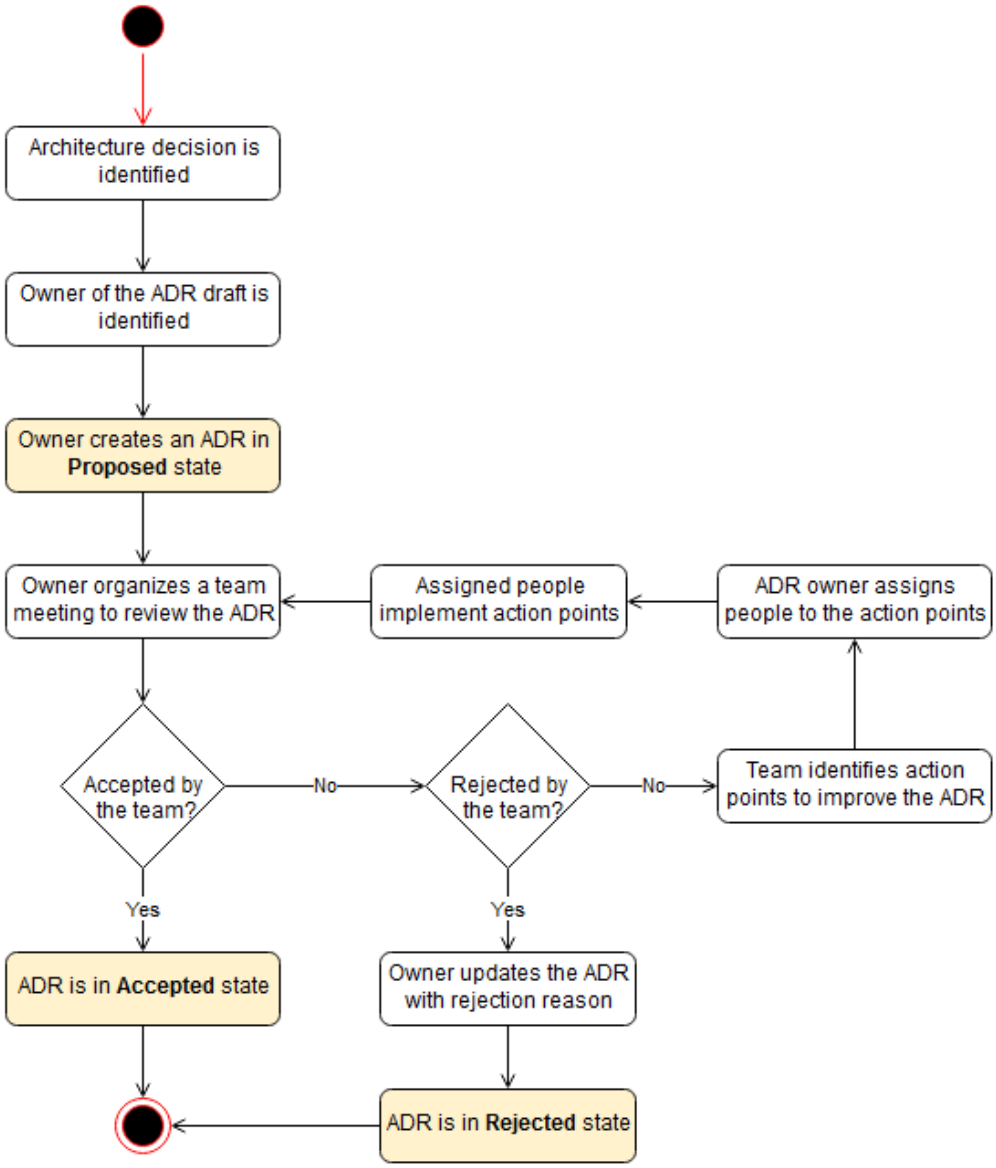
Quando a equipe identifica a necessidade de um ADR, um membro da equipe começa a escrever o ADR com base em um modelo de todo o projeto. (Consulte oOrganização ADR GitHubpor exemplo, modelos.) O modelo simplifica a criação do ADR e garante que o ADR capture todas as informações relevantes. No mínimo, cada ADR deve definir o contexto da decisão, a decisão em si e as consequências da decisão para o projeto e seus resultados. (Para obter exemplos dessas seções, consulte aApêndice (p. 12).) Um dos aspectos mais poderosos da estrutura do ADR é que ela se concentra no motivo da decisão, em vez de como a equipe a implementou. Compreender por que a equipe tomou a decisão torna mais fácil para outros membros da equipe adotarem a decisão e impede que outros arquitetos que não estavam envolvidos no processo de tomada de decisão anulem essa decisão no future.

Processo de adoção do ADR

Cada membro da equipe pode criar um ADR, mas a equipe deve estabelecer uma definição de propriedade para um ADR. Cada autor que é o proprietário de um ADR deve manter e comunicar ativamente o conteúdo do ADR. Para esclarecer essa propriedade, este guia se refere aos autores do

ADR comoProprietários do ADRnas seções a seguir. Outros membros da equipe sempre podem contribuir para um ADR. Se o conteúdo de um ADR mudar antes que a equipe aceite o ADR, o proprietário deverá aprovar essas alterações.

O diagrama a seguir ilustra o processo de criação, propriedade e adoção do ADR.



Depois que a equipe identifica uma decisão arquitetônica e seu proprietário, o proprietário do ADR fornece o ADR noPropostastate no início do processo. ADRs noPropostastate estão prontos para revisão.

O proprietário do ADR inicia o processo de revisão para o ADR. O objetivo do processo de revisão do ADR é decidir se a equipe aceita o ADR, determina que ele precisa de retrabalho ou rejeita o ADR. A equipe do projeto, incluindo o proprietário, analisa o ADR. A reunião de revisão deve começar com um horário dedicado para ler o ADR. Em média, 10 a 15 minutos devem ser suficientes. Durante esse período, cada membro da equipe lê o documento e adiciona comentários e perguntas para sinalizar tópicos pouco claros. Após a fase de revisão, o proprietário do ADR lê e discute cada comentário com a equipe.

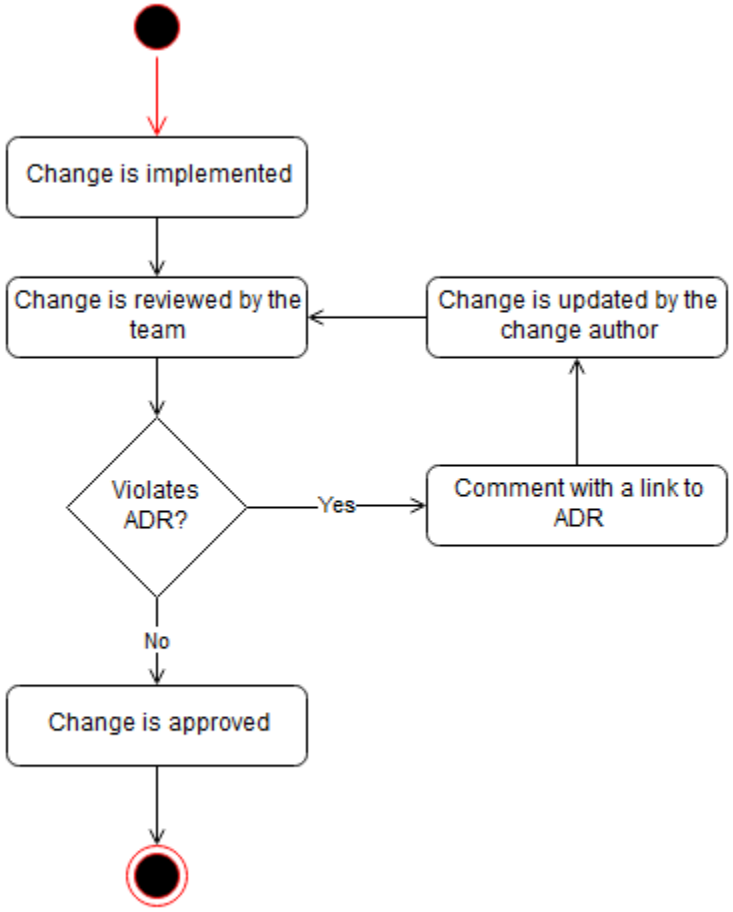
Se a equipe encontrar pontos de ação para melhorar o ADR, o estado do ADR permaneceProposta. O proprietário do ADR formula as ações e, em colaboração com a equipe, adiciona um destinatário a cada ação. Cada membro da equipe pode contribuir e resolver os pontos de ação. É responsabilidade do proprietário do ADR reagendar o processo de revisão.

A equipe também pode decidir rejeitar o ADR. Nesse caso, o proprietário do ADR adiciona um motivo para a rejeição para evitar future discussões sobre o mesmo tópico. O proprietário altera o estado do ADR paraRejeitado.

Se a equipe aprovar o ADR, o proprietário adicionará um carimbo de data/hora, versão e lista de partes interessadas. O proprietário atualiza o estado paraAceito.

ADRs e o registro de decisões que eles criam representam decisões tomadas pela equipe e fornecem um histórico de todas as decisões. A equipe usa os ADRs como referência durante revisões de código e arquitetura, sempre que possível. Além de realizar revisões de código, tarefas de projeto e tarefas de implementação, os membros da equipe devem consultar ADRs para decisões estratégicas para o produto.

O diagrama a seguir mostra o processo de aplicação de um ADR para validar se uma alteração em um componente de software estiver em conformidade com as decisões acordadas.

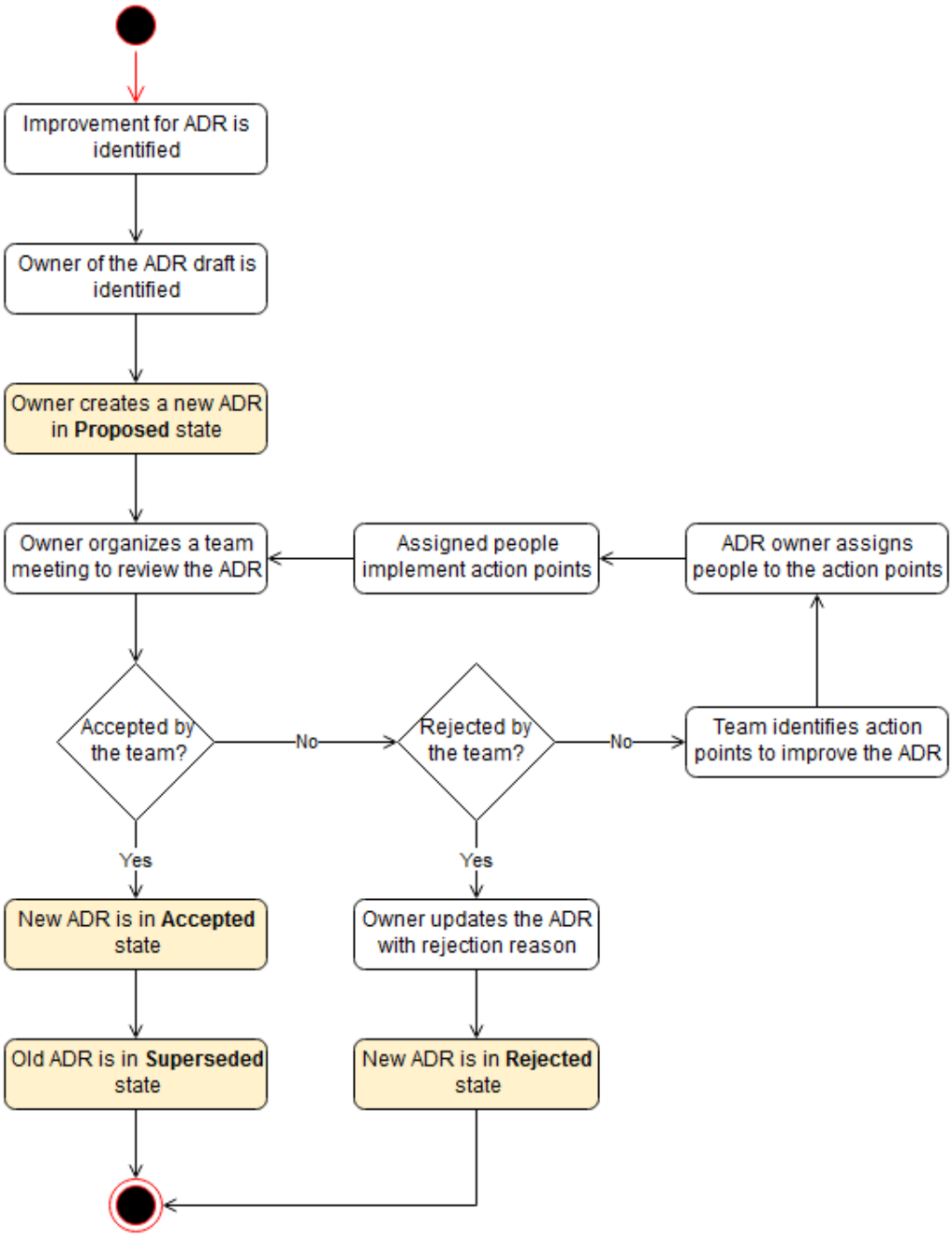


Como uma boa prática, cada alteração de software deve passar por revisões por pares e exigir pelo menos uma aprovação. Durante a revisão do código, um revisor de código pode encontrar alterações que violam um ou mais ADRs. Nesse caso, o revisor pede ao autor da alteração de código para atualizar o código e

compartilha um link para o ADR. Quando o autor atualiza o código, ele é aprovado por revisores pares e mesclado na base de código principal.

Processo de análise do ADR

A equipe deve tratar os ADRs como documentos imutáveis depois que a equipe os aceita ou rejeita. Alterações em um ADR existente exigem a criação de um novo ADR, estabelecer um processo de revisão para o novo ADR e aprovar o ADR. Se a equipe aprovar o novo ADR, o proprietário deverá alterar o estado do ADR antigo paraSubstituído. O diagrama a seguir ilustra o processo de atualização.


Improvement for ADR is identified
Owner of the ADR draft is identified
Owner creates a new ADR in **Proposed** state
Owner organizes a team meeting to review the ADR
Assigned people implement action points
ADR owner assigns people to the action points
Accepted by the team?
No
Rejected by the team?
No
Team identifies action points to improve the ADR
Yes
Yes
New ADR is in **Accepted** state
Owner updates the ADR with rejection reason
Old ADR is in **Superseded** state
New ADR is in **Rejected** state

Práticas recomendadas

Promote a propriedade. Cada membro da equipe do projeto deve ter o poder de criar e possuir um ADR. Essa prática distribui o trabalho de pesquisa arquitetônica entre os membros da equipe e descarrega que trabalham com o arquiteto de soluções ou líder da equipe. Também promove um senso de propriedade no processo de tomada de decisão. Isso ajuda a equipe a adotar essas decisões mais rapidamente em vez de tratá-las como decisões impostas a partir de níveis mais altos da organização.

Preserve o histórico de ADR. Os ADRs devem ter um histórico de alterações, e cada alteração deve ter um proprietário. Quando o proprietário do ADR atualiza o ADR, ele deve alterar o status do ADR antigo paraSubstituído, observe suas alterações no histórico de alterações do novo ADR e mantenha o ADR antigo no registro de decisões.

Agende reuniões regulares de revisão. Se você estiver em um novo projeto (greenfield), o processo de ADR pode ser bastante intenso no início. Recomendamos que você estabeleça uma cadência de reuniões regulares de discussão e revisão de ADR antes ou depois do stand up diário. Com essa abordagem, os ADRs definidos se estabilizarão em dois ou três sprints, e você poderá construir uma base sólida com menos reuniões.

Armazene ADRs em um local central. Cada membro do projeto deve ter acesso à coleção de ADRs. Recomendamos que você armazene os ADRs em um local central e faça referência a eles na página principal da documentação do projeto. Existem duas opções populares para armazenar ADRs:

- Um repositório Git, o que facilita a versão dos ADRs
- Uma página wiki, que torna as ADRs acessíveis a todos os membros da equipe

Endereço do código que não está em conformidade. O processo ADR não resolve o problema de código legado não compatível. Se você tiver um código legado que não suporta os ADRs estabelecidos, você pode atualizar a base de código desatualizada ou artefatos gradualmente, ao introduzir novas alterações, ou sua equipe pode decidir refatorar o código explicitamente criando tarefas de débito técnico.

Perguntas frequentes

Quais são os benefícios da criação de um processo de ADR?

A equipe do projeto deve criar um processo de ADR para agilizar a tomada de decisões arquitetônicas, evitar discussões repetidas sobre os mesmos tópicos arquitetônicos e comunicar decisões arquitetônicas de forma eficaz.

Quando a equipe do projeto deve criar um ADR?

A equipe do projeto deve criar um ADR para todos os aspectos do software que afetam a estrutura (padrões como microsserviços), requisitos não funcionais (segurança, alta disponibilidade e tolerância a falhas), dependências (acoplamento de componentes), interfaces (APIs e contratos publicados) e construção técnicas (bibliotecas, estruturas, ferramentas e processos).

Com que frequência a equipe do projeto deve analisar um ADR?

A equipe do projeto deve revisar o ADR pelo menos uma vez antes de aceitá-lo.

Quem deve criar um ADR?

Todo membro da equipe pode criar um ADR. Recomendamos que você promova uma noção de propriedade para ADRs. Um autor que possui o ADR deve manter e comunicar ativamente o conteúdo do ADR. Outros membros da equipe sempre podem contribuir para um ADR. O proprietário do ADR deve aprovar alterações em um ADR.

Quais informações um ADR deve conter?

No mínimo, cada ADR tem que definir o contexto da decisão, a decisão em si e as consequências da decisão para o projeto e seus resultados. O contexto deve mencionar possíveis soluções que a equipe considerou. Ele também deve conter qualquer informação relevante relacionada ao projeto, ao cliente ou à pilha de tecnologia. A decisão deve indicar claramente, em linguagem imperativa, a solução que a equipe decidiu adotar. Evite usar palavras como “deveria” e frase cada decisão para dizer “Usamos...” ou “A equipe tem que usar...” A seção de consequências deve mencionar todos os trade-offs conhecidos de tomar a decisão. Cada ADR deve ter um status e um changelog que contenha a data da alteração e a pessoa responsável pela alteração.

Onde posso encontrar modelos de ADR?

Existem várias versões e variantes de modelos ADR disponíveis. Para obter uma coleção pública de modelos ADR comumente usados, consulte aADR GitHub repositório.

Próximas etapas e recursos

Recomendamos que você comece pequeno e veja o benefício que os ADRs trazem para sua equipe. Se você estiver trabalhando em um projeto em andamento, identifique a próxima alteração arquitetônica e aplique o processo ADR proposto para criar seu primeiro ADR.

Outro ponto de partida é documentar seu processo geral de desenvolvimento de software usando ADRs. Muitas vezes, o processo de desenvolvimento é baseado no conhecimento tácito que a equipe não capturou em nenhuma documentação. A documentação desse processo permite uma experiência mais suave para novos membros da equipe.

Se você estiver em um projeto greenfield, aplique o processo ADR e comece a capturar todas as decisões desde o início em algumas frases. Em seguida, você pode iterar esses ADRs e complementá-los com novas informações. Depois de estabelecer seus ADRs, você pode começar a usá-los como referência no processo de revisão de código.

Recursos

- Registros de decisão de arquitetura.https://adr.github.io/.
- Richards, Mark e Neal Ford. 2020. Fundamentos da arquitetura de software. Sebastopol: O'Reilly Media.

Apêndice: Exemplo de ADR

Título

Essa decisão define a abordagem do ciclo de vida do desenvolvimento de software para o desenvolvimento de aplicativos ABC.

Status

Aceito

Data

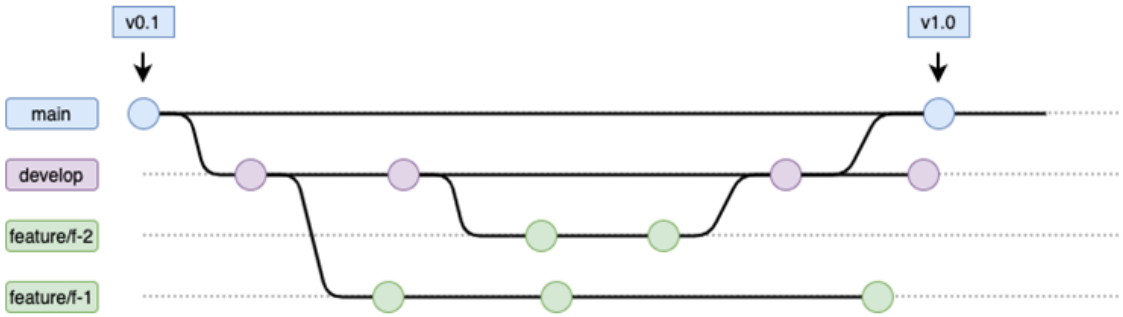
2022-03-11

Context

O aplicativo ABC é uma solução empacotada, que será implantada no ambiente do cliente usando um pacote de implantação. Precisamos ter um processo de desenvolvimento que nos permita ter um recurso controlável, hotfix e pipeline de lançamento.

Decisão

Usamos uma versão adaptada doFluxo de trabalho do GitFlowpara desenvolver o aplicativo ABC.



Por simplicidade, não usaremos o`hotfix/*`e`release/*`branches, porque o aplicativo ABC será empacotado em vez de ser implantado em um ambiente específico. Por esse motivo, não há necessidade de complexidade adicional que possa nos impedir de reagir rapidamente para corrigir bugs em versões de produção ou testar versões em um ambiente separado.

A seguir está a estratégia de ramificação acordada:

- Cada repositório deve ter um protegido`main`branch que será usado para marcar lançamentos.
- Cada repositório deve ter um protegido`develop`ramo para todo o trabalho de desenvolvimento em andamento.

Consequências

Positivo:

- Adaptado GitFlow processo nos permitirá controlar o controle de versão da versão do aplicativo ABC.

Negativo

- O GitFlow é mais complicado do que o desenvolvimento baseado em troncos ou GitHub flui e tem mais sobrecarga.

Conformidade

- O`main`e`develop`branches em cada repositório devem ser marcados como`Protected`.
- Alterações no`main`e`develop`as ramificações devem ser propagadas usando solicitações de mesclagem.
- Pelo menos uma aprovação é necessária para cada solicitação de mesclagem.

Observações

- Autor: Jane Doe
- Versão: 0.1
- Log de alterações:
  - 0.1: Versão inicial proposta

AWSGlossário de orientação prescritiva

Termos de IA e ML (p. 14)|Termos de migração (p. 15)|Termos de modernização (p. 20)

Termos de IA e ML

A seguir são termos comumente usados em estratégias, guias e padrões relacionados à inteligência artificial (IA) e aprendizado de máquina (ML) fornecidos porAWSOrientação prescritiva. Para sugerir entradas, use oComo fornecer feedbacklink no final do glossário.

Classificação binária	Um processo que prevê um resultado binário (uma das duas classes possíveis). Por exemplo, seu modelo de ML pode precisar prever problemas como “Este e-mail é spam ou não é spam?” ou “Este produto é um livro ou um carro?”
classificação	Um processo de categorização que ajuda a gerar previsões. Modelos de ML para problemas de classificação preveem um valor discreto. Valores discretos são sempre distintos um do outro. Por exemplo, um modelo pode precisar avaliar se há ou não um carro em uma imagem.
Pré-processamento de dados	Para transformar dados brutos em um formato que é facilmente analisado pelo seu modelo de ML. O pré-processamento de dados pode significar remover determinadas colunas ou linhas e endereçar valores ausentes, inconsistentes ou duplicados.
conjunto profundo	Combinar vários modelos de aprendizado profundo para previsão. Você pode usar conjuntos profundos para obter uma previsão mais precisa ou para estimar a incerteza nas previsões.
deep learning	Um subcampo ML que usa várias camadas de redes neurais artificiais para identificar o mapeamento entre dados de entrada e variáveis de destino de interesse.
Análise de dados exploratórios (EDA)	O processo de análise de 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 calculando estatísticas resumidas e criando visualizações de dados.
features	Os dados de entrada que você usa para fazer uma previsão. Por exemplo, em um contexto de fabricação, os recursos podem ser imagens capturadas periodicamente da linha de fabricação.
Importância do recurso	Quão significativo é um recurso 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 Explications (SHAP) e gradientes integrados. Para obter mais informações, consulteInterpretabilidade do modelo de machine learning com a AWS.

transformação de recurso	Para otimizar dados para o processo de ML, incluindo o enriquecimento de dados com fontes adicionais, valores de dimensionamento ou extração de vários conjuntos de informações de um único campo de dados. Isso permite que o modelo ML se beneficie dos dados. Por exemplo, se você dividir a data “2021-05-27 00:15:37” em “2021”, “Maio”, “Qui” e “15”, você pode ajudar o algoritmo de aprendizado a aprender padrões matizados associados a diferentes componentes de dados.
Interpretabilidade	Uma característica de um modelo de aprendizado de máquina que descreve o grau em que um humano pode entender como as previsões do modelo dependem de suas entradas. Para obter mais informações, consulteInterpretabilidade do modelo de machine learning com a AWS.
Classificação multiclasse	Um processo que ajuda a gerar previsões para várias classes (prevendo um entre mais de dois resultados). Por exemplo, um modelo de ML pode perguntar “Este produto é um livro, um carro ou telefone?” ou “Qual categoria de produto é mais interessante para este cliente?”
regressão	Uma técnica de ML que prevê um valor numérico. Por exemplo, para resolver o problema de “Qual será o preço de venda desta casa?” um modelo de ML poderia 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).
formação	Para fornecer dados para o modelo de ML para aprender. Os dados do treinamento devem conter a resposta correta. O algoritmo de aprendizagem localiza padrões nos dados de treinamento que mapeiam os atributos dos dados de entrada para o destino (a resposta que você deseja prever). Ele produz um modelo de ML que captura esses padrões. Em seguida, você pode usar o modelo de ML para fazer previsões dos novos dados cujo destino você não conhece.
Variável de destino	O valor que você está tentando prever em ML supervisionado. Isso também é chamado deVariável de resultado. Por exemplo, em uma configuração de fabricação, a variável de destino pode ser um defeito do produto.
ajustar	Para alterar aspectos do seu processo de treinamento para melhorar a precisão do modelo de ML. Por exemplo, você pode treinar o modelo de ML gerando um conjunto de rotulamento, adicionando rótulos e repetindo essas etapas várias vezes sob configurações diferentes para otimizar o modelo.
incerteza	Um conceito que se refere a informações imprecisas, incompletas ou desconhecidas que podem prejudicar a confiabilidade dos modelos preditivos de ML. Há dois tipos de incerteza: Incerteza epistêmicaé causada por dados limitados e incompletos, enquantoincerteza aleatóriaé causada pelo ruído e aleatoriedade inerentes aos dados. Para obter mais informações, consulte o .Quantificando a incerteza em sistemas de aprendizado profundoguia.

Termos de migração

A seguir são termos comumente usados em estratégias, guias e padrões relacionados à migração fornecidos porAWSOrientação prescritiva. Para sugerir entradas, use oComo fornecer feedbacklink no final do glossário.

7 Rs	Sete estratégias de migração comuns para mover aplicativos para a nuvem. Essas estratégias se baseiam nos 5 Rs que a Gartner identificou em 2011 e consistem no seguinte: • Refator/rearquiteto — Mova um aplicativo e modifique sua arquitetura aproveitando ao máximo os recursos nativos da nuvem para melhorar a agilidade, o desempenho e a escalabilidade. Isso geralmente envolve portar o

	sistema operacional e o banco de dados. Exemplo: Migre seu banco de dados Oracle no local para o Amazon Aurora, edição compatível com PostgreSQL. • Replataforma (elevar e remodelar) — Mova um aplicativo para a nuvem e introduza algum nível de otimização para aproveitar os recursos da nuvem. Exemplo: Migre seu banco de dados Oracle no local para o Amazon Relational Database Service (Amazon RDS) para Oracle noAWSCloud. • Recompra (drop and shop) — Mude para um produto diferente, geralmente passando de uma licença tradicional para um modelo SaaS. Exemplo: Migre seu sistema CRM (Customer Relationship Management) para Salesforce.com. • Rehost (lift and shift) — Mova um aplicativo para a nuvem sem fazer alterações para aproveitar os recursos da nuvem. Exemplo: Migrar seu banco de dados Oracle no local para o Oracle em uma instância do EC2 noAWSCloud. • Realocar (elevação e mudança no nível do hipervisor) — Mova a infraestrutura para a nuvem sem comprar novo hardware, reescrever aplicativos ou modificar suas operações existentes. Esse cenário de migração é específico para o VMware Cloud onAWS, que oferece suporte à compatibilidade de máquinas virtuais (VM) e portabilidade de carga de trabalho entre seu ambiente on-premises eAWS. Você pode usar as tecnologias VMware Cloud Foundation de seus data centers locais ao migrar sua infraestrutura para o VMware Cloud onAWS. Exemplo: Realocar o hipervisor que hospeda seu banco de dados Oracle para o VMware Cloud emAWS. • Retain (revisitar) — Mantenha os aplicativos em seu ambiente de origem. Isso pode incluir aplicativos que exigem refatoração importante, e você deseja adiar esse trabalho até um momento posterior, e aplicativos legados que você deseja manter, porque não há justificativa de negócios para migrá-los. • Desativação — Desative ou remova aplicativos que não são mais necessários em seu ambiente de origem.
portfólio de aplicativos	Uma coleção de informações detalhadas sobre cada aplicativo usado por uma organização, incluindo o custo para criar e manter o aplicativo e seu valor comercial. Essas informações são fundamentais parao processo de descoberta e análise do portfólioe ajuda a identificar e priorizar os aplicativos a serem migrados, modernizados e otimizados.
operações de inteligência artificial (AIOps)	O processo de usar técnicas de aprendizado de máquina para resolver problemas operacionais, reduzir incidentes operacionais e intervenção humana e aumentar a qualidade do serviço. Para obter mais informações sobre como o AIOps é usado noAWSestratégia de migração, veja oguia de integração de operações.
AWSEstrutura de adoção da nuvem (AWSCAF)	Uma estrutura de diretrizes e melhores práticas deAWSpara ajudar as organizações a desenvolver um plano eficiente e eficaz para migrar com sucesso para a nuvem.AWS A CAF organiza orientações em seis áreas de foco chamadas perspectivas: negócios, pessoas, governança, plataforma, segurança e operações. As perspectivas de negócios, pessoas e governança se concentram em habilidades e processos de negócios; as perspectivas de plataforma, segurança e operações se concentram 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 pessoas. Para essa perspectiva,AWSA CAF fornece orientação para desenvolvimento, treinamento e comunicação de pessoas para ajudar a preparar a organização para uma adoção bem-sucedida na nuvem. Para obter mais informações, consulte o .AWSSite da CAFO e aAWSWhitepaper da CAF.
AWSlanding zone	Uma landing zone é uma conta múltipla e bem arquitetadaAWSambiente escalável e seguro. Esse é um ponto de partida a partir do qual suas organizações podem iniciar e implantar rapidamente cargas de trabalho e aplicativos com confiança em seu ambiente de segurança e infraestrutura. Para obter mais informações

	sobre zonas de pouso, consulteConfigurando uma conta múltipla segura e escalávelAWSmeio Ambiente.
AWSWorkload Qualification Framework (AWSWQF)	Uma ferramenta que avalia cargas de trabalho de migração de banco de dados, recomenda estratégias de migração e fornece estimativas de trabalho.AWS O WQF está incluído com oAWS Schema Conversion Tool(AWS SCT). Ele analisa esquemas de banco de dados e objetos de código, código de aplicativo, dependências e características de desempenho, além de fornecer relatórios de avaliação.
planejamento de continuidade de negócios (BCP)	Um plano que aborda o impacto potencial de um evento disruptivo, como uma migração em larga escala, nas operações e permite que uma empresa retome as operações rapidamente.
Cloud Center of Excellence (CCoE)	Uma equipe multidisciplinar que impulsiona os esforços de adoção da nuvem em uma organização, incluindo o desenvolvimento de melhores práticas de nuvem, mobilização de recursos, estabelecimento de cronogramas de migração e liderando a organização por meio de transformações em larga escala. Para obter mais informações, consulte o .Publicações do CCoEnoAWSBlog de estratégia empresarial em nuvem.
estágios de adoção na nuvem	As quatro fases pelas quais as organizações normalmente passam quando migram para oAWSCloud: • Projeto — Executando alguns projetos relacionados à nuvem para fins de prova de conceito e aprendizado • Fundação — Fazer investimentos fundamentais para dimensionar a adoção da nuvem (por exemplo, criar uma landing zone, definir um CCoE, estabelecer um modelo de operações) • Migração — migração de aplicativos individuais • Reinvenção — Otimizando produtos e serviços e inovando na nuvem Essas etapas foram definidas por Stephen Orban na postagem do blogA jornada rumo à nuvem em primeiro lugar e os estágios de adoçãonoAWSBlog da Cloud Enterprise Strategy. Para obter informações sobre como eles se relacionam com oAWSestratégia de migração, veja oguia de prontidão para migração.
banco de dados de gerenciamento de configuração (CMDB)	Um banco de dados que contém informações sobre produtos de hardware e software, configurações e interdependências de uma empresa. Normalmente, você usa dados de um CMDB no estágio de descoberta e análise de portfólio da migração.
épico	Em metodologias ágeis, categorias funcionais que ajudam a organizar e priorizar seu trabalho. O Epics fornece uma descrição de alto nível dos requisitos e tarefas de implementação. Por exemplo,AWSOs epics de segurança 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 épicos noAWSestratégia de migração, veja oguia de implementação do programa.
Migração heterogênea de banco	Migrando 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 o Amazon Aurora). A migração heterogênea geralmente faz parte de um esforço de rearquitetura, e a conversão do esquema pode ser uma tarefa complexa.AWSforneceAWS SCTque ajuda com conversões de esquema.
Migração homogênea de banco	Migrando 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 o Amazon RDS for SQL Server). A migração homogênea geralmente faz parte de um esforço de rehospedagem ou replataforma. Você pode usar utilitários de banco de dados nativos para migrar o esquema.
Aplicação ociosa	Um aplicativo que tem um uso médio de CPU e memória entre 5% e 20% durante um período de 90 dias. Em um projeto de migração, é comum aposentar esses aplicativos ou mantê-los no local.
Biblioteca de informações de TI (ITIL)	Um conjunto de práticas recomendadas para fornecer serviços de TI e alinhar esses serviços com os requisitos de negócios. A ITIL fornece a base para o ITSM.
Gerenciamento de serviços de TI (ITSM)	Atividades associadas ao projeto, implementação, gerenciamento e suporte a serviços de TI para uma organização. Para obter informações sobre a integração de operações de nuvem com as ferramentas do ITSM, consulte oguia de integração de operações.
Migração grande	Uma migração de 300 ou mais servidores.
Migration Acceleration Program (MAP)	UmaAWSprograma que fornece suporte de consultoria, treinamento e serviços para ajudar as organizações a construir 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 de migração comuns.
Avaliação do portfólio de migração (MPA)	Uma ferramenta online que fornece informações para validar o business case para migração para oAWSCloud. O MPA fornece avaliação detalhada do portfólio (dimensionamento correto do servidor, preços, comparações de TCO, análise de custo de migração), bem como planejamento de migração (análise de dados de aplicativos e coleta de dados, agrupamento de aplicativos, priorização de migração e planejamento de ondas). OFerramenta MPA(requer login) está disponível gratuitamente para todosAWSconsultores e consultores parceiros da APN.
Avaliação de prontidão para migração (MRA)	O processo de obter insights sobre o status de prontidão para a nuvem de uma organização, identificar pontos fortes e fracos e construir um plano de ação para fechar lacunas identificadas, usando oAWSCAF. Para obter mais informações, consulte o .guia de prontidão para migração. O MRA é a primeira fase doEstratégia de migração da AWS.
Migração em escala	O processo de mover a maioria do portfólio de aplicativos para a nuvem em ondas, com mais aplicativos movidos a um ritmo mais rápido em cada onda. Esta fase usa as melhores práticas e lições aprendidas nas fases anteriores para implementar umFábrica de migraçãode equipes, ferramentas e processos para simplificar a migração de cargas de trabalho por meio de automação e entrega ágil. Esta é a terceira fase doAWSEstratégia de migração.
Fábrica de migração	Equipes multifuncionais que simplificam a migração de cargas de trabalho por meio de abordagens automatizadas e ágeis. As equipes de 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 aplicativos corporativos consiste em padrões repetidos que podem ser otimizados por uma abordagem de fábrica. Para obter mais informações, consulte o .discussão de fábricas de migraçãoO e aGuia do CloudEndure Migration Factoryneste conjunto de conteúdo.
Metadados de migração	As informações sobre o aplicativo e o servidor necessárias para concluir a migração. Cada padrão de migração requer um conjunto diferente de metadados de migração. Exemplos de metadados de migração incluem a sub-rede de destino, o security group eAWSconta.

Padrão de migração	Uma tarefa de migração repetível que detalha a estratégia de migração, o destino da migração e o aplicativo ou serviço de migração usado. Exemplo: Rehospede a migração para o Amazon EC2 comAWSApplication Migration Service
Estratégia de migração	A abordagem usada para migrar uma carga de trabalho para oAWSCloud. Para obter mais informações, consulte o .7 Rs (p. 15)entrada neste glossário e vejaMobilize sua organização para acelerar migrações em larga escala.
acordo de nível operacional (OLA)	Um contrato que esclarece o que os grupos de TI funcionais prometem entregar uns aos outros, para dar suporte a um SLA (contrato de nível de serviço).
integração de operações (OI)	O processo de modernização das operações na nuvem, que envolve planejamento de prontidão, automação e integração. Para obter mais informações, consulte o .guia de integração de operações.
gerenciamento de mudanças organizacionais (OCM)	Uma estrutura para gerenciar transformações empresariais importantes e disruptivas a partir de uma perspectiva de pessoas, cultura e liderança. O OCM ajuda as organizações a se prepararem e fazer a transição para novos sistemas e estratégias, acelerando a adoção de mudanças, abordando questões de transição e impulsionando mudanças culturais e organizacionais. NoAWSestratégia de migração, essa estrutura é chamadaaceleração de pessoas, devido à velocidade de mudança necessária em projetos de adoção de nuvem. Para obter mais informações, consulte o .Guia do OCM.
manual	Um conjunto de etapas predefinidas que capturam o trabalho associado às migrações, como fornecer funções principais de operações na nuvem. Um manual pode assumir a forma de scripts, runbooks automatizados ou um resumo dos processos ou etapas necessárias para operar seu ambiente modernizado.
Avaliação do portfólio	Um processo de descoberta, análise e priorização do portfólio de aplicativos para planejar a migração. Para obter mais informações, consulteAvaliar a prontidão para migração.
matriz responsável, responsável, consultada, informada (RACI)	Uma matriz que define e atribui funções e responsabilidades em um projeto. Por exemplo, você pode criar um RACI para definir a propriedade do controle de segurança ou para identificar funções e responsabilidades para tarefas específicas em um projeto de migração.
runbook	Um conjunto de procedimentos manuais ou automatizados necessários para executar uma tarefa específica. Normalmente, eles são criados para simplificar operações ou procedimentos repetitivos com altas taxas de erro.
contrato de nível de serviço (SLA)	Um contrato que esclarece o que uma equipe de TI promete entregar aos clientes, como tempo de atividade e desempenho do serviço.
Lista de tarefas	Uma ferramenta usada para rastrear 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, inclui o tempo estimado necessário, o proprietário e o progresso.
fluxo de trabalho	Grupos funcionais em um projeto de migração que são responsáveis por um conjunto específico de tarefas. Cada fluxo de trabalho é independente, mas suporta os outros fluxos de trabalho no projeto. Por exemplo, o fluxo de trabalho do portfólio é responsável por priorizar aplicativos, planejar ondas e coletar metadados de migração. O fluxo de trabalho do portfólio fornece esses ativos para o fluxo de trabalho de migração, que então migra os servidores e aplicativos.
aplicativo zombie	Um aplicativo que tem um uso médio de CPU e memória abaixo de 5%. Em um projeto de migração, é comum desativar esses aplicativos.

Termos de modernização

A seguir são termos comumente usados em estratégias, guias e padrões relacionados à modernização fornecidos porAWSOrientação prescritiva. Para sugerir entradas, use oComo fornecer feedbacklink no final do glossário.

capacidade comercial	O que uma empresa faz para gerar valor (por exemplo, vendas, atendimento ao cliente ou marketing). Arquiteturas de microsserviços e decisões de desenvolvimento podem ser orientadas por recursos de negócios. Para obter mais informações, consulte o .Organizado em torno de recursos empresariaisSeção do doExecutando microsserviços em contêineres emAWSwhitepaper.
Design com base no domínio	Uma abordagem para o desenvolvimento de um sistema de software complexo conectando seus componentes a domínios em evolução, ou objetivos principais de negócios, que cada componente atende. Este conceito foi introduzido por Eric Evans em seu livro,Design orientado por domínio: Enfrentando 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 de figo estrangulador, consulteModernizando os serviços web herdados Microsoft ASP.NET (ASMX) de forma incremental usando contêineres e Amazon API Gateway.
microsserviço	Um serviço pequeno e independente que se comunica por APIs bem definidas e normalmente pertence a equipes pequenas e autônomas. Por exemplo, um sistema de seguros pode incluir microsserviços mapeados para recursos de negócios, como vendas ou marketing, ou subdomínios, como compras, reivindicações ou análises. Os benefícios dos microsserviços incluem agilidade, dimensionamento flexível, fácil implantação, código reutilizável e resiliência. Para obter mais informações, consulteIntegração de microsserviços usandoAWSserviços sem servidor.
Arquitetura de microsserviços	Uma abordagem para criar um aplicativo com componentes independentes que executam cada processo de aplicativo como um microsserviço. Esses microsserviços se comunicam por meio de uma interface bem definida usando APIs leves. Cada microsserviço nessa arquitetura pode ser atualizado, implantado e dimensionado para atender à demanda por funções específicas de um aplicativo. Para obter mais informações, consulteImplementar microsserviços noAWS.
modernização	Transformar um aplicativo desatualizado (legado ou monolítico) 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, consulteEstratégia para modernizar aplicativos noAWSNuvem.
Avaliação da prontidão para modernização	Uma avaliação que ajuda a determinar a prontidão para modernização dos aplicativos de uma organização; identifica benefícios, riscos e dependências; e determina o quão bem a organização pode suportar o estado future desses aplicativos. O resultado da avaliação é um projeto da arquitetura-alvo, um roteiro que detalha as fases de desenvolvimento e marcos para o processo de modernização e um plano de ação para abordar as lacunas identificadas. Para obter mais informações, consulteAvaliando a prontidão para modernização para aplicativos noAWSNuvem.
aplicações monolíticas (monólitos)	Aplicativos que são executados como um único serviço com processos bem acoplados. Aplicações monolíticas têm várias desvantagens. Se um recurso de aplicativo sofrer um pico na demanda, toda a arquitetura deve ser dimensionada. Adicionar ou melhorar os recursos de um aplicativo monolítico também se torna mais complexo quando a base de código cresce. Para resolver esses problemas, você pode usar uma arquitetura de microsserviços. Para obter mais informações, consulteDecompondo monólitos em microsserviços.

Persistência poliglota	Escolhendo independentemente 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 encontrar desafios de implementação ou experimentar um desempenho ruim. Os microsserviços são implementados com mais facilidade e obtêm melhor desempenho e escalabilidade se usarem o armazenamento de dados melhor adaptado às suas necessidades. Para obter mais informações, consulteHabilitando a persistência de dados em microsserviços.
modelo de divisão e semente	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 dá suporte à inovação rápida. Para obter mais informações, consulteAbordagem faseada para modernizar aplicativos noAWSNuvem.
Padrão de figo estrangulador	Uma abordagem para modernizar sistemas monolíticos reescrevendo e substituindo incrementalmente a funcionalidade do sistema até que o sistema legado possa ser desativado. Esse padrão usa a analogia de uma videira de figo que cresce em uma árvore estabelecida e eventualmente supera e substitui seu hospedeiro. O padrão eraintroduzido por Martin Fowlercomo forma de gerenciar riscos ao reescrever sistemas monolíticos. Para obter um exemplo de como aplicar esse padrão, consulteModernizando os serviços web herdados Microsoft ASP.NET (ASMX) de forma incremental usando contêineres e Amazon API Gateway.
equipe de duas pizzas	Um pequeno DevOps equipe que você pode alimentar com duas pizzas. Um tamanho de equipe de duas pizzas garante a melhor oportunidade possível de colaboração no desenvolvimento de software. Para obter mais informações, consulte o .Equipe de duas pizzaSeção do doIntrodução ao DevOps emAWSwhitepaper.

Histórico do documento

A tabela a seguir descreve alterações significativas feitas neste guia. Se quiser ser notificado sobre future atualizações, assine umFeed RSS.

update-history-change	update-history-description	update-history-date
Publicação inicial (p. 22)	—	16 de março de 2022

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.