Scelta di una strategia di ramificazione Git per ambienti con più account DevOps

AWS Guida prescrittiva

AWS Guida prescrittiva: Scelta di una strategia di ramificazione Git per ambienti con più account DevOps

Table of Contents

Scelta di una strategia di ramificazione Git per ambienti con più account DevOps

Amazon Web Services (collaboratori)

Febbraio 2024 (cronologia dei documenti)

Passare a un approccio basato sul cloud e fornire soluzioni software AWS può essere trasformativo. Potrebbe richiedere modifiche al processo del ciclo di vita dello sviluppo del software. In genere, Account AWS durante il processo di sviluppo in. Cloud AWS La scelta di una strategia di ramificazione Git compatibile da abbinare ai DevOps processi è essenziale per il successo. La scelta della strategia di ramificazione Git giusta per la tua organizzazione ti aiuta a comunicare in modo conciso DevOps gli standard e le migliori pratiche tra i team di sviluppo. La ramificazione di Git può essere semplice in un singolo ambiente, ma può creare confusione se applicata in più ambienti, come sandbox, sviluppo, test, staging e ambienti di produzione. La presenza di più ambienti aumenta la complessità dell'implementazione. DevOps

Questa guida fornisce diagrammi visivi delle strategie di ramificazione di Git che mostrano come un'organizzazione può implementare un processo multi-account. DevOps Le guide visive aiutano i team a capire come unire le loro strategie di ramificazione Git con le loro DevOps pratiche. L'utilizzo di un modello di ramificazione standard, come Gitflow, Flow o Trunk, GitHub per la gestione del repository del codice sorgente aiuta i team di sviluppo ad allineare il proprio lavoro. Questi team possono anche utilizzare risorse di formazione Git standard su Internet per comprendere e implementare tali modelli e strategie.

Per le DevOps migliori pratiche in materia AWS, DevOpsconsulta la Guida in AWS Well-Architected. Durante la lettura di questa guida, utilizzate la due diligence per selezionare la strategia di ramificazione giusta per la vostra organizzazione. Alcune strategie potrebbero adattarsi meglio di altre al tuo caso d'uso.

Obiettivi

Questa guida fa parte di una serie di documentazione sulla scelta e l'implementazione di strategie di DevOps ramificazione per organizzazioni con più Account AWS membri. Questa serie è progettata per aiutarti ad applicare la strategia che meglio soddisfa i tuoi requisiti, obiettivi e best practice sin dall'inizio, per semplificare la tua esperienza nel. Cloud AWS Questa guida non contiene script

DevOps eseguibili perché variano in base al motore di integrazione e distribuzione continua (CI/CD) e ai framework tecnologici utilizzati dall'organizzazione.

Questa guida spiega le differenze tra tre strategie di ramificazione Git comuni: GitHub Flow, Gitflow e Trunk. I consigli contenuti in questa guida aiutano i team a identificare una strategia di ramificazione in linea con i loro obiettivi organizzativi. Dopo aver esaminato questa guida, dovresti essere in grado di scegliere una strategia di ramificazione per la tua organizzazione. Dopo aver scelto una strategia, puoi utilizzare uno dei seguenti schemi per aiutarti a implementarla con i tuoi team di sviluppo:

- Implementa una strategia di ramificazione Trunk per ambienti con più account DevOps
- Implementa una strategia di ramificazione GitHub Flow per ambienti con più account DevOps
- Implementa una strategia di ramificazione Gitflow per ambienti con più account DevOps

È importante notare che ciò che funziona per un'organizzazione, un team o un progetto potrebbe non essere adatto per altri. La scelta tra le strategie di ramificazione di Git dipende da vari fattori, come la dimensione del team, i requisiti del progetto e l'equilibrio desiderato tra collaborazione, frequenza di integrazione e gestione delle release.

Utilizzo delle pratiche CI/CD

AWS consiglia di implementare l'integrazione e la distribuzione continue (le 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 pipeline forniscono anche governance e barriere per i team di sviluppo). Applicano coerenza, standard, best practice e livelli minimi di accettazione per l'accettazione e l'implementazione delle funzionalità. Per ulteriori informazioni, consulta Practicing Continuous Integration and Continuous Delivery su. AWS

Tutte le strategie di ramificazione discusse in questa guida sono adatte a 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 stabilire barriere e meccanismi per i team di sviluppo che impediscano agli sviluppatori di aggirare intenzionalmente o meno il processo definito.

AWS offre una suite di servizi per sviluppatori progettati per aiutarvi a creare pipeline CI/CD. Ad esempio, AWS CodePipelineè un servizio di distribuzione continua completamente gestito che consente di automatizzare le pipeline di rilascio per aggiornamenti rapidi e affidabili di applicazioni e infrastrutture. AWS CodeBuildcompila il codice sorgente, esegue test e produce ready-to-deploy pacchetti software. Per ulteriori informazioni, consulta Developer Tools on AWS.

Comprendere gli DevOps ambienti

Per comprendere le strategie di ramificazione, è necessario comprendere lo scopo e le attività che si svolgono in ogni ambiente. La creazione di diversi ambienti consente di suddividere le attività di sviluppo in fasi, monitorare tali attività e prevenire il rilascio involontario di funzionalità non approvate. È possibile averne uno o più Account AWS in ogni ambiente.

La maggior parte delle organizzazioni dispone di diversi ambienti predisposti per l'uso. Tuttavia, il numero di ambienti può variare a seconda dell'organizzazione e in base alle politiche di sviluppo del software. Questa serie di documentazione presuppone che siano presenti i seguenti cinque ambienti comuni che coprono la pipeline di sviluppo, sebbene possano essere chiamati con nomi diversi:

- Sandbox: un ambiente in cui gli sviluppatori scrivono codice, commettono errori ed eseguono prove di fattibilità.
- Sviluppo: un ambiente in cui gli sviluppatori integrano il codice per confermare che tutto funzioni come un'unica applicazione coesa.
- Test: un ambiente in cui si svolgono i team di controllo qualità o i test di accettazione. I team spesso eseguono test delle prestazioni o dell'integrazione in questo ambiente.
- Staging: un ambiente di preproduzione in cui si verifica che il codice e l'infrastruttura funzionino come previsto in circostanze equivalenti alla produzione. Questo ambiente è configurato per essere il più simile possibile all'ambiente di produzione.
- Produzione: un ambiente che gestisce il traffico proveniente dagli utenti finali e dai clienti.

Questa sezione descrive ogni ambiente in dettaglio. Descrive inoltre le fasi di creazione, le fasi di implementazione e i criteri di uscita per ogni ambiente in modo da poter passare a quello successivo. L'immagine seguente mostra questi ambienti in sequenza.

Argomenti in questa sezione:

- Ambiente sandbox
- Ambiente di sviluppo

- Ambiente di test
- Ambiente di messa in scena
- Ambiente di produzione

Ambiente sandbox

L'ambiente sandbox è il luogo in cui gli sviluppatori scrivono codice, commettono errori ed eseguono prove di fattibilità. È possibile eseguire la distribuzione in un ambiente sandbox da una workstation locale o tramite uno script su una workstation locale.

Accesso

Gli sviluppatori devono avere pieno accesso all'ambiente sandbox.

Costruisci passaggi

Gli sviluppatori eseguono manualmente la build sulle workstation locali quando sono pronti a implementare le modifiche all'ambiente sandbox.

1. Usa git-secrets () GitHub per cercare informazioni sensibili
2. Lint il codice sorgente
3. Compila e compila il codice sorgente, se applicabile
4. Esegui test unitari
5. Esegui l'analisi della copertura del codice
6. Esecuzione dell'analisi statica del codice
7. Crea un'infrastruttura come codice (IaC)
8. Esegui l'analisi della sicurezza IaC
9. Estrai licenze open source
10. Pubblica gli artefatti della build

Fasi della distribuzione

Se utilizzi i modelli Gitflow o Trunk, le fasi di implementazione vengono avviate automaticamente quando una `feature` filiale viene creata correttamente nell'ambiente sandbox. Se utilizzi il modello

GitHub Flow, esegui manualmente i seguenti passaggi di distribuzione. Di seguito sono riportati i passaggi di distribuzione nell'ambiente sandbox:

1. Scarica gli artefatti pubblicati
2. Esegui il controllo delle versioni del database
3. Eseguire l'implementazione di IaC
4. Esegui test di integrazione

Aspettative prima di passare all'ambiente di sviluppo

- Creazione riuscita della `feature` filiale nell'ambiente sandbox
- Uno sviluppatore ha implementato e testato manualmente la funzionalità nell'ambiente sandbox

Ambiente di sviluppo

L'ambiente di sviluppo è il luogo in cui gli sviluppatori integrano il codice per garantire che tutto funzioni come un'unica applicazione coesa. In Gitflow, l'ambiente di sviluppo contiene le funzionalità più recenti incluse nella richiesta di unione e sono pronte per il rilascio. Nelle strategie GitHub Flow and Trunk, l'ambiente di sviluppo è considerato un ambiente di test e la base di codice potrebbe essere instabile e inadatta alla distribuzione in produzione.

Accesso

Assegna le autorizzazioni in base al principio del privilegio minimo. Il privilegio minimo è la best practice di sicurezza per la concessione delle autorizzazioni minime richieste per eseguire un'attività. Gli sviluppatori dovrebbero avere meno accesso all'ambiente di sviluppo rispetto all'ambiente sandbox.

Costruisci passaggi

La creazione di una richiesta di unione al `develop` ramo (Gitflow) o al `main` ramo (Trunk o GitHub Flow) avvia automaticamente la build.

1. Usa git-secrets () GitHub per cercare informazioni sensibili
2. Lint il codice sorgente

3. Compila e compila il codice sorgente, se applicabile
4. Esegui test unitari
5. Esegui l'analisi della copertura del codice
6. Esecuzione dell'analisi statica del codice
7. Costruisci iAc
8. Esegui l'analisi di sicurezza IaC
9. Estrai licenze open source

Fasi della distribuzione

Se utilizzi il modello Gitflow, le fasi di implementazione vengono avviate automaticamente quando una `develop` filiale viene creata con successo nell'ambiente di sviluppo. Se utilizzi il modello GitHub Flow o il modello Trunk, le fasi di distribuzione vengono avviate automaticamente quando viene creata una richiesta di unione sul ramo. `main` Di seguito sono riportate le fasi di implementazione nell'ambiente di sviluppo:

1. Scarica gli artefatti pubblicati dalle fasi di compilazione
2. Esegui il controllo delle versioni del database
3. Eseguire l'implementazione di IaC
4. Esegui test di integrazione

Aspettative prima di passare all'ambiente di test

- Creazione e implementazione riuscite del `develop` ramo (Gitflow) o del `main` ramo (Trunk o GitHub Flow) nell'ambiente di sviluppo
- I test unitari vengono superati al 100%
- Build IaC di successo
- Gli artefatti di distribuzione sono stati creati con successo
- Uno sviluppatore ha eseguito una verifica manuale per confermare che la funzionalità funzioni come previsto

Ambiente di test

Il personale addetto al controllo della qualità (QA) utilizza l'ambiente di test per convalidare le funzionalità. Approvano le modifiche dopo aver terminato i test. Una volta approvate, la filiale passa all'ambiente successivo, lo staging. In Gitflow, questo ambiente e altri superiori sono disponibili per la distribuzione solo dalle filiali. `release` Un `release` ramo si basa su un `develop` ramo che contiene le funzionalità pianificate.

Accesso

Assegna le autorizzazioni in base al principio del privilegio minimo. Gli sviluppatori dovrebbero avere meno accesso all'ambiente di test rispetto a quello di sviluppo. Il personale addetto al controllo qualità richiede autorizzazioni sufficienti per testare la funzionalità.

Costruisci passaggi

Il processo di compilazione in questo ambiente è applicabile solo per le correzioni di bug quando si utilizza la strategia Gitflow. La creazione di una richiesta di unione al ramo avvia automaticamente la build`bugfix`.

1. Usa git-secrets (GitHub) per cercare informazioni sensibili
2. Lint il codice sorgente
3. Compila e compila il codice sorgente, se applicabile
4. Esegui test unitari
5. Esegui l'analisi della copertura del codice
6. Esecuzione dell'analisi statica del codice
7. Costruisci iAc
8. Esegui l'analisi di sicurezza IaC
9. Estrai licenze open source

Fasi della distribuzione

Avvia automaticamente la distribuzione della `release` filiale (Gitflow) o della `main` filiale (Trunk o GitHub Flow) nell'ambiente di test dopo l'implementazione nell'ambiente di sviluppo. Di seguito sono riportate le fasi di implementazione nell'ambiente di test:

1. Implementa il `release` ramo (Gitflow) o il `main` ramo (Trunk o GitHub Flow) nell'ambiente di test
2. Pausa per l'approvazione manuale da parte del personale designato
3. Scarica gli artefatti pubblicati
4. Esegui il controllo delle versioni del database
5. Eseguire l'implementazione di IaC
6. Esegui test di integrazione
7. Esegui test delle prestazioni
8. Approvazione del controllo di qualità

Aspettative prima di passare all'ambiente di allestimento

- I team di sviluppo e controllo qualità hanno eseguito test sufficienti per soddisfare i requisiti dell'organizzazione.
- Il team di sviluppo ha risolto tutti i bug rilevati tramite una `bugfix` filiale.

Ambiente di messa in scena

L'ambiente di staging è configurato per essere lo stesso dell'ambiente di produzione. Ad esempio, la configurazione dei dati dovrebbe essere simile per ambito e dimensioni ai carichi di lavoro di produzione. Utilizza l'ambiente di staging per verificare che il codice e l'infrastruttura funzionino come previsto. Questo ambiente è anche la scelta preferita per casi d'uso aziendali, come anteprime o dimostrazioni per i clienti.

Accesso

Assegna le autorizzazioni in base al principio del privilegio minimo. Gli sviluppatori devono avere lo stesso accesso all'ambiente di staging che hanno all'ambiente di produzione.

Costruisci i passaggi

Nessuna. Gli stessi artefatti utilizzati nell'ambiente di test vengono riutilizzati nell'ambiente di staging.

Fasi della distribuzione

Avvia automaticamente la distribuzione della `release` filiale (Gitflow) o della `main` filiale (Trunk o GitHub Flow) nell'ambiente di staging dopo l'approvazione e la distribuzione nell'ambiente di test. Di seguito sono riportate le fasi di implementazione nell'ambiente di staging:

1. Implementa il `release` ramo (Gitflow) o il `main` ramo (Trunk o GitHub Flow) nell'ambiente di staging
2. Pausa per l'approvazione manuale da parte del personale designato
3. Scarica gli artefatti pubblicati
4. Esegui il controllo delle versioni del database
5. Eseguire l'implementazione di IaC
6. (Facoltativo) Eseguire test di integrazione
7. (Facoltativo) Eseguire test di carico
8. Ottenete l'approvazione dei responsabili dello sviluppo, del controllo qualità, del prodotto o dell'azienda necessari

Aspettative prima di passare all'ambiente di produzione

- Una release equivalente alla produzione è stata implementata con successo nell'ambiente di staging
- (Facoltativo) I test di integrazione e carico hanno avuto esito positivo

Ambiente di produzione

L'ambiente di produzione supporta il prodotto rilasciato e gestisce dati reali di clienti reali. Si tratta di un ambiente protetto a cui viene assegnato l'accesso in base al privilegio minimo e l'accesso elevato deve essere consentito solo attraverso un processo di eccezione verificato per un periodo di tempo limitato.

Accesso

Nell'ambiente di produzione, gli sviluppatori dovrebbero avere un accesso limitato e in sola lettura alla Console di gestione AWS. Ad esempio, gli sviluppatori dovrebbero essere in grado di accedere ai dati

di registro per le day-to-day operazioni. Tutte le versioni in produzione devono essere controllate da una fase di approvazione prima della distribuzione.

Fasi di costruzione

Nessuna. Gli stessi artefatti utilizzati negli ambienti di test e staging vengono riutilizzati nell'ambiente di produzione.

Fasi della distribuzione

Avvia automaticamente la distribuzione della `release` filiale (Gitflow) o della `main` filiale (Trunk o GitHub Flow) nell'ambiente di produzione dopo l'approvazione e la distribuzione nell'ambiente di staging. Di seguito sono riportate le fasi di implementazione nell'ambiente di produzione:

1. Implementa il `release` ramo (Gitflow) o il `main` ramo (Trunk o GitHub Flow) nell'ambiente di produzione
2. Pausa per l'approvazione manuale da parte del personale designato
3. Scarica gli artefatti pubblicati
4. Esegui il controllo delle versioni del database
5. Eseguire l'implementazione di IaC

Le migliori pratiche per lo sviluppo basato su Git

Per adottare con successo lo sviluppo basato su Git, è importante seguire una serie di best practice che promuovono la collaborazione, mantengono la qualità del codice e supportano l'integrazione e la distribuzione continue (CI/CD). Oltre alle best practice riportate in questa guida, consulta la AWS DevOps Well-Architected Guidance. Di seguito sono riportate alcune best practice chiave per lo sviluppo basato su Git su: AWS

- Mantieni le modifiche piccole e frequenti: incoraggia gli sviluppatori a apportare modifiche o funzionalità piccole e incrementali. Ciò riduce il rischio di conflitti di fusione e semplifica l'identificazione e la risoluzione rapida dei problemi.
- Usa gli interruttori di funzionalità: per gestire il rilascio di funzionalità incomplete o sperimentali, utilizza gli interruttori di funzionalità o i flag di funzionalità. Ciò consente di nascondere, abilitare o disabilitare funzionalità specifiche in produzione senza influire sulla stabilità della filiale principale.
- Mantieni una suite di test solida: una suite di test completa e ben gestita è fondamentale per rilevare tempestivamente i problemi e verificare che la base di codice rimanga stabile. Investi nell'automazione dei test e dai la priorità alla correzione di eventuali test non riusciti.
- Adotta l'integrazione continua: utilizza strumenti e pratiche di integrazione continua per creare, testare e integrare automaticamente le modifiche al codice nella `develop` filiale (Gitflow) o nella filiale (Trunk o `main` Flow). GitHub In questo modo è possibile individuare tempestivamente i problemi e semplificare il processo di sviluppo.
- Esegui revisioni del codice: incoraggia le revisioni tra pari del codice per mantenere la qualità, condividere le conoscenze e individuare potenziali problemi prima che vengano integrati nella `main` filiale. Utilizza le pull request o altri strumenti di revisione del codice per facilitare questo processo.
- Monitora e correggi le build difettose: quando una build si interrompe o i test falliscono, dai la priorità alla risoluzione del problema il prima possibile. Ciò mantiene il `develop` ramo (Gitflow) o il `main` ramo (Trunk o GitHub Flow) in uno stato di rilascio e riduce al minimo l'impatto sugli altri sviluppatori.
- Comunica e collabora: promuovi la comunicazione e la collaborazione aperte tra i membri del team. Assicurati che gli sviluppatori siano a conoscenza del lavoro in corso e delle modifiche apportate al codice base.

- Effettua il refactoring continuo: rifattorizza regolarmente la base di codice per migliorarne la manutenibilità e ridurre il debito tecnico. Incoraggia gli sviluppatori a lasciare il codice in uno stato migliore di quello in cui lo hanno trovato.
- Usa rami di breve durata per attività complesse: per attività più grandi o più complesse, utilizza rami di breve durata (noti anche come rami di attività) per lavorare sulle modifiche. Tuttavia, assicurati di mantenere la durata della filiale breve, in genere meno di un giorno. Unisci nuovamente le modifiche nel `develop` ramo (Gitflow) o nel `main` ramo (Trunk o GitHub Flow) il prima possibile. Le fusioni e le revisioni più piccole e frequenti sono più facili da utilizzare ed elaborare per un team rispetto a una richiesta di unione di grandi dimensioni.
- Forma e supporta il team: fornisci formazione e supporto agli sviluppatori che sono nuovi allo sviluppo basato su Git o che necessitano di una guida per adottarne le migliori pratiche.

Strategie di ramificazione Git

In ordine dal meno complesso al più complesso, questa guida descrive in dettaglio le seguenti strategie di ramificazione basate su Git:

- Trunk — Lo sviluppo basato su Trunk è una pratica di sviluppo software in cui tutti gli sviluppatori lavorano su un unico ramo, in genere chiamato ramo o. `trunk` `main` L'idea alla base di questo approccio è quella di mantenere la base di codice in uno stato di rilascio continuo integrando frequentemente le modifiche al codice e facendo affidamento su test automatizzati e integrazione continua.
- GitHub Flow: GitHub Flow è un flusso di lavoro leggero e basato su filiali sviluppato da. GitHub Si basa sull'idea di filiali di breve durata`feature`. Quando una funzionalità è completa e pronta per essere implementata, viene unita al ramo. `main`
- Gitflow: con un approccio Gitflow, lo sviluppo viene completato in singoli rami di funzionalità. Dopo l'approvazione, unisci le `feature` filiali in un ramo di integrazione che di solito viene denominato. `develop` Quando nel `develop` ramo si sono accumulate un numero sufficiente di funzionalità, viene creato un `release` ramo per distribuirle negli ambienti superiori.

Ogni strategia di ramificazione presenta vantaggi e svantaggi. Sebbene utilizzino tutti gli stessi ambienti, non tutte utilizzano le stesse filiali o le stesse fasi di approvazione manuale. In questa sezione della guida, esaminate in dettaglio ogni strategia di ramificazione in modo da conoscerne le sfumature e valutare se si adatta al caso d'uso della vostra organizzazione.

Argomenti in questa sezione:

- Strategia di ramificazione del tronco
- GitHub Strategia di ramificazione del flusso
- Strategia di ramificazione Gitflow

Strategia di ramificazione del tronco

Lo sviluppo basato su trunk è una pratica di sviluppo software in cui tutti gli sviluppatori lavorano su un unico ramo, in genere chiamato ramo o. `trunk` `main` L'idea alla base di questo approccio è quella di mantenere la base di codice in uno stato di rilascio continuo integrando frequentemente le modifiche al codice e facendo affidamento su test automatizzati e integrazione continua.

Nello sviluppo basato su trunk, gli sviluppatori apportano le modifiche alla `main` filiale più volte al giorno, con l'obiettivo di effettuare aggiornamenti piccoli e incrementali. Ciò consente cicli di feedback rapidi, riduce il rischio di conflitti di fusione e favorisce la collaborazione tra i membri del team. La pratica sottolinea l'importanza di una suite di test ben gestita perché si basa su test automatizzati per individuare tempestivamente potenziali problemi e assicurarsi che la base di codice rimanga stabile e rilasciabile.

Lo sviluppo basato su trunk viene spesso contrapposto allo sviluppo basato sulle funzionalità (noto anche come ramificazione delle funzionalità o sviluppo basato sulle funzionalità), in cui ogni nuova funzionalità o correzione di bug viene sviluppata in un ramo dedicato, separato dal ramo principale. La scelta tra sviluppo basato su trunk e sviluppo basato su funzionalità dipende da fattori quali la dimensione del team, i requisiti del progetto e l'equilibrio desiderato tra collaborazione, frequenza di integrazione e gestione delle release.

Per ulteriori informazioni sulla strategia di ramificazione Trunk, consulta le seguenti risorse:

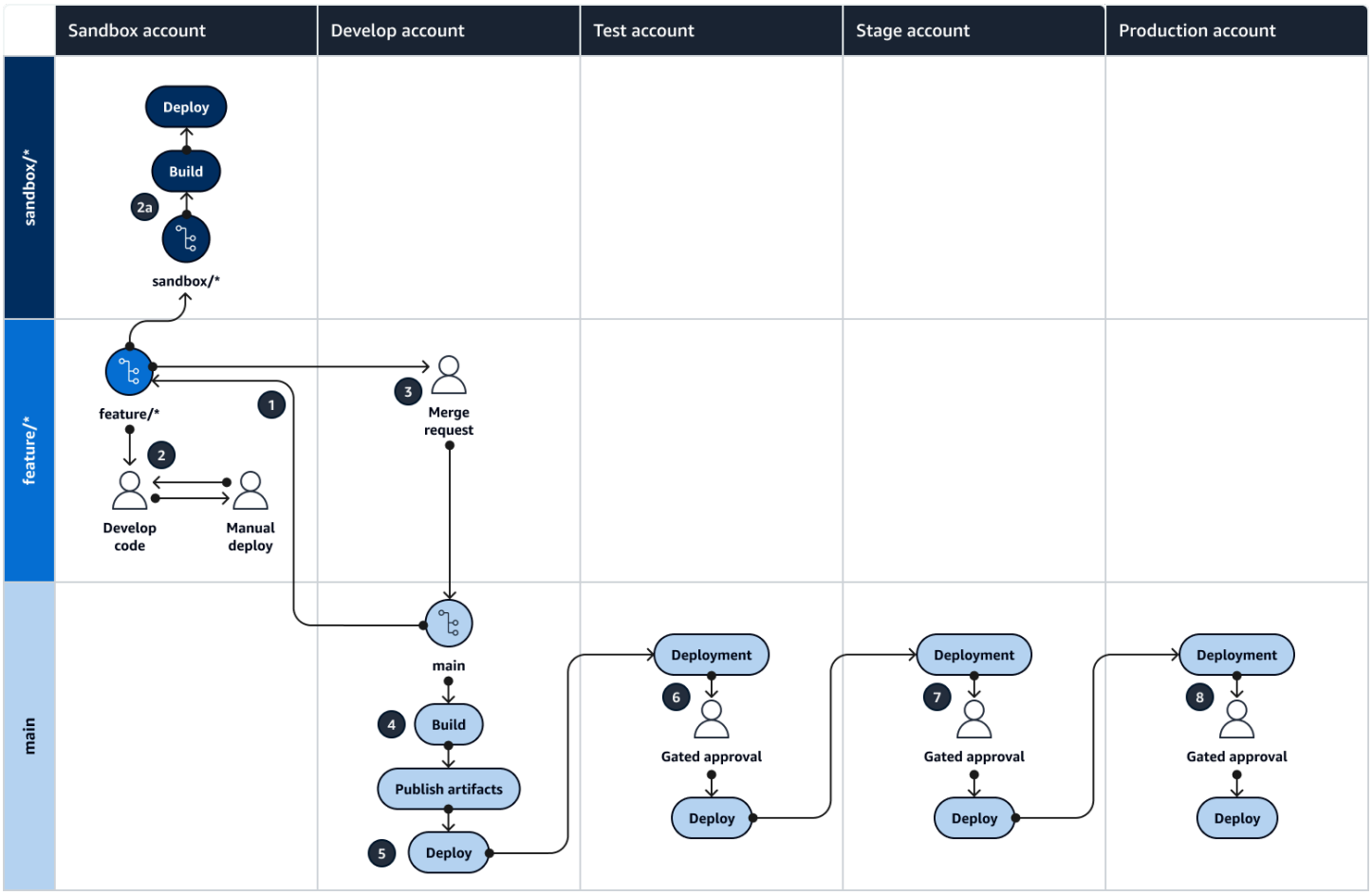
- Implementa una strategia di ramificazione Trunk per DevOps ambienti con più account (AWS Prescriptive Guidance)
- Introduzione allo sviluppo basato su Trunk (sito Web Trunk Based Development)

Argomenti in questa sezione:

- Panoramica visiva della strategia Trunk
- Strategia Branches in a Trunk
- Vantaggi e svantaggi della strategia Trunk

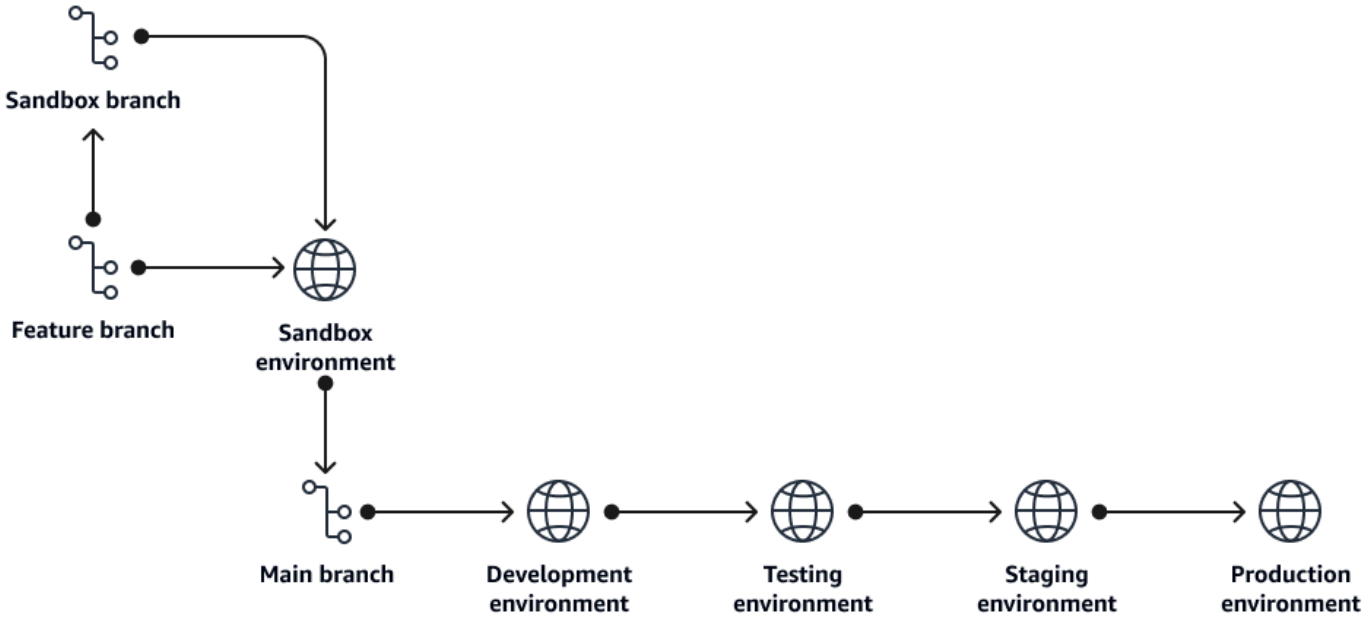
Panoramica visiva della strategia Trunk

Il diagramma seguente può essere usato come un quadrato di Punnett (Wikipedia) per comprendere la strategia di ramificazione Trunk. Allinea i rami sull'asse verticale con AWS gli ambienti sull'asse orizzontale per determinare quali azioni eseguire in ogni scenario. I numeri cerchiati guidano l'utente attraverso la sequenza di azioni rappresentata nel diagramma. Questo diagramma mostra il flusso di lavoro di sviluppo di una strategia di ramificazione Trunk, da una `feature` filiale nell'ambiente sandbox alla versione di produzione della filiale. `main` Per ulteriori informazioni sulle attività che si svolgono in ogni ambiente, consulta DevOps gli ambienti in questa guida.



Strategia Branches in a Trunk

Una strategia di ramificazione Trunk ha in genere i seguenti rami.



ramo di funzionalità

Sviluppate funzionalità o create un hotfix in un `feature` ramo. Per creare un `feature` ramo, si distacca dal `main` ramo. Gli sviluppatori eseguono iterazioni, eseguono il commit e testano il codice in un `feature` ramo. Quando una funzionalità è completa, lo sviluppatore la promuove. Ci sono solo due percorsi che partono da un `feature` ramo:

- Unisciti al ramo `sandbox`
- Crea una richiesta di unione nel ramo `main`

Convenzione di denominazione:	`feature/<story number>_<developer initials>_<descriptor>`
Esempio di convenzione di denominazione:	`feature/123456_MS_Implement_Feature_A`

ramo sandbox

Questo ramo è un branch principale non standard, ma è utile per lo sviluppo di pipeline CI/CD. Il `sandbox` ramo viene utilizzato principalmente per i seguenti scopi:

- Esegui una distribuzione completa nell'ambiente sandbox utilizzando le pipeline CI/CD
- Sviluppa e testa una pipeline prima di inviare richieste di unione per il test completo in un ambiente inferiore, ad esempio sviluppo o test.

`Sandbox`le filiali sono di natura temporanea e sono destinate a essere di breve durata. Dovrebbero essere cancellate dopo il completamento del test specifico.

Convenzione di denominazione:	`sandbox/<story number>_<developer initials>_<descriptor>`
Esempio di convenzione di denominazione:	`sandbox/123456_MS_Test_Pipeline_Deploy`

ramo principale

Il `main` ramo rappresenta sempre il codice in esecuzione in produzione. Il codice viene diramato da`main`, sviluppato e quindi ricongiunto a. `main` Le implementazioni di `main` potrebbero essere indirizzate a qualsiasi ambiente. Per proteggerla dall'eliminazione, abilita la protezione della `main` filiale.

Convenzione di denominazione:	`main`

ramo hotfix

Non esiste un `hotfix` ramo dedicato in un flusso di lavoro basato su trunk. Gli hotfix utilizzano i rami. `feature`

Vantaggi e svantaggi della strategia Trunk

La strategia di ramificazione Trunk è ideale per team di sviluppo più piccoli e maturi che hanno forti capacità di comunicazione. Funziona bene anche se si dispone di versioni continue e continue di funzionalità per l'applicazione. Non è adatto se avete team di sviluppo numerosi o frammentati o se avete rilasci di funzionalità espansivi e programmati. In questo modello si verificheranno conflitti di fusione, quindi tenete presente che la risoluzione dei conflitti di fusione è un'abilità chiave. Tutti i membri del team devono essere formati di conseguenza.

Vantaggi

Lo sviluppo basato su Trunk offre diversi vantaggi che possono migliorare il processo di sviluppo, semplificare la collaborazione e migliorare la qualità complessiva del software. Di seguito sono riportati alcuni dei vantaggi principali:

- Cicli di feedback più rapidi: con lo sviluppo basato su trunk, gli sviluppatori integrano le modifiche al codice frequentemente, spesso più volte al giorno. Ciò consente un feedback più rapido sui potenziali problemi e aiuta gli sviluppatori a identificare e risolvere i problemi più rapidamente di quanto farebbero in un modello di sviluppo basato su funzionalità.
- Riduzione dei conflitti di fusione: nello sviluppo basato su tronchi, il rischio di conflitti di fusione ampi e complicati è ridotto al minimo perché le modifiche vengono integrate continuamente. Questo aiuta a mantenere una base di codice più pulita e riduce la quantità di tempo impiegato per risolvere i conflitti. La risoluzione dei conflitti può richiedere molto tempo ed essere soggetta a errori nello sviluppo basato sulle funzionalità.
- Collaborazione migliorata: lo sviluppo basato su Trunk incoraggia gli sviluppatori a collaborare nello stesso ramo, promuovendo una migliore comunicazione e collaborazione all'interno del team. Ciò può portare a una risoluzione dei problemi più rapida e a una dinamica di squadra più coesa.
- Revisioni del codice più semplici: poiché le modifiche al codice sono più piccole e più frequenti nello sviluppo basato su trunk, può essere più semplice condurre revisioni approfondite del codice. Le modifiche più piccole sono generalmente più facili da comprendere e rivedere, il che consente un'identificazione più efficace di potenziali problemi e miglioramenti.
- Integrazione e distribuzione continue: lo sviluppo basato su Trunk supporta i principi dell'integrazione e della distribuzione continue (CI/CD). Mantenendo la base di codice in uno stato rilasciabile e integrando frequentemente le modifiche, i team possono adottare più facilmente le pratiche CI/CD, il che porta a cicli di implementazione più rapidi e a una migliore qualità del software.
- Migliore qualità del codice: grazie a integrazioni, test e revisioni del codice frequenti, lo sviluppo basato su trunk può contribuire a migliorare la qualità complessiva del codice. Gli sviluppatori possono catturare e risolvere i problemi più rapidamente, riducendo la probabilità che il debito tecnico si accumuli nel tempo.
- Strategia di ramificazione semplificata: lo sviluppo basato su Trunk semplifica la strategia di ramificazione riducendo il numero di filiali di lunga durata. Ciò può semplificare la gestione e la manutenzione della base di codice, in particolare per progetti o team di grandi dimensioni.

Svantaggi

Lo sviluppo basato su trunk presenta alcuni svantaggi, che possono influire sul processo di sviluppo e sulle dinamiche del team. Di seguito sono riportati alcuni importanti inconvenienti:

- Isolamento limitato: poiché tutti gli sviluppatori lavorano sullo stesso ramo, le modifiche apportate sono immediatamente visibili a tutti i membri del team. Ciò può causare interferenze o conflitti, causare effetti collaterali indesiderati o interrompere la build. Al contrario, lo sviluppo basato sulle funzionalità isola meglio le modifiche in modo che gli sviluppatori possano lavorare in modo più indipendente.
- Maggiore pressione sui test: lo sviluppo basato su Trunk si affida all'integrazione continua e ai test automatizzati per individuare rapidamente i problemi. Tuttavia, questo approccio può esercitare molta pressione sull'infrastruttura di test e richiede una suite di test ben gestita. Se i test non sono completi o affidabili, possono verificarsi problemi non rilevati nel ramo principale.
- Meno controllo sui rilasci: lo sviluppo basato su Trunk mira a mantenere la base di codice in uno stato di rilascio continuo. Sebbene ciò possa essere vantaggioso, potrebbe non essere sempre adatto a progetti con scadenze di rilascio rigorose o a quelli che richiedono il rilascio congiunto di funzionalità specifiche. Lo sviluppo basato sulle funzionalità offre un maggiore controllo su quando e come le funzionalità vengono rilasciate.
- Code churn: con gli sviluppatori che integrano costantemente le modifiche nel ramo principale, lo sviluppo basato su trunk può portare a un aumento del tasso di abbandono del codice. Ciò può rendere difficile per gli sviluppatori tenere traccia dello stato attuale del codice base e può creare confusione quando si cerca di comprendere l'effetto delle modifiche recenti.
- Richiede una solida cultura del team: lo sviluppo basato su Trunk richiede un elevato livello di disciplina, comunicazione e collaborazione tra i membri del team. Questo può essere difficile da mantenere, in particolare in team più grandi o quando si lavora con sviluppatori meno esperti con questo approccio.
- Sfide di scalabilità: con l'aumentare delle dimensioni del team di sviluppo, il numero di modifiche al codice da integrare nella filiale principale può aumentare rapidamente. Ciò può portare a interruzioni di compilazione e errori di test più frequenti, rendendo difficile mantenere la base di codice in uno stato rilasciabile.

GitHub Strategia di ramificazione del flusso

GitHub Flow è un flusso di lavoro leggero basato su filiali sviluppato da. GitHub GitHubFlow si basa sull'idea di rami di funzionalità di breve durata che vengono uniti nel ramo principale quando la funzionalità è completa e pronta per essere implementata. I principi chiave di Flow sono: GitHub

- La ramificazione è leggera: gli sviluppatori possono creare feature branch per il proprio lavoro con pochi clic, migliorando la capacità di collaborare e sperimentare senza influire sul ramo principale.
- Implementazione continua: le modifiche vengono implementate non appena vengono incorporate nella filiale principale, il che consente un feedback e un'iterazione rapidi.
- Richieste di unione: gli sviluppatori utilizzano le richieste di unione per avviare un processo di discussione e revisione prima di unire le modifiche nel ramo principale.

Per ulteriori informazioni su GitHub Flow, consulta le seguenti risorse:

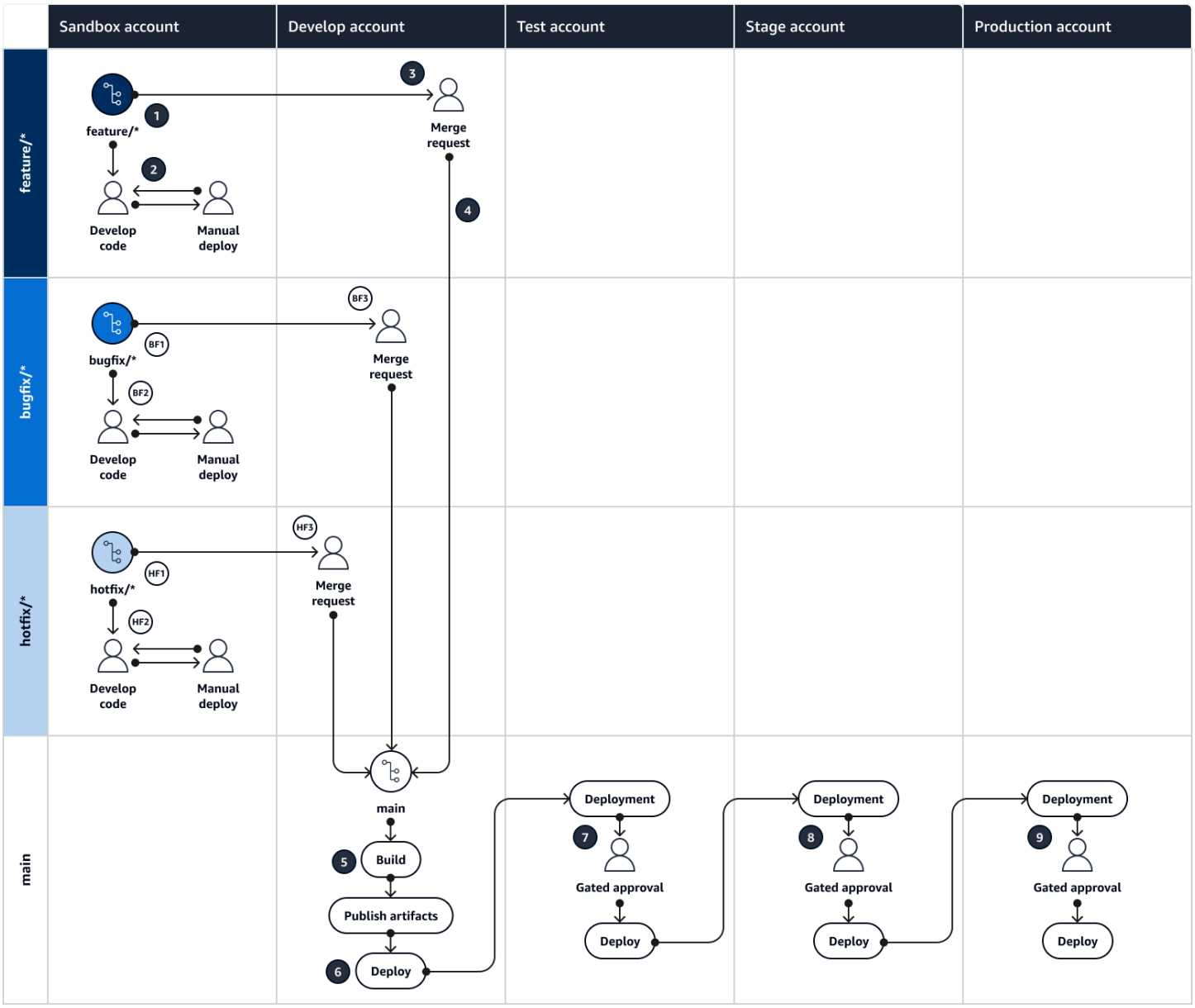
- Implementa una strategia di ramificazione GitHub Flow per DevOps ambienti con più account (AWS Prescriptive Guidance)
- GitHub Flow Quickstart (documentazione) GitHub
- Perché GitHub Flow? (Sito web GitHub Flow)

Argomenti in questa sezione:

- Panoramica visiva della strategia GitHub Flow
- Filiali in una GitHub strategia Flow
- Vantaggi e svantaggi della strategia GitHub Flow

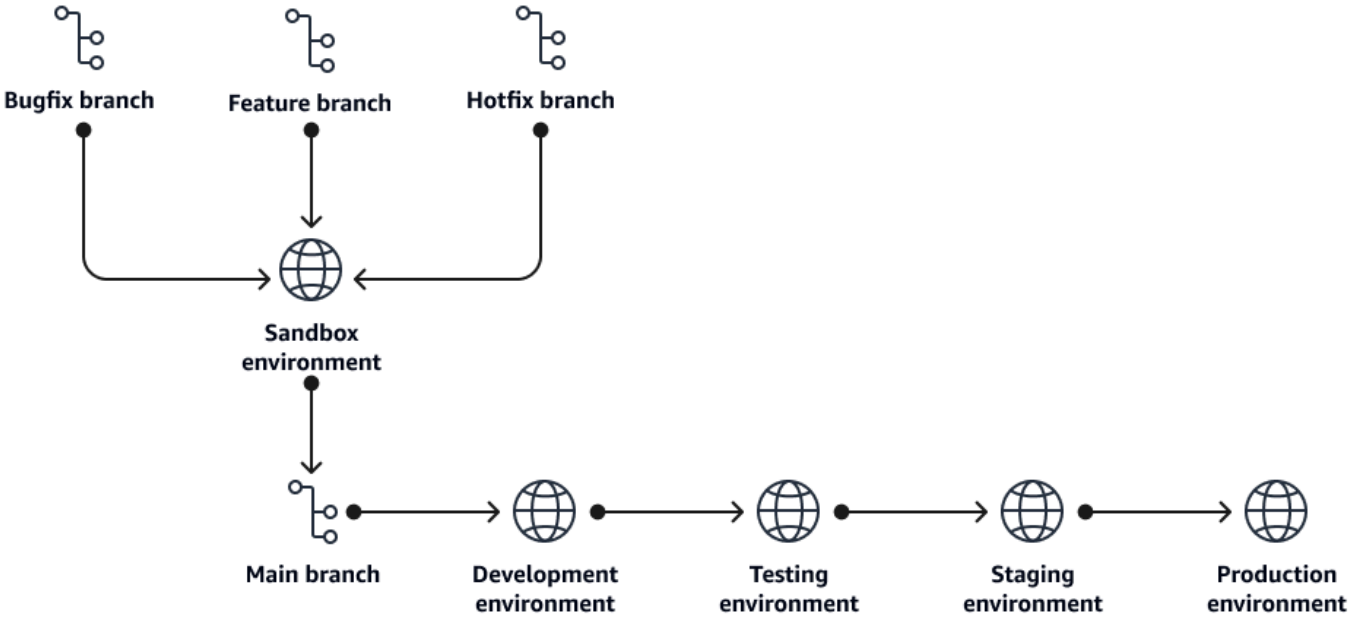
Panoramica visiva della strategia GitHub Flow

Il diagramma seguente può essere usato come un quadrato di Punnett per comprendere la strategia di ramificazione di Flow. GitHub Allinea i rami sull'asse verticale con AWS gli ambienti sull'asse orizzontale per determinare quali azioni eseguire in ogni scenario. I numeri cerchiati guidano l'utente attraverso la sequenza di azioni rappresentata nel diagramma. Questo diagramma mostra il flusso di lavoro di sviluppo di una strategia di ramificazione GitHub Flow, da un feature branch nell'ambiente sandbox alla release di produzione del branch principale. Per ulteriori informazioni sulle attività che si svolgono in ogni ambiente, consulta DevOps gli ambienti in questa guida.



Filiali in una GitHub strategia Flow

Una strategia GitHub di ramificazione Flow ha in genere i seguenti rami.



ramo di funzionalità

Sviluppi funzionalità nelle `feature` filiali. Per creare un `feature` ramo, si distacca dal `main` ramo. Gli sviluppatori eseguono iterazioni, eseguono il commit e testano il codice nel `feature` ramo. Quando una funzionalità è completa, lo sviluppatore la promuove creando una richiesta di unione a. `main`

Convenzione di denominazione:	`feature/<story number>_<developer initials>_<descriptor>`
Esempio di convenzione di denominazione:	`feature/123456_MS_Implement_Feature_A`

ramo bugfix

Il `bugfix` ramo viene utilizzato per risolvere i problemi. Questi rami si diramano dal `main` ramo. Dopo che il bugfix è stato testato nella sandbox o in uno qualsiasi degli ambienti inferiori, può essere promosso ad ambienti superiori unendolo a `main` una richiesta di unione. Questa è una convenzione di denominazione consigliata per l'organizzazione e il tracciamento; questo processo può anche essere gestito utilizzando un feature branch.

Convenzione di denominazione:	`bugfix/<ticket number>_<developer initials>_<descriptor>`
Esempio di convenzione di denominazione:	`bugfix/123456_MS_Fix_Problem_A`

ramo hotfix

Il `hotfix` branch viene utilizzato per risolvere problemi critici ad alto impatto con un ritardo minimo tra il personale di sviluppo e il codice distribuito in produzione. Queste filiali sono ramificate dalla `main` filiale. Dopo aver testato l'hotfix nella sandbox o in uno qualsiasi degli ambienti inferiori, può essere promosso ad ambienti superiori unendolo a `main` una richiesta di unione. Si tratta di una convenzione di denominazione consigliata per l'organizzazione e il monitoraggio; questo processo può anche essere gestito utilizzando un feature branch.

Convenzione di denominazione:	`hotfix/<ticket number>_<developer initials>_<descriptor>`
Esempio di convenzione di denominazione:	`hotfix/123456_MS_Fix_Problem_A`

ramo principale

Il `main` ramo rappresenta sempre il codice in esecuzione in produzione. Il codice viene unito al `main` ramo proveniente dalle `feature` filiali utilizzando le richieste di unione. Per proteggerti dalla cancellazione e impedire agli sviluppatori di inviare il codice direttamente al ramo`main`, abilita la protezione della filiale. `main`

Convenzione di denominazione:	`main`

Vantaggi e svantaggi della strategia GitHub Flow

La strategia di ramificazione di Github Flow è adatta a team di sviluppo più piccoli e maturi che hanno forti capacità di comunicazione. Questa strategia è adatta ai team che desiderano implementare la distribuzione continua ed è ben supportata dai motori CI/CD comuni. GitHub Flow è leggero, non ha troppe regole ed è in grado di supportare team in rapida evoluzione. Non è adatto se i team devono

seguire rigorosi processi di conformità o rilascio. I conflitti di fusione sono comuni in questo modello e probabilmente si verificheranno spesso. La risoluzione dei conflitti di fusione è un'abilità chiave ed è necessario addestrare di conseguenza tutti i membri del team.

Vantaggi

GitHub Flow offre diversi vantaggi che possono migliorare il processo di sviluppo, semplificare la collaborazione e migliorare la qualità complessiva del software. Di seguito sono riportati alcuni dei vantaggi principali:

- Flessibile e leggero: GitHub Flow è un flusso di lavoro leggero e flessibile che aiuta gli sviluppatori a collaborare su progetti di sviluppo software. Consente iterazioni e sperimentazioni rapide con una complessità minima.
- Collaborazione semplificata: GitHub Flow fornisce un processo chiaro e semplificato per la gestione dello sviluppo delle funzionalità. Incoraggia cambiamenti piccoli e mirati che possono essere rapidamente rivisti e uniti, migliorando l'efficienza.
- Controllo chiaro delle versioni: con GitHub Flow, ogni modifica viene apportata in un ramo separato. Ciò stabilisce una cronologia di controllo delle versioni chiara e tracciabile. Questo aiuta gli sviluppatori a tracciare e comprendere le modifiche, ripristinarle se necessario e mantenere una base di codice affidabile.
- Integrazione continua senza interruzioni: GitHub Flow si integra con strumenti di integrazione continua. La creazione di richieste pull può avviare processi di test e implementazione automatizzati. Gli strumenti CI consentono di testare a fondo le modifiche prima che vengano incorporate nel `main` ramo, riducendo il rischio di introdurre bug nella base di codice.
- Feedback rapido e miglioramento continuo: GitHub Flow incoraggia un ciclo di feedback rapido promuovendo revisioni e discussioni frequenti del codice tramite pull request. Ciò facilita l'individuazione precoce dei problemi, promuove la condivisione delle conoscenze tra i membri del team e, in ultima analisi, porta a una maggiore qualità del codice e a una migliore collaborazione all'interno del team di sviluppo.
- Ripristini e ripristini semplificati: nel caso in cui una modifica al codice introduca un bug o un problema imprevisto, GitHub Flow semplifica il processo di ripristino o ripristino della modifica. Grazie a una cronologia chiara di commit e branch, è più facile identificare e annullare le modifiche problematiche, contribuendo a mantenere una base di codice stabile e funzionale.
- Curva di apprendimento leggera: GitHub Flow può essere più facile da imparare e da adottare rispetto a Gitflow, soprattutto per i team che hanno già familiarità con Git e i concetti di controllo delle versioni. La sua semplicità e il suo modello di ramificazione intuitivo lo rendono accessibile

a sviluppatori con diversi livelli di esperienza, riducendo la curva di apprendimento associata all'adozione di nuovi flussi di lavoro di sviluppo.

- Sviluppo continuo: GitHub Flow consente ai team di adottare un approccio di implementazione continua, consentendo l'implementazione immediata di ogni modifica non appena viene incorporata nella filiale. `main` Questo processo semplificato elimina inutili ritardi e assicura che gli aggiornamenti e i miglioramenti più recenti siano rapidamente resi disponibili agli utenti. Ciò si traduce in un ciclo di sviluppo più agile e reattivo.

Svantaggi

Sebbene GitHub Flow offra diversi vantaggi, è importante considerare anche i suoi potenziali svantaggi:

- Idoneità limitata per progetti di grandi dimensioni: GitHub Flow potrebbe non essere altrettanto adatto per progetti su larga scala con basi di codice complesse e molteplici funzionalità a lungo termine. In questi casi, un flusso di lavoro più strutturato, come Gitflow, potrebbe fornire un migliore controllo sullo sviluppo simultaneo e sulla gestione delle release.
- Mancanza di una struttura di rilascio formale: GitHub Flow non definisce esplicitamente un processo di rilascio né supporta funzionalità come il controllo delle versioni, gli hotfix o i rami di manutenzione. Questa può essere una limitazione per i progetti che richiedono una gestione rigorosa dei rilasci o che necessitano di supporto e manutenzione a lungo termine.
- Supporto limitato per la pianificazione dei rilasci a lungo termine: GitHub Flow si concentra su funzionalità di breve durata, che potrebbero non adattarsi bene ai progetti che richiedono una pianificazione del rilascio a lungo termine, come quelli con roadmap rigorose o ampie dipendenze dalle funzionalità. La gestione di pianificazioni di rilascio complesse può essere difficile entro i limiti di Flow. GitHub
- Potenziale rischio di frequenti conflitti di fusione: poiché GitHub Flow incoraggia le ramificazioni e le fusioni frequenti, esiste la possibilità che si verifichino conflitti di fusione, specialmente nei progetti con molta attività di sviluppo. La risoluzione di questi conflitti può richiedere molto tempo e un ulteriore impegno da parte del team di sviluppo.
- Mancanza di fasi formalizzate del GitHub flusso di lavoro: Flow non definisce fasi esplicite per lo sviluppo, come le fasi alpha, beta o release candidate. Ciò può rendere più difficile la comunicazione dello stato attuale del progetto o del livello di stabilità di diverse filiali o versioni.
- Impatto delle modifiche interrotte: poiché GitHub Flow incoraggia l'unione frequente delle modifiche nel `main` ramo, esiste un rischio maggiore di introdurre modifiche sostanziali che influiscono sulla

stabilità della base di codice. Pratiche rigorose di revisione e test del codice sono fondamentali per mitigare efficacemente questo rischio.

Strategia di ramificazione Gitflow

Gitflow è un modello di ramificazione che prevede l'uso di più rami per spostare il codice dallo sviluppo alla produzione. Gitflow è ideale per i team che hanno cicli di rilascio programmati e devono definire una raccolta di funzionalità come release. Lo sviluppo viene completato in singoli rami di funzionalità che vengono uniti, previa approvazione, in un ramo di sviluppo, utilizzato per l'integrazione. Le funzionalità di questo ramo sono considerate pronte per la produzione. Quando tutte le funzionalità pianificate sono state accumulate nel ramo di sviluppo, viene creato un ramo di rilascio per le implementazioni negli ambienti superiori. Questa separazione migliora il controllo su quali modifiche vengono trasferite a quale ambiente denominato in base a una pianificazione definita. Se necessario, è possibile accelerare questo processo verso un modello di implementazione più rapido.

Per ulteriori informazioni sulla strategia di ramificazione di Gitflow, consulta le seguenti risorse:

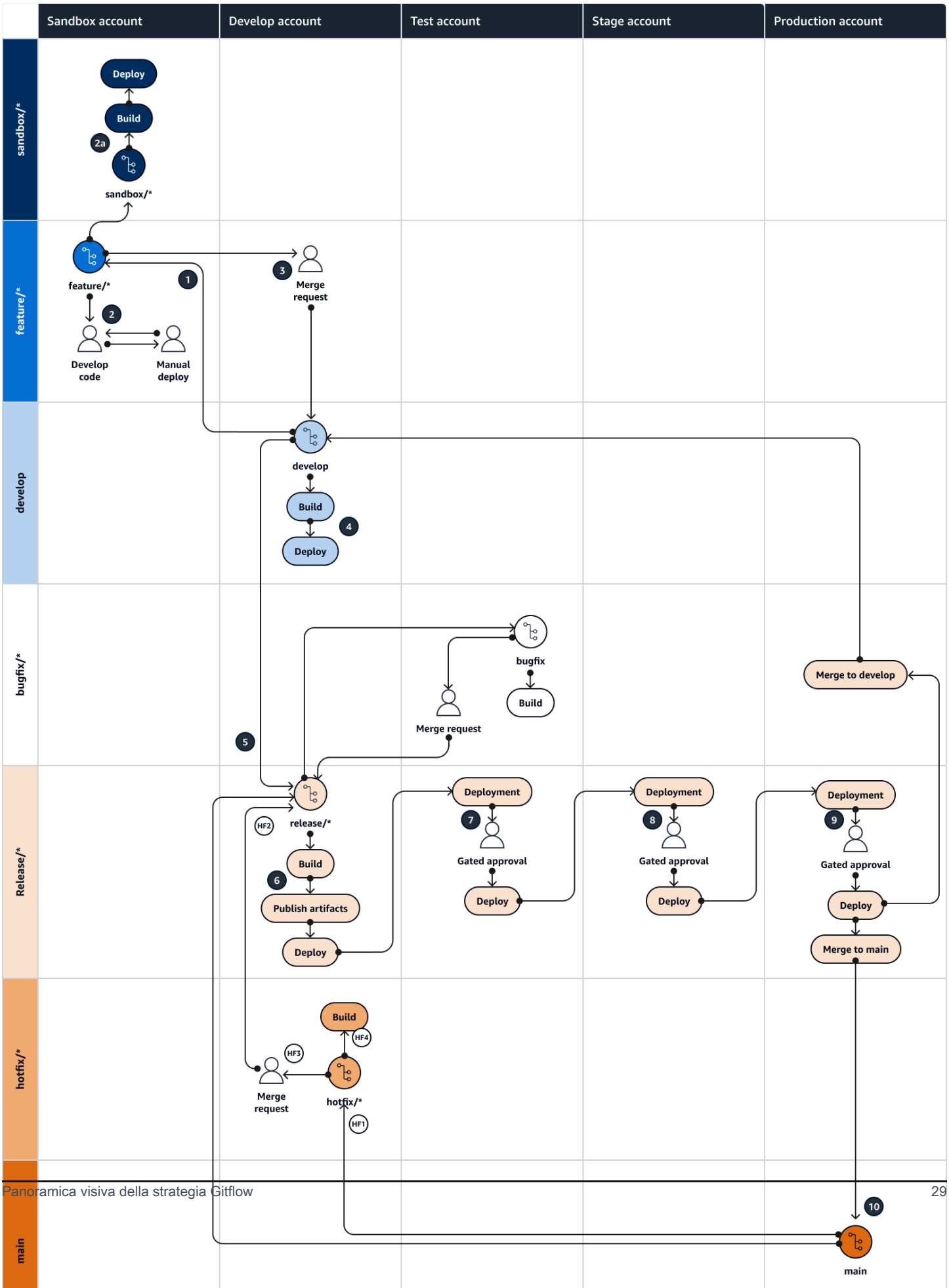
- Implementa una strategia di ramificazione Gitflow per ambienti con più account (Prescriptive Guidance) DevOps AWS
- Il blog originale di Gitflow (post sul blog di Vincent Driessen)
- Flusso di lavoro Gitflow (Atlassian)

Argomenti in questa sezione:

- Panoramica visiva della strategia Gitflow
- Filiali in una strategia Gitflow
- Vantaggi e svantaggi della strategia Gitflow

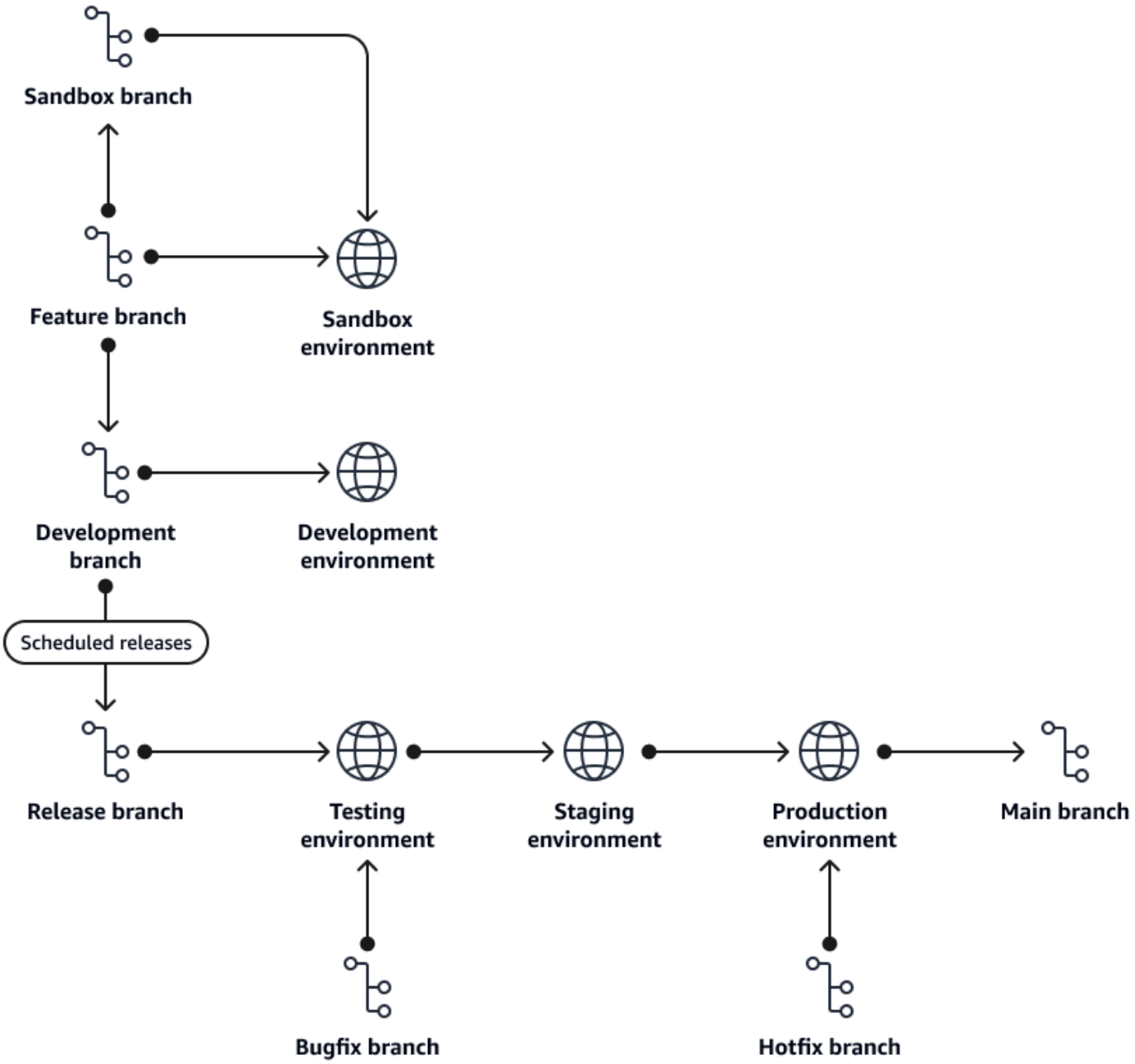
Panoramica visiva della strategia Gitflow

Il diagramma seguente può essere usato come un quadrato di Punnett per comprendere la strategia di ramificazione di Gitflow. Allinea i rami sull'asse verticale con AWS gli ambienti sull'asse orizzontale per determinare quali azioni eseguire in ogni scenario. I numeri cerchiati guidano l'utente attraverso la sequenza di azioni rappresentata nel diagramma. Per ulteriori informazioni sulle attività che si svolgono in ogni ambiente, consulta DevOps gli ambienti in questa guida.


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

Filiali in una strategia Gitflow

Una strategia di ramificazione Gitflow ha in genere i seguenti rami.



ramo di funzionalità

`Feature`ele filiali sono filiali a breve termine in cui si sviluppano funzionalità. Il `feature` ramo viene creato dalla diramazione del `develop` ramo. Gli sviluppatori eseguono iterazioni, eseguono il

commit e testano il `feature` codice nel ramo. Quando la funzionalità è completa, lo sviluppatore la promuove. Esistono solo due percorsi da seguire da un ramo di funzionalità:

- Unisciti al ramo `sandbox`
- Crea una richiesta di unione nel ramo `develop`

Convenzione di denominazione:	`feature/<story number>_<developer initials>_<descriptor>`
Esempio di convenzione di denominazione:	`feature/123456_MS_Implement_Feature_A`

ramo sandbox

La `sandbox` filiale è una filiale non standard a breve termine per Gitflow. Tuttavia, è utile per lo sviluppo di pipeline CI/CD. La `sandbox` filiale viene utilizzata principalmente per i seguenti scopi:

- Esegui una distribuzione completa nell'ambiente sandbox utilizzando le pipeline CI/CD anziché una distribuzione manuale.
- Sviluppa e testa una pipeline prima di inviare richieste di unione per il test completo in un ambiente inferiore, come lo sviluppo o il test.

`Sandbox`le filiali sono di natura temporanea e non sono destinate a durare a lungo. Dovrebbero essere cancellate dopo il completamento del test specifico.

Convenzione di denominazione:	`sandbox/<story number>_<developer initials>_<descriptor>`
Esempio di convenzione di denominazione:	`sandbox/123456_MS_Test_Pipeline_Deploy`

sviluppare un ramo

La `develop` filiale è una filiale longeva in cui le funzionalità vengono integrate, create, convalidate e implementate nell'ambiente di sviluppo. Tutte le `feature` filiali vengono unite nella filiale. `develop`

Le fusioni nel `develop` ramo vengono completate tramite una richiesta di unione che richiede una build corretta e due approvazioni da parte degli sviluppatori. Per impedire l'eliminazione, abilita la protezione del ramo sul ramo. `develop`

Convenzione di denominazione:	`develop`

ramo di rilascio

In Gitflow, le `release` filiali sono filiali a breve termine. Queste filiali sono speciali perché puoi distribuirle in più ambienti, adottando la metodologia build-once, deploy-many. `Release`le filiali possono rivolgersi agli ambienti di test, staging o produzione. Dopo che un team di sviluppo ha deciso di promuovere le funzionalità in ambienti superiori, crea una nuova `release` filiale e utilizza l'incremento del numero di versione rispetto alla versione precedente. Ai gate di ogni ambiente, le implementazioni richiedono approvazioni manuali per procedere. `Release`le filiali devono richiedere la modifica di una richiesta di unione.

Una volta che la `release` filiale è entrata in produzione, dovrebbe essere ricongiunta ai `main` rami `develop` and per assicurarsi che eventuali correzioni di bug o hotfix vengano reintegrati nelle attività di sviluppo future.

Convenzione di denominazione:	`release/v{major}.{minor}`
Esempio di convenzione di denominazione:	`release/v1.0`

ramo principale

Il `main` ramo è un ramo di lunga durata che rappresenta sempre il codice in esecuzione in produzione. Il codice viene unito automaticamente al `main` ramo da un ramo di rilascio dopo una corretta distribuzione dalla pipeline di rilascio. Per impedire l'eliminazione, abilita la protezione del ramo sul `main` ramo.

Convenzione di denominazione:	`main`

ramo bugfix

Il `bugfix` ramo è un ramo a breve termine che viene utilizzato per risolvere problemi nei rami di rilascio che non sono stati rilasciati in produzione. Una `bugfix` filiale deve essere utilizzata solo per promuovere le correzioni nelle `release` filiali negli ambienti di test, staging o produzione. Una `bugfix` filiale è sempre ramificata da una filiale. `release`

Dopo che il bugfix è stato testato, può essere promosso al `release` ramo tramite una richiesta di unione. Quindi puoi far avanzare il `release` ramo seguendo la procedura di rilascio standard.

Convenzione di denominazione:	`bugfix/<ticket>_<developer initials>_<descriptor>`
Esempio di convenzione di denominazione:	`bugfix/123456_MS_Fix_Problem_A`

ramo hotfix

La `hotfix` filiale è una filiale a breve termine utilizzata per risolvere problemi di produzione. Viene utilizzato solo per promuovere correzioni che devono essere accelerate per raggiungere l'ambiente di produzione. Un `hotfix` ramo è sempre ramificato da. `main`

Dopo aver testato l'hotfix, è possibile promuoverlo alla produzione tramite una richiesta di unione nel `release` ramo da cui è stato creato. `main` Per il test, è quindi possibile portare avanti il `release` ramo seguendo la procedura di rilascio standard.

Convenzione di denominazione:	`hotfix/<ticket>_<developer initials>_<descriptor>`
Esempio di convenzione di denominazione:	`hotfix/123456_MS_Fix_Problem_A`

Vantaggi e svantaggi della strategia Gitflow

La strategia di ramificazione di Gitflow è adatta a team più grandi e distribuiti che hanno requisiti di rilascio e conformità rigorosi. Gitflow contribuisce a un ciclo di rilascio prevedibile per l'organizzazione, e questo è spesso preferito dalle organizzazioni più grandi. Gitflow è ideale anche

per i team che necessitano di barriere per completare correttamente il ciclo di vita dello sviluppo del software. Questo perché ci sono molteplici opportunità di revisione e controllo della qualità integrate nella strategia. Gitflow è adatto anche per i team che devono mantenere contemporaneamente più versioni delle versioni di produzione. Alcuni svantaggi di GitFlow sono che è più complesso rispetto ad altri modelli di ramificazione e richiede una stretta aderenza allo schema per essere completato con successo. Gitflow non funziona bene per le organizzazioni che cercano una distribuzione continua a causa della natura rigida della gestione delle filiali di rilascio. Le filiali di rilascio di Gitflow possono essere filiali di lunga durata che possono accumulare debiti tecnici se non gestite adeguatamente in modo tempestivo.

Vantaggi

Lo sviluppo basato su Gitflow offre diversi vantaggi che possono migliorare il processo di sviluppo, semplificare la collaborazione e migliorare la qualità complessiva del software. Di seguito sono riportati alcuni dei vantaggi principali:

- Processo di rilascio prevedibile: Gitflow segue un processo di rilascio regolare e prevedibile. È adatto a team con cadenze di sviluppo e rilascio regolari.
- Collaborazione migliorata: Gitflow incoraggia l'uso di e filiali. `feature` `release` Queste due filiali aiutano i team a lavorare in parallelo con dipendenze minime l'una dall'altra.
- Adatto a più ambienti: Gitflow utilizza `release` i rami, che possono essere rami più longevi. Queste filiali consentono ai team di indirizzare i rilasci individuali per un periodo di tempo più lungo.
- Versioni multiple in produzione: se il tuo team supporta più versioni del software in produzione, le `release` filiali Gitflow supportano questo requisito.
- Revisioni integrate della qualità del codice: Gitflow richiede e incoraggia l'uso di revisioni e approvazioni del codice prima che il codice venga promosso in un altro ambiente. Questo processo elimina l'attrito tra gli sviluppatori richiedendo questo passaggio per tutte le promozioni di codice.
- Vantaggi organizzativi: Gitflow presenta vantaggi anche a livello di organizzazione. Gitflow incoraggia l'uso di un ciclo di rilascio standard, che aiuta l'organizzazione a comprendere e anticipare il programma di rilascio. Poiché l'azienda ora capisce quando è possibile fornire nuove funzionalità, le tempistiche sono ridotte grazie alle date di consegna prestabilite.

Svantaggi

Lo sviluppo basato su GitFlow presenta alcuni svantaggi che possono influire sul processo di sviluppo e sulle dinamiche del team. Di seguito sono riportati alcuni importanti inconvenienti:

- Complessità: Gitflow è un modello complesso da imparare per i nuovi team e devi rispettare le regole di Gitflow per utilizzarlo con successo.
- Distribuzione continua: Gitflow non si adatta a un modello in cui molte implementazioni vengono rilasciate in produzione in modo rapido. Questo perché Gitflow richiede l'uso di più filiali e un flusso di lavoro rigoroso che regola la filiale. `release`
- Gestione delle filiali: Gitflow utilizza molte filiali, la cui manutenzione può diventare onerosa. Può essere difficile tenere traccia delle varie filiali e unire il codice rilasciato per mantenere le filiali correttamente allineate tra loro.
- Debito tecnico: poiché le versioni di Gitflow sono in genere più lente rispetto agli altri modelli ramificati, prima del rilascio possono accumularsi più funzionalità, il che può causare l'accumulo di debiti tecnici.

I team devono considerare attentamente questi inconvenienti quando decidono se lo sviluppo basato su GitFlow sia l'approccio giusto per il loro progetto.

Passaggi successivi

Questa guida spiega le differenze tra tre strategie di ramificazione Git comuni: GitHub Flow, Gitflow e Trunk. Descrive in dettaglio i loro flussi di lavoro e fornisce anche i vantaggi e gli svantaggi di ciascuno. I passaggi successivi consistono nella scelta di uno di questi flussi di lavoro standard per l'organizzazione. Per implementare una di queste strategie di ramificazione, consulta quanto segue:

- Implementa una strategia di ramificazione Trunk per ambienti con più account DevOps
- Implementa una strategia di ramificazione GitHub Flow per ambienti con più account DevOps
- Implementa una strategia di ramificazione Gitflow per ambienti con più account DevOps

Se non sai da dove iniziare il percorso del tuo team verso l'utilizzo di Git e DevOps dei processi, ti consigliamo di scegliere una soluzione standard e testarla. L'utilizzo di una convenzione di ramificazione standard aiuta il team a sviluppare la documentazione esistente e a scoprire cosa funziona meglio per loro.

Non abbiate paura di cambiare la vostra strategia se non funziona per la vostra organizzazione o per i team di sviluppo. Le esigenze e i requisiti dei team di sviluppo possono cambiare nel tempo e non esiste un'unica soluzione perfetta.

Risorse

Questa guida non include la formazione per Git; tuttavia, ci sono molte risorse di alta qualità disponibili su Internet se hai bisogno di questa formazione. Ti consigliamo di iniziare dal sito di documentazione di Git.

Le seguenti risorse possono aiutarti nel tuo percorso di ramificazione di Git in. Cloud AWS

AWS Guida prescrittiva

- Implementa una strategia di ramificazione Trunk per ambienti con più account DevOps
- Implementa una strategia di ramificazione GitHub Flow per ambienti con più account DevOps
- Implementa una strategia di ramificazione Gitflow per ambienti con più account DevOps

Altre indicazioni AWS

- AWS DevOps Guida
- AWS Architettura di riferimento della pipeline di distribuzione
- Che cos'è DevOps?
- DevOpsrisorse

Altre risorse

- Metodologia delle app a dodici fattori (12factor.net)
- Segreti Git () GitHub
- Gitflow
  - Il blog originale di Gitflow (post sul blog di Vincent Driessen)
  - Flusso di lavoro Gitflow (Atlassian)
  - Gitflow su GitHub: Come usare i flussi di lavoro Git Flow con repository GitHub basati (video) YouTube
  - Esempio di inizializzazione di Git Flow (YouTube video)
  - Il ramo di rilascio di Gitflow dall'inizio alla fine (video) YouTube

- GitHub Flusso
  - GitHub Flow Quickstart (GitHub documentazione)
  - Perché GitHub Flow? (Sito web GitHub Flow)
- Tronco
  - Introduzione allo sviluppo basato su Trunk (sito Web Trunk Based Development)

Collaboratori

Creazione

- Mike Stephens, architetto senior di applicazioni cloud, AWS
- Rayjan Wilson, architetto senior di applicazioni cloud, AWS
- Abhilash Vinod, responsabile del team, architetto senior di applicazioni cloud, AWS

Revisione

- Stephen DiCato, consulente senior per la sicurezza, AWS
- Gaurav Samudra, architetto di applicazioni cloud, AWS
- Steven Guggenheimer, responsabile del team, architetto senior di applicazioni cloud, AWS

Scrittura tecnica

- Lilly AbouHarb, scrittrice tecnica senior, AWS

Cronologia dei documenti

La tabella seguente descrive le modifiche significative apportate a questa guida. Per ricevere notifiche sugli aggiornamenti futuri, puoi abbonarti a un feed RSS.

Modifica	Descrizione	Data
Pubblicazione iniziale	—	15 febbraio 2024

AWS Glossario delle linee guida prescrittive

I seguenti sono termini di uso comune nelle strategie, nelle guide e nei modelli forniti da AWS Prescriptive Guidance. Per suggerire voci, utilizza il link Fornisci feedback alla fine del glossario.

Numeri

7 R

Sette strategie di migrazione comuni per trasferire le applicazioni sul cloud. Queste strategie si basano sulle 5 R identificate da Gartner nel 2011 e sono le seguenti:

- Rifattorizzare/riprogettare: trasferisci un'applicazione e modifica la sua architettura sfruttando appieno le funzionalità native del cloud per migliorare l'agilità, le prestazioni e la scalabilità. Ciò comporta in genere la portabilità del sistema operativo e del database. Esempio: migra il tuo database Oracle locale all'edizione compatibile con Amazon Aurora SQL Postgre.
- Ridefinire la piattaforma (lift and reshape): trasferisci un'applicazione nel cloud e introduci un certo livello di ottimizzazione per sfruttare le funzionalità del cloud. Esempio: migra il tuo database Oracle locale ad Amazon Relational Database Service (AmazonRDS) per Oracle in. Cloud AWS
- Riacquistare (drop and shop): passa a un prodotto diverso, in genere effettuando la transizione da una licenza tradizionale a un modello SaaS. Esempio: migra il tuo sistema di gestione delle relazioni con i clienti (CRM) su Salesforce.com.
- Eseguire il rehosting (lift and shift): trasferisci un'applicazione sul cloud senza apportare modifiche per sfruttare le funzionalità del cloud. Esempio: migra il database Oracle locale su Oracle su un'istanza in. EC2 Cloud AWS
- Trasferire (eseguire il rehosting a livello hypervisor): trasferisci l'infrastruttura sul cloud senza acquistare nuovo hardware, riscrivere le applicazioni o modificare le operazioni esistenti. Si esegue la migrazione dei server da una piattaforma locale a un servizio cloud per la stessa piattaforma. Esempio: migrare un Microsoft Hyper-V applicazione a. AWS
- Riesaminare (mantenere): mantieni le applicazioni nell'ambiente di origine. Queste potrebbero includere applicazioni che richiedono una rifattorizzazione significativa che desideri rimandare a un momento successivo e applicazioni legacy che desideri mantenere, perché non vi è alcuna giustificazione aziendale per effettuarne la migrazione.
- Ritirare: disattiva o rimuovi le applicazioni che non sono più necessarie nell'ambiente di origine.

A

ABAC

Vedi controllo degli accessi basato sugli attributi.

servizi astratti

Vedi servizi gestiti.

ACID

Scopri atomicità, coerenza, isolamento e durata.

migrazione attiva-attiva

Un metodo di migrazione del database in cui i database di origine e di destinazione vengono mantenuti sincronizzati (utilizzando uno strumento di replica bidirezionale o operazioni di doppia scrittura) ed entrambi i database gestiscono le transazioni provenienti dalle applicazioni di connessione durante la migrazione. Questo metodo supporta la migrazione in piccoli batch controllati anziché richiedere una conversione una tantum. È più flessibile ma richiede più lavoro rispetto alla migrazione attiva-passiva.

migrazione attiva-passiva

Un metodo di migrazione di database in cui i database di origine e di destinazione vengono mantenuti sincronizzati, ma solo il database di origine gestisce le transazioni provenienti dalle applicazioni di connessione mentre i dati vengono replicati nel database di destinazione. Il database di destinazione non accetta alcuna transazione durante la migrazione.

funzione aggregata

Una SQL funzione che opera su un gruppo di righe e calcola un singolo valore restituito per il gruppo. Esempi di funzioni aggregate includono `SUM` e. `MAX`

Intelligenza artificiale

Vedi intelligenza artificiale.

AIOps

Guarda le operazioni di intelligenza artificiale.

anonimizzazione

Il processo di eliminazione permanente delle informazioni personali in un set di dati. L'anonimizzazione può aiutare a proteggere la privacy personale. I dati anonimi non sono più considerati dati personali.

anti-modello

Una soluzione utilizzata di frequente per un problema ricorrente in cui la soluzione è controproducente, inefficace o meno efficace di un'alternativa.

controllo delle applicazioni

Un approccio alla sicurezza che consente l'uso solo di applicazioni approvate per proteggere un sistema dal malware.

portfolio di applicazioni

Una raccolta di informazioni dettagliate su ogni applicazione utilizzata da un'organizzazione, compresi i costi di creazione e manutenzione dell'applicazione e il relativo valore aziendale. Queste informazioni sono fondamentali per il processo di scoperta e analisi del portfolio e aiutano a identificare e ad assegnare la priorità alle applicazioni da migrare, modernizzare e ottimizzare.

intelligenza artificiale (IA)

Il campo dell'informatica dedicato all'uso delle tecnologie informatiche per svolgere funzioni cognitive tipicamente associate agli esseri umani, come l'apprendimento, la risoluzione di problemi e il riconoscimento di schemi. Per ulteriori informazioni, consulta la sezione Che cos'è l'intelligenza artificiale?

operazioni di intelligenza artificiale (AIOps)

Il processo di utilizzo delle tecniche di machine learning per risolvere problemi operativi, ridurre gli incidenti operativi e l'intervento umano e aumentare la qualità del servizio. Per ulteriori informazioni su come AIOps viene utilizzata nella strategia di AWS migrazione, consulta la guida all'integrazione delle operazioni.

crittografia asimmetrica

Un algoritmo di crittografia che utilizza una coppia di chiavi, una chiave pubblica per la crittografia e una chiave privata per la decrittografia. Puoi condividere la chiave pubblica perché non viene utilizzata per la decrittografia, ma l'accesso alla chiave privata deve essere altamente limitato.

atomicità, consistenza, isolamento, durata () ACID

Un insieme di proprietà del software che garantiscono la validità dei dati e l'affidabilità operativa di un database, anche in caso di errori, interruzioni di corrente o altri problemi.

controllo degli accessi basato sugli attributi () ABAC

La pratica di creare autorizzazioni dettagliate basate su attributi utente, come reparto, ruolo professionale e nome del team. Per ulteriori informazioni, vedere ABACfor AWS nella documentazione AWS Identity and Access Management ()IAM.

fonte di dati autorevole

Una posizione in cui è archiviata la versione principale dei dati, considerata la fonte di informazioni più affidabile. È possibile copiare i dati dalla fonte di dati autorevole in altre posizioni allo scopo di elaborarli o modificarli, ad esempio anonimizzandoli, oscurandoli o pseudonimizzandoli.

Zona di disponibilità

Una posizione distinta all'interno di un edificio Regione AWS che è isolata dai guasti in altre zone di disponibilità e offre una connettività di rete economica e a bassa latenza verso altre zone di disponibilità nella stessa regione.

AWS Framework di adozione del cloud ()AWS CAF

Un framework di linee guida e best practice AWS per aiutare le organizzazioni a sviluppare un piano efficiente ed efficace per passare con successo al cloud. AWS CAForganizza le linee guida in sei aree di interesse chiamate prospettive: business, persone, governance, piattaforma, sicurezza e operazioni. Le prospettive relative ad azienda, persone e governance si concentrano sulle competenze e sui processi aziendali; le prospettive relative alla piattaforma, alla sicurezza e alle operazioni si concentrano sulle competenze e sui processi tecnici. Ad esempio, la prospettiva relativa alle persone si rivolge alle parti interessate che gestiscono le risorse umane (HR), le funzioni del personale e la gestione del personale. In questa prospettiva, AWS CAF fornisce linee guida per lo sviluppo del personale, la formazione e le comunicazioni per aiutare l'organizzazione a un'adozione efficace del cloud. Per ulteriori informazioni, consulta il AWS CAFsito Web e il AWS CAFwhite paper.

AWS Quadro di qualificazione del carico di lavoro ()AWS WQF

Uno strumento che valuta i carichi di lavoro di migrazione dei database, consiglia strategie di migrazione e fornisce stime del lavoro. AWS WQFè incluso in AWS Schema Conversion Tool ()AWS SCT. Analizza gli schemi di database e gli oggetti di codice, il codice dell'applicazione, le dipendenze e le caratteristiche delle prestazioni e fornisce report di valutazione.

B

bot difettoso

Un bot che ha lo scopo di interrompere o causare danni a individui o organizzazioni.

BCP

Vedi la pianificazione della continuità operativa.

grafico comportamentale

Una vista unificata, interattiva dei comportamenti delle risorse e delle interazioni nel tempo. Puoi utilizzare un grafico comportamentale con Amazon Detective per esaminare tentativi di accesso falliti, API chiamate sospette e azioni simili. Per ulteriori informazioni, consulta Dati in un grafico comportamentale nella documentazione di Detective.

sistema big-endian

Un sistema che memorizza per primo il byte più importante. Vedi anche endianness.

Classificazione binaria

Un processo che prevede un risultato binario (una delle due classi possibili). Ad esempio, il modello di machine learning potrebbe dover prevedere problemi come "Questa e-mail è spam o non è spam?" o "Questo prodotto è un libro o un'auto?"

filtro Bloom

Una struttura di dati probabilistica ed efficiente in termini di memoria che viene utilizzata per verificare se un elemento fa parte di un set.

distribuzioni blu/verdi

Una strategia di implementazione in cui si creano due ambienti separati ma identici. La versione corrente dell'applicazione viene eseguita in un ambiente (blu) e la nuova versione dell'applicazione nell'altro ambiente (verde). Questa strategia consente di ripristinare rapidamente il sistema con un impatto minimo.

bot

Un'applicazione software che esegue attività automatizzate su Internet e simula l'attività o l'interazione umana. Alcuni bot sono utili o utili, come i web crawler che indicizzano le informazioni su Internet. Alcuni altri bot, noti come bot dannosi, hanno lo scopo di disturbare o causare danni a individui o organizzazioni.

botnet

Reti di bot infettate da malware e controllate da un'unica parte, nota come bot herder o bot operator. Le botnet sono il meccanismo più noto per scalare i bot e il loro impatto.

ramo

Un'area contenuta di un repository di codice. Il primo ramo creato in un repository è il ramo principale. È possibile creare un nuovo ramo a partire da un ramo esistente e quindi sviluppare funzionalità o correggere bug al suo interno. Un ramo creato per sviluppare una funzionalità viene comunemente detto ramo di funzionalità. Quando la funzionalità è pronta per il rilascio, il ramo di funzionalità viene ricongiunto al ramo principale. Per ulteriori informazioni, consulta Informazioni sulle filiali (documentazione). GitHub

accesso break-glass

In circostanze eccezionali e tramite una procedura approvata, un mezzo rapido per consentire a un utente di accedere a un sito a Account AWS cui in genere non dispone delle autorizzazioni necessarie. Per ulteriori informazioni, vedere l'indicatore Implementate break-glass procedures nella guida Well-Architected AWS .

strategia brownfield

L'infrastruttura esistente nell'ambiente. Quando si adotta una strategia brownfield per un'architettura di sistema, si progetta l'architettura in base ai vincoli dei sistemi e dell'infrastruttura attuali. Per l'espansione dell'infrastruttura esistente, è possibile combinare strategie brownfield e greenfield.

cache del buffer

L'area di memoria in cui sono archiviati i dati a cui si accede con maggiore frequenza.

capacità di business

Azioni intraprese da un'azienda per generare valore (ad esempio vendite, assistenza clienti o marketing). Le architetture dei microservizi e le decisioni di sviluppo possono essere guidate dalle capacità aziendali. Per ulteriori informazioni, consulta la sezione Organizzazione in base alle funzionalità aziendali del whitepaper Esecuzione di microservizi containerizzati su AWS.

pianificazione della continuità operativa () BCP

Un piano che affronta il potenziale impatto di un evento che comporta l'interruzione dell'attività, come una migrazione su larga scala, sulle operazioni e consente a un'azienda di riprendere rapidamente le operazioni.

C

CAF

Vedi AWS Cloud Adoption Framework.

implementazione canaria

Il rilascio lento e incrementale di una versione agli utenti finali. Quando sei sicuro, distribuisci la nuova versione e sostituisci la versione corrente nella sua interezza.

CCoE

Vedi Cloud Center of Excellence.

CDC

Vedi Change Data Capture.

modifica l'acquisizione dei dati (CDC)

Il processo di tracciamento delle modifiche a un'origine dati, ad esempio una tabella di database, e di registrazione dei metadati relativi alla modifica. È possibile utilizzarlo CDC per vari scopi, come il controllo o la replica delle modifiche in un sistema di destinazione per mantenere la sincronizzazione.

ingegneria del caos

Introduzione intenzionale di guasti o eventi dirompenti per testare la resilienza di un sistema. Puoi usare AWS Fault Injection Service (AWS FIS) per eseguire esperimenti che stressano i tuoi AWS carichi di lavoro e valutarne la risposta.

CI/CD

Vedi integrazione continua e distribuzione continua.

classificazione

Un processo di categorizzazione che aiuta a generare previsioni. I modelli di ML per problemi di classificazione prevedono un valore discreto. I valori discreti sono sempre distinti l'uno dall'altro. Ad esempio, un modello potrebbe dover valutare se in un'immagine è presente o meno un'auto.

crittografia lato client

Crittografia dei dati a livello locale, prima che il destinatario li Servizio AWS riceva.

Centro di eccellenza cloud (CCoE)

Un team multidisciplinare che guida le iniziative di adozione del cloud in tutta l'organizzazione, tra cui lo sviluppo di best practice per il cloud, la mobilitazione delle risorse, la definizione delle tempistiche di migrazione e la guida dell'organizzazione attraverso trasformazioni su larga scala. Per ulteriori informazioni, consulta i CCoEpost sull' Cloud AWS Enterprise Strategy Blog.

cloud computing

La tecnologia cloud generalmente utilizzata per l'archiviazione remota di dati e la gestione dei dispositivi IoT. Il cloud computing è generalmente collegato alla tecnologia di edge computing.

modello operativo cloud

In un'organizzazione IT, il modello operativo utilizzato per creare, maturare e ottimizzare uno o più ambienti cloud. Per ulteriori informazioni, consulta Building your Cloud Operating Model.

fasi di adozione del cloud

Le quattro fasi che le organizzazioni in genere attraversano quando migrano verso Cloud AWS:

- Progetto: esecuzione di alcuni progetti relativi al cloud per scopi di dimostrazione e apprendimento
- Fondamento: effettuare investimenti fondamentali per scalare l'adozione del cloud (ad esempio, creazione di una landing zoneCCoE, definizione di un modello operativo)
- Migrazione: migrazione di singole applicazioni
- Reinvenzione: ottimizzazione di prodotti e servizi e innovazione nel cloud

Queste fasi sono state definite da Stephen Orban nel post sul blog The Journey Toward Cloud-First & the Stages of Adoption on the Enterprise Strategy. Cloud AWS Per informazioni su come si relazionano alla strategia di AWS migrazione, consulta la guida alla preparazione alla migrazione.

CMDB

Vedi database di gestione della configurazione.

repository di codice

Una posizione in cui il codice di origine e altri asset, come documentazione, esempi e script, vengono archiviati e aggiornati attraverso processi di controllo delle versioni. Gli archivi cloud comuni includono GitHub oppure Bitbucket Cloud. Ogni versione del codice è denominata branch.

In una struttura a microservizi, ogni repository è dedicato a una singola funzionalità. Una singola pipeline CI/CD può utilizzare più repository.

cache fredda

Una cache del buffer vuota, non ben popolata o contenente dati obsoleti o irrilevanti. Ciò influisce sulle prestazioni perché l'istanza di database deve leggere dalla memoria o dal disco principale, il che richiede più tempo rispetto alla lettura dalla cache del buffer.

dati freddi

Dati a cui si accede raramente e che in genere sono storici. Quando si eseguono interrogazioni di questo tipo di dati, le interrogazioni lente sono in genere accettabili. Lo spostamento di questi dati su livelli o classi di storage meno costosi e con prestazioni inferiori può ridurre i costi.

visione artificiale (CV)

Un campo dell'intelligenza artificiale che utilizza l'apprendimento automatico per analizzare ed estrarre informazioni da formati visivi come immagini e video digitali. Ad esempio, AWS Panorama offre dispositivi che aggiungono CV alle reti di telecamere locali e Amazon SageMaker fornisce algoritmi di elaborazione delle immagini per CV.

deriva della configurazione

Per un carico di lavoro, una modifica della configurazione rispetto allo stato previsto. Potrebbe causare la non conformità del carico di lavoro e in genere è graduale e involontaria.

database CMDB di gestione della configurazione ()

Un repository che archivia e gestisce le informazioni su un database e il relativo ambiente IT, inclusi i componenti hardware e software e le relative configurazioni. In genere si utilizzano i dati provenienti da una CMDB fase di individuazione e analisi del portafoglio durante la migrazione.

Pacchetto di conformità

Una raccolta di AWS Config regole e azioni correttive che puoi assemblare per personalizzare i controlli di conformità e sicurezza. È possibile distribuire un pacchetto di conformità come singola entità in una regione Account AWS AND o all'interno di un'organizzazione utilizzando un modello. YAML Per ulteriori informazioni, consulta i Conformance pack nella documentazione. AWS Config

integrazione e distribuzione continua (continuous integration and continuous delivery, CI/CD)

Il processo di automazione delle fasi di origine, creazione, test, gestione temporanea e produzione del processo di rilascio del software. CI/CD is commonly described as a pipeline. CI/CDpuò

aiutarti ad automatizzare i processi, migliorare la produttività, migliorare la qualità del codice e velocizzare le consegne. Per ulteriori informazioni, consulta Vantaggi della distribuzione continua. CD può anche significare continuous deployment (implementazione continua). Per ulteriori informazioni, consulta Distribuzione continua e implementazione continua a confronto.

CV

Vedi visione artificiale.

D

dati a riposo

Dati stazionari nella rete, ad esempio i dati archiviati.

classificazione dei dati

Un processo per identificare e classificare i dati nella rete in base alla loro criticità e sensibilità. È un componente fondamentale di qualsiasi strategia di gestione dei rischi di sicurezza informatica perché consente di determinare i controlli di protezione e conservazione appropriati per i dati. La classificazione dei dati è un componente del pilastro della sicurezza nel AWS Well-Architected Framework. Per ulteriori informazioni, consulta Classificazione dei dati.

deriva dei dati

Una variazione significativa tra i dati di produzione e i dati utilizzati per addestrare un modello di machine learning o una modifica significativa dei dati di input nel tempo. La deriva dei dati può ridurre la qualità, l'accuratezza e l'equità complessive nelle previsioni dei modelli ML.

dati in transito

Dati che si spostano attivamente attraverso la rete, ad esempio tra le risorse di rete.

rete di dati

Un framework architettonico che fornisce la proprietà distribuita e decentralizzata dei dati con gestione e governance centralizzate.

riduzione al minimo dei dati

Il principio della raccolta e del trattamento dei soli dati strettamente necessari. Praticare la riduzione al minimo dei dati in the Cloud AWS può ridurre i rischi per la privacy, i costi e l'impronta di carbonio delle analisi.

perimetro dei dati

Una serie di barriere preventive nell' AWS ambiente che aiutano a garantire che solo le identità attendibili accedano alle risorse attendibili delle reti previste. Per ulteriori informazioni, consulta Building a data perimeter on. AWS

pre-elaborazione dei dati

Trasformare i dati grezzi in un formato che possa essere facilmente analizzato dal modello di ML. La pre-elaborazione dei dati può comportare la rimozione di determinate colonne o righe e l'eliminazione di valori mancanti, incoerenti o duplicati.

provenienza dei dati

Il processo di tracciamento dell'origine e della cronologia dei dati durante il loro ciclo di vita, ad esempio il modo in cui i dati sono stati generati, trasmessi e archiviati.

soggetto dei dati

Un individuo i cui dati vengono raccolti ed elaborati.

data warehouse

Un sistema di gestione dei dati che supporta la business intelligence, come l'analisi. I data warehouse contengono in genere grandi quantità di dati storici e vengono generalmente utilizzati per interrogazioni e analisi.

linguaggio di definizione del database () DDL

Istruzioni o comandi per creare o modificare la struttura di tabelle e oggetti in un database.

linguaggio di manipolazione del database () DML

Istruzioni o comandi per modificare (inserire, aggiornare ed eliminare) informazioni in un database.

DDL

Vedi linguaggio di definizione del database.

deep ensemble

Combinare più modelli di deep learning per la previsione. È possibile utilizzare i deep ensemble per ottenere una previsione più accurata o per stimare l'incertezza nelle previsioni.

deep learning

Un sottocampo del ML che utilizza più livelli di reti neurali artificiali per identificare la mappatura tra i dati di input e le variabili target di interesse.

defense-in-depth

Un approccio alla sicurezza delle informazioni in cui una serie di meccanismi e controlli di sicurezza sono accuratamente stratificati su una rete di computer per proteggere la riservatezza, l'integrità e la disponibilità della rete e dei dati al suo interno. Quando si adotta questa strategia AWS, si aggiungono più controlli a diversi livelli della AWS Organizations struttura per proteggere le risorse. Ad esempio, un defense-in-depth approccio potrebbe combinare l'autenticazione a più fattori, la segmentazione della rete e la crittografia.

amministratore delegato

In AWS Organizations, un servizio compatibile può registrare un account AWS membro per amministrare gli account dell'organizzazione e gestire le autorizzazioni per quel servizio. Questo account è denominato amministratore delegato per quel servizio specifico. Per ulteriori informazioni e un elenco di servizi compatibili, consulta Servizi che funzionano con AWS Organizations nella documentazione di AWS Organizations .

implementazione

Il processo di creazione di un'applicazione, di nuove funzionalità o di correzioni di codice disponibili nell'ambiente di destinazione. L'implementazione prevede l'applicazione di modifiche in una base di codice, seguita dalla creazione e dall'esecuzione di tale base di codice negli ambienti applicativi.

Ambiente di sviluppo

Vedi ambiente.

controllo di rilevamento

Un controllo di sicurezza progettato per rilevare, registrare e avvisare dopo che si è verificato un evento. Questi controlli rappresentano una seconda linea di difesa e avvisano l'utente in caso di eventi di sicurezza che aggirano i controlli preventivi in vigore. Per ulteriori informazioni, consulta Controlli di rilevamento in Implementazione dei controlli di sicurezza in AWS.

mappatura del flusso di valore di sviluppo () DVSM

Un processo utilizzato per identificare e dare priorità ai vincoli che influiscono negativamente sulla velocità e sulla qualità nel ciclo di vita dello sviluppo del software. DVSMestende il processo di

mappatura del flusso di valore originariamente progettato per pratiche di produzione snella. Si concentra sulle fasi e sui team necessari per creare e trasferire valore attraverso il processo di sviluppo del software.

gemello digitale

Una rappresentazione virtuale di un sistema reale, ad esempio un edificio, una fabbrica, un'attrezzatura industriale o una linea di produzione. I gemelli digitali supportano la manutenzione predittiva, il monitoraggio remoto e l'ottimizzazione della produzione.

tabella delle dimensioni

In uno schema a stella, una tabella più piccola che contiene gli attributi dei dati quantitativi in una tabella dei fatti. Gli attributi della tabella delle dimensioni sono in genere campi di testo o numeri discreti che si comportano come testo. Questi attributi vengono comunemente utilizzati per il vincolo delle query, il filtraggio e l'etichettatura dei set di risultati.

disastro

Un evento che impedisce a un carico di lavoro o a un sistema di raggiungere gli obiettivi aziendali nella sua sede principale di implementazione. Questi eventi possono essere disastri naturali, guasti tecnici o il risultato di azioni umane, come errori di configurazione involontari o attacchi di malware.

disaster recovery (DR)

La strategia e il processo utilizzati per ridurre al minimo i tempi di inattività e la perdita di dati causati da un disastro. Per ulteriori informazioni, consulta Disaster Recovery of Workloads su AWS: Recovery in the Cloud in the AWS Well-Architected Framework.

DML

Vedi linguaggio di manipolazione del database.

progettazione basata sul dominio

Un approccio allo sviluppo di un sistema software complesso collegandone i componenti a domini in evoluzione, o obiettivi aziendali principali, perseguiti da ciascun componente. Questo concetto è stato introdotto da Eric Evans nel suo libro,Domain-Driven Design: Tackling Complexity in the Heart of Software (Boston: Addison-Wesley Professional, 2003). Per informazioni su come utilizzare la progettazione basata sul dominio con lo strangler fig pattern, vedi Modernizing legacy Microsoft. ASP NET(ASMX) servizi web in modo incrementale utilizzando contenitori e Amazon API Gateway.

DOTT.

Vedi disaster recovery.

rilevamento della deriva

Tracciamento delle deviazioni da una configurazione di base. Ad esempio, puoi utilizzarlo AWS CloudFormation per rilevare la deriva nelle risorse di sistema oppure puoi usarlo AWS Control Tower per rilevare cambiamenti nella tua landing zone che potrebbero influire sulla conformità ai requisiti di governance.

DVSM

Vedi la mappatura del flusso di valore dello sviluppo.

E

EDA

Vedi analisi esplorativa dei dati.

EDI

Vedi scambio elettronico di dati.

edge computing

La tecnologia che aumenta la potenza di calcolo per i dispositivi intelligenti all'edge di una rete IoT. Rispetto al cloud computing, l'edge computing può ridurre la latenza di comunicazione e migliorare i tempi di risposta.

scambio elettronico di dati () EDI

Lo scambio automatizzato di documenti commerciali tra organizzazioni. Per ulteriori informazioni, vedere Cos'è lo scambio elettronico di dati.

crittografia

Un processo di elaborazione che trasforma i dati in chiaro, leggibili dall'uomo, in testo cifrato.

chiave crittografica

Una stringa crittografica di bit randomizzati generata da un algoritmo di crittografia. Le chiavi possono variare di lunghezza e ogni chiave è progettata per essere imprevedibile e univoca.

endianità

L'ordine in cui i byte vengono archiviati nella memoria del computer. I sistemi big-endian memorizzano per primo il byte più importante. I sistemi little-endian memorizzano per primo il byte meno importante.

endpoint

Vedi service endpoint.

servizio endpoint

Un servizio che puoi ospitare in un cloud privato virtuale (VPC) per condividerlo con altri utenti. È possibile creare un servizio endpoint con AWS PrivateLink e concedere autorizzazioni ad altri Account AWS o a AWS Identity and Access Management (IAM) principali. Questi account o responsabili possono connettersi al servizio endpoint in modo privato creando endpoint di interfaccia. VPC Per ulteriori informazioni, consulta Creare un servizio endpoint nella documentazione di Amazon Virtual Private Cloud (AmazonVPC).

pianificazione delle risorse aziendali () ERP

Un sistema che automatizza e gestisce i processi aziendali chiave (come la contabilità e MESla gestione dei progetti) per un'azienda.

crittografia envelope

Il processo di crittografia di una chiave di crittografia con un'altra chiave di crittografia. Per ulteriori informazioni, vedete Envelope encryption nella documentazione AWS Key Management Service (AWS KMS).

ambiente

Un'istanza di un'applicazione in esecuzione. Di seguito sono riportati i tipi di ambiente più comuni nel cloud computing:

- ambiente di sviluppo: un'istanza di un'applicazione in esecuzione disponibile solo per il team principale responsabile della manutenzione dell'applicazione. Gli ambienti di sviluppo vengono utilizzati per testare le modifiche prima di promuoverle negli ambienti superiori. Questo tipo di ambiente viene talvolta definito ambiente di test.
- ambienti inferiori: tutti gli ambienti di sviluppo di un'applicazione, ad esempio quelli utilizzati per le build e i test iniziali.

- ambiente di produzione: un'istanza di un'applicazione in esecuzione a cui gli utenti finali possono accedere. In una pipeline CI/CD, l'ambiente di produzione è l'ultimo ambiente di implementazione.
- ambienti superiori: tutti gli ambienti a cui possono accedere utenti diversi dal team di sviluppo principale. Si può trattare di un ambiente di produzione, ambienti di preproduzione e ambienti per i test di accettazione da parte degli utenti.

epica

Nelle metodologie agili, categorie funzionali che aiutano a organizzare e dare priorità al lavoro. Le epiche forniscono una descrizione di alto livello dei requisiti e delle attività di implementazione. Ad esempio, le epopee relative AWS CAF alla sicurezza includono la gestione delle identità e degli accessi, i controlli investigativi, la sicurezza dell'infrastruttura, la protezione dei dati e la risposta agli incidenti. Per ulteriori informazioni sulle epiche, consulta la strategia di migrazione AWS , consulta la guida all'implementazione del programma.

ERP

Vedi la pianificazione delle risorse aziendali.

analisi esplorativa dei dati () EDA

Il processo di analisi di un set di dati per comprenderne le caratteristiche principali. Si raccolgono o si aggregano dati e quindi si eseguono indagini iniziali per trovare modelli, rilevare anomalie e verificare ipotesi. EDAviene eseguita calcolando statistiche riassuntive e creando visualizzazioni di dati.

F

tabella dei fatti

Il tavolo centrale in uno schema a stella. Memorizza dati quantitativi sulle operazioni aziendali. In genere, una tabella dei fatti contiene due tipi di colonne: quelle che contengono misure e quelle che contengono una chiave esterna per una tabella di dimensioni.

fallire velocemente

Una filosofia che utilizza test frequenti e incrementali per ridurre il ciclo di vita dello sviluppo. È una parte fondamentale di un approccio agile.

limite di isolamento dei guasti

Nel Cloud AWS, un limite come una zona di disponibilità Regione AWS, un piano di controllo o un piano dati che limita l'effetto di un errore e aiuta a migliorare la resilienza dei carichi di lavoro. Per ulteriori informazioni, consulta AWS Fault Isolation Boundaries.

ramo di funzionalità

Vedi filiale.

caratteristiche

I dati di input che usi per fare una previsione. Ad esempio, in un contesto di produzione, le caratteristiche potrebbero essere immagini acquisite periodicamente dalla linea di produzione.

importanza delle caratteristiche

Quanto è importante una caratteristica per le previsioni di un modello. Di solito viene espresso come punteggio numerico che può essere calcolato con varie tecniche, come Shapley Additive Explanations (SHAP) e gradienti integrati. Per ulteriori informazioni, vedere Interpretabilità del modello di machine learning con:.AWS

trasformazione delle funzionalità

Per ottimizzare i dati per il processo di machine learning, incluso l'arricchimento dei dati con fonti aggiuntive, il dimensionamento dei valori o l'estrazione di più set di informazioni da un singolo campo di dati. Ciò consente al modello di ML di trarre vantaggio dai dati. Ad esempio, se suddividi la data "2021-05-27 00:15:37" in "2021", "maggio", "giovedì" e "15", puoi aiutare l'algoritmo di apprendimento ad apprendere modelli sfumati associati a diversi componenti dei dati.

prompt con pochi scatti

Fornire un LLMnumero limitato di esempi che dimostrino l'attività e il risultato desiderato prima di chiedergli di eseguire un'attività simile. Questa tecnica è un'applicazione dell'apprendimento contestuale, in cui i modelli imparano da esempi (immagini) incorporati nei prompt. I prompt con pochi passaggi possono essere efficaci per attività che richiedono una formattazione, un ragionamento o una conoscenza del dominio specifici. Vedi anche zero-shot prompting.

FGAC

Vedi Controllo granulare degli accessi.

controllo granulare degli accessi () FGAC

L'uso di più condizioni per consentire o rifiutare una richiesta di accesso.

migrazione flash-cut

Un metodo di migrazione del database che utilizza la replica continua dei dati tramite l'acquisizione dei dati delle modifiche per migrare i dati nel più breve tempo possibile, anziché utilizzare un approccio graduale. L'obiettivo è ridurre al minimo i tempi di inattività.

FM

Vedi il modello di base.

modello di fondazione (FM)

Una grande rete neurale di deep learning che si è addestrata su enormi set di dati generalizzati e non etichettati. FMssono in grado di svolgere un'ampia varietà di attività generali, come comprendere il linguaggio, generare testo e immagini e conversare in linguaggio naturale. Per ulteriori informazioni, consulta Cosa sono i modelli Foundation.

G

AI generativa

Un sottoinsieme di modelli di intelligenza artificiale che sono stati addestrati su grandi quantità di dati e che possono utilizzare un semplice prompt di testo per creare nuovi contenuti e artefatti, come immagini, video, testo e audio. Per ulteriori informazioni, consulta Cos'è l'IA generativa.

blocco geografico

Vedi restrizioni geografiche.

limitazioni geografiche (blocco geografico)

In Amazon CloudFront, un'opzione per impedire agli utenti di determinati paesi di accedere alle distribuzioni di contenuti. Puoi utilizzare un elenco consentito o un elenco di blocco per specificare i paesi approvati e vietati. Per ulteriori informazioni, consulta Limitare la distribuzione geografica dei contenuti nella CloudFront documentazione.

Flusso di lavoro di GitFlow

Un approccio in cui gli ambienti inferiori e superiori utilizzano rami diversi in un repository di codice di origine. Il flusso di lavoro Gitflow è considerato obsoleto e il flusso di lavoro basato su trunk è l'approccio moderno e preferito.

immagine dorata

Un'istantanea di un sistema o di un software utilizzata come modello per distribuire nuove istanze di quel sistema o software. Ad esempio, nella produzione, un'immagine dorata può essere utilizzata per fornire software su più dispositivi e contribuire a migliorare la velocità, la scalabilità e la produttività nelle operazioni di produzione dei dispositivi.

strategia greenfield

L'assenza di infrastrutture esistenti in un nuovo ambiente. Quando si adotta una strategia greenfield per un'architettura di sistema, è possibile selezionare tutte le nuove tecnologie senza il vincolo della compatibilità con l'infrastruttura esistente, nota anche come brownfield. Per l'espansione dell'infrastruttura esistente, è possibile combinare strategie brownfield e greenfield.

guardrail

Una regola di alto livello che aiuta a governare le risorse, le politiche e la conformità tra le unità organizzative (). OUs I guardrail preventivi applicano le policy per garantire l'allineamento agli standard di conformità. Vengono implementate utilizzando le politiche di controllo del servizio e i limiti delle IAM autorizzazioni. I guardrail di rilevamento rilevano le violazioni delle policy e i problemi di conformità e generano avvisi per porvi rimedio. Sono implementati utilizzando Amazon AWS Config AWS Security Hub GuardDuty AWS Trusted Advisor, Amazon Inspector e controlli personalizzati AWS Lambda .

H

AH

Vedi disponibilità elevata.

migrazione di database eterogenea

Migrazione del database di origine in un database di destinazione che utilizza un motore di database diverso (ad esempio, da Oracle ad Amazon Aurora). La migrazione eterogenea fa in genere parte di uno sforzo di riprogettazione e la conversione dello schema può essere un'attività complessa. AWS offre AWS SCT che aiuta con le conversioni dello schema.

alta disponibilità (HA)

La capacità di un carico di lavoro di funzionare in modo continuo, senza intervento, in caso di sfide o disastri. I sistemi HA sono progettati per il failover automatico, fornire costantemente prestazioni di alta qualità e gestire carichi e guasti diversi con un impatto minimo sulle prestazioni.

modernizzazione storica

Un approccio utilizzato per modernizzare e aggiornare i sistemi di tecnologia operativa (OT) per soddisfare meglio le esigenze dell'industria manifatturiera. Uno storico è un tipo di database utilizzato per raccogliere e archiviare dati da varie fonti in una fabbrica.

dati di blocco

Una parte di dati storici etichettati che viene trattenuta da un set di dati utilizzata per addestrare un modello di apprendimento automatico. È possibile utilizzare i dati di holdout per valutare le prestazioni del modello confrontando le previsioni del modello con i dati di holdout.

migrazione di database omogenea

Migrazione del database di origine in un database di destinazione che condivide lo stesso motore di database (ad esempio, da Microsoft SQL Server ad Amazon RDS for SQL Server). La migrazione omogenea fa in genere parte di un'operazione di rehosting o ridefinizione della piattaforma. Per migrare lo schema è possibile utilizzare le utilità native del database.

dati caldi

Dati a cui si accede frequentemente, come dati in tempo reale o dati di traduzione recenti. Questi dati richiedono in genere un livello o una classe di storage ad alte prestazioni per fornire risposte rapide alle query.

hotfix

Una soluzione urgente per un problema critico in un ambiente di produzione. A causa della sua urgenza, un hotfix viene in genere creato al di fuori del tipico DevOps flusso di lavoro di rilascio.

periodo di hypercare

Subito dopo la conversione, il periodo di tempo in cui un team di migrazione gestisce e monitora le applicazioni migrate nel cloud per risolvere eventuali problemi. In genere, questo periodo dura da 1 a 4 giorni. Al termine del periodo di hypercare, il team addetto alla migrazione in genere trasferisce la responsabilità delle applicazioni al team addetto alle operazioni cloud.

I

IaC

Vedi l'infrastruttura come codice.

Policy basata su identità

Una politica allegata a uno o più IAM principi che definisce le relative autorizzazioni all'interno dell' Cloud AWS ambiente.

applicazione inattiva

Un'applicazione con un utilizzo medio CPU e della memoria compreso tra il 5 e il 20% in un periodo di 90 giorni. In un progetto di migrazione, è normale ritirare queste applicazioni o mantenerle on-premise.

IIoT

Vedi Industrial Internet of Things.

infrastruttura immutabile

Un modello che implementa una nuova infrastruttura per i carichi di lavoro di produzione anziché aggiornare, applicare patch o modificare l'infrastruttura esistente. Le infrastrutture immutabili sono intrinsecamente più coerenti, affidabili e prevedibili delle infrastrutture mutabili. Per ulteriori informazioni, consulta la best practice Deploy using immutable infrastructure in Well-Architected AWS Framework.

in entrata (ingresso) VPC

In un'architettura AWS multi-account, VPC che accetta, ispeziona e indirizza le connessioni di rete dall'esterno di un'applicazione. La AWS Security Reference Architecture consiglia di configurare l'account di rete con funzionalità in entrata, in uscita e di ispezione VPCs per proteggere l'interfaccia bidirezionale tra l'applicazione e Internet in generale.

migrazione incrementale

Una strategia di conversione in cui si esegue la migrazione dell'applicazione in piccole parti anziché eseguire una conversione singola e completa. Ad esempio, inizialmente potresti spostare solo alcuni microservizi o utenti nel nuovo sistema. Dopo aver verificato che tutto funzioni correttamente, puoi spostare in modo incrementale microservizi o utenti aggiuntivi fino alla disattivazione del sistema legacy. Questa strategia riduce i rischi associati alle migrazioni di grandi dimensioni.

Industria 4.0

Un termine introdotto da Klaus Schwab nel 2016 per riferirsi alla modernizzazione dei processi di produzione attraverso progressi in termini di connettività, dati in tempo reale, automazione, analisi e AI/ML.

infrastruttura

Tutte le risorse e gli asset contenuti nell'ambiente di un'applicazione.

infrastruttura come codice (IaC)

Il processo di provisioning e gestione dell'infrastruttura di un'applicazione tramite un insieme di file di configurazione. Il processo IaC è progettato per aiutarti a centralizzare la gestione dell'infrastruttura, a standardizzare le risorse e a dimensionare rapidamente, in modo che i nuovi ambienti siano ripetibili, affidabili e coerenti.

Internet delle cose industriale () IIoT

L'uso di sensori e dispositivi connessi a Internet nei settori industriali, come quello manifatturiero, energetico, automobilistico, sanitario, delle scienze della vita e dell'agricoltura. Per ulteriori informazioni, vedere Building an industrial Internet of Things (IIoT) strategia di trasformazione digitale.

ispezione VPC

In un'architettura AWS multi-account, un'architettura centralizzata VPC che gestisce le ispezioni del traffico di rete tra VPCs (nello stesso o in modo diverso Regioni AWS), Internet e le reti locali. La AWS Security Reference Architecture consiglia di configurare l'account di rete con funzioni in entrata, in uscita e di ispezione VPCs per proteggere l'interfaccia bidirezionale tra l'applicazione e Internet in generale.

Internet of Things (IoT)

La rete di oggetti fisici connessi con sensori o processori incorporati che comunicano con altri dispositivi e sistemi tramite Internet o una rete di comunicazione locale. Per ulteriori informazioni, consulta Cos'è l'IoT?

interpretabilità

Una caratteristica di un modello di machine learning che descrive il grado in cui un essere umano è in grado di comprendere in che modo le previsioni del modello dipendono dai suoi input. Per ulteriori informazioni, vedere Interpretabilità del modello di machine learning con. AWS

IoT

Vedi Internet of Things.

Libreria di informazioni IT (ITIL)

Una serie di best practice per offrire servizi IT e allinearli ai requisiti aziendali. ITILfornisce le basi perITSM.

gestione dei servizi IT (ITSM)

Attività associate alla progettazione, implementazione, gestione e supporto dei servizi IT per un'organizzazione. Per informazioni sull'integrazione delle operazioni cloud con ITSM gli strumenti, consulta la guida all'integrazione delle operazioni.

ITIL

Vedi la libreria di informazioni IT.

ITSM

Vedi Gestione dei servizi IT.

L

controllo degli accessi basato su etichette () LBAC

Un'implementazione del controllo di accesso obbligatorio (MAC) in cui agli utenti e ai dati stessi viene assegnato esplicitamente un valore di etichetta di sicurezza. L'intersezione tra l'etichetta di sicurezza dell'utente e l'etichetta di sicurezza dei dati determina quali righe e colonne possono essere visualizzate dall'utente.

zona di destinazione

Una landing zone è un AWS ambiente multi-account ben progettato, scalabile e sicuro. Questo è un punto di partenza dal quale le organizzazioni possono avviare e distribuire rapidamente carichi di lavoro e applicazioni con fiducia nel loro ambiente di sicurezza e infrastruttura. Per ulteriori informazioni sulle zone di destinazione, consulta la sezione Configurazione di un ambiente AWS multi-account sicuro e scalabile.

modello linguistico di grandi dimensioni () LLM

Un modello di intelligenza artificiale di deep learning preaddestrato su una grande quantità di dati. An LLM può eseguire più attività, come rispondere a domande, riepilogare documenti, tradurre testo in altre lingue e completare frasi. Per ulteriori informazioni, consulta Cosa sono. LLMs

migrazione su larga scala

Una migrazione di 300 o più server.

LBAC

Vedi Controllo degli accessi basato su etichette.

Privilegio minimo

La best practice di sicurezza per la concessione delle autorizzazioni minime richieste per eseguire un'attività. Per ulteriori informazioni, consulta Applicare le autorizzazioni con privilegi minimi nella documentazione. IAM

eseguire il rehosting (lift and shift)

Vedi 7 R.

sistema little-endian

Un sistema che memorizza per primo il byte meno importante. Vedi anche endianità.

LLM

Vedi modello linguistico di grandi dimensioni.

ambienti inferiori

Vedi ambiente.

M

machine learning (ML)

Un tipo di intelligenza artificiale che utilizza algoritmi e tecniche per il riconoscimento e l'apprendimento di schemi. Il machine learning analizza e apprende dai dati registrati, come i dati dell'Internet delle cose (IoT), per generare un modello statistico basato su modelli. Per ulteriori informazioni, consulta la sezione Machine learning.

ramo principale

Vedi filiale.

malware

Software progettato per compromettere la sicurezza o la privacy del computer. Il malware potrebbe interrompere i sistemi informatici, divulgare informazioni sensibili o ottenere accessi

non autorizzati. Esempi di malware includono virus, worm, ransomware, trojan horse, spyware e keylogger.

servizi gestiti

Servizi AWS per cui AWS gestisce il livello di infrastruttura, il sistema operativo e le piattaforme e si accede agli endpoint per archiviare e recuperare i dati. Amazon Simple Storage Service (Amazon S3) Simple Storage Service (Amazon S3) e Amazon DynamoDB sono esempi di servizi gestiti. Questi sono noti anche come servizi astratti.

sistema di esecuzione della produzione () MES

Un sistema software per tracciare, monitorare, documentare e controllare i processi di produzione che convertono le materie prime in prodotti finiti in officina.

MAP

Vedi Migration Acceleration Program.

meccanismo

Un processo completo in cui si crea uno strumento, si promuove l'adozione dello strumento e quindi si esaminano i risultati per apportare le modifiche. Un meccanismo è un ciclo che si rafforza e si migliora man mano che funziona. Per ulteriori informazioni, consulta Creazione di meccanismi nel AWS Well-Architected Framework.

account membro

Tutti gli account Account AWS diversi dall'account di gestione che fanno parte di un'organizzazione in. AWS Organizations Un account può essere membro di una sola organizzazione alla volta.

MES

Vedi Manufacturing Execution System.

Trasporto di telemetria in accodamento dei messaggi () MQTT

Un protocollo di comunicazione machine-to-machine (M2M) leggero, basato sul modello di pubblicazione/sottoscrizione, per dispositivi IoT con risorse limitate.

microservizio

Un servizio piccolo e indipendente che comunica tramite canali ben definiti ed è in genere di proprietà di piccoli team autonomi. APIs Ad esempio, un sistema assicurativo potrebbe includere

microservizi che si riferiscono a funzionalità aziendali, come vendite o marketing, o sottodomini, come acquisti, reclami o analisi. I vantaggi dei microservizi includono agilità, dimensionamento flessibile, facilità di implementazione, codice riutilizzabile e resilienza. Per ulteriori informazioni, consulta Integrazione dei microservizi utilizzando servizi serverless. AWS

architettura di microservizi

Un approccio alla creazione di un'applicazione con componenti indipendenti che eseguono ogni processo applicativo come microservizio. Questi microservizi comunicano attraverso un'interfaccia ben definita utilizzando sistemi leggeri. APIs Ogni microservizio in questa architettura può essere aggiornato, distribuito e dimensionato per soddisfare la richiesta di funzioni specifiche di un'applicazione. Per ulteriori informazioni, vedere Implementazione dei microservizi su. AWS

Programma MAP di accelerazione della migrazione ()

Un AWS programma che fornisce consulenza, supporto, formazione e servizi per aiutare le organizzazioni a costruire una solida base operativa per il passaggio al cloud e per contribuire a compensare il costo iniziale delle migrazioni. MAPinclude una metodologia di migrazione per eseguire le migrazioni precedenti in modo metodico e un set di strumenti per automatizzare e accelerare gli scenari di migrazione comuni.

migrazione su larga scala

Il processo di trasferimento della maggior parte del portfolio di applicazioni sul cloud avviene a ondate, con più applicazioni trasferite a una velocità maggiore in ogni ondata. Questa fase utilizza le migliori pratiche e le lezioni apprese nelle fasi precedenti per implementare una fabbrica di migrazione di team, strumenti e processi per semplificare la migrazione dei carichi di lavoro attraverso l'automazione e la distribuzione agile. Questa è la terza fase della strategia di migrazione AWS.

fabbrica di migrazione

Team interfunzionali che semplificano la migrazione dei carichi di lavoro attraverso approcci automatizzati e agili. I team di Migration Factory includono in genere operazioni, analisti e proprietari aziendali, ingegneri addetti alla migrazione, sviluppatori e DevOps professionisti che lavorano nell'ambito degli sprint. Tra il 20% e il 50% di un portfolio di applicazioni aziendali è costituito da schemi ripetuti che possono essere ottimizzati con un approccio di fabbrica. Per ulteriori informazioni, consulta la discussione sulle fabbriche di migrazione e la Guida alla fabbrica di migrazione al cloud in questo set di contenuti.

metadati di migrazione

Le informazioni sull'applicazione e sul server necessarie per completare la migrazione. Ogni modello di migrazione richiede un set diverso di metadati di migrazione. Esempi di metadati di migrazione includono la sottorete, il gruppo di sicurezza e l'account di destinazione. AWS

modello di migrazione

Un'attività di migrazione ripetibile che descrive in dettaglio la strategia di migrazione, la destinazione della migrazione e l'applicazione o il servizio di migrazione utilizzati. Esempio: riorganizza la migrazione su Amazon EC2 con AWS Application Migration Service.

Valutazione del portafoglio di migrazione () MPA

Uno strumento online che fornisce informazioni per la convalida del business case per la migrazione a. Cloud AWS MPAfornisce una valutazione dettagliata del portafoglio (dimensionamento corretto dei server, prezzi, TCO confronti, analisi dei costi di migrazione) e pianificazione della migrazione (analisi e raccolta dei dati delle applicazioni, raggruppamento delle applicazioni, prioritizzazione delle migrazioni e pianificazione delle ondate). Lo MPAstrumento (richiede l'accesso) è disponibile gratuitamente per tutti i consulenti e i consulenti partner. AWS APN

Valutazione della preparazione alla migrazione () MRA

Il processo di acquisizione di informazioni sullo stato di preparazione al cloud di un'organizzazione, l'identificazione dei punti di forza e di debolezza e la creazione di un piano d'azione per colmare le lacune identificate, utilizzando il. AWS CAF Per ulteriori informazioni, consulta la guida di preparazione alla migrazione. MRAè la prima fase della strategia di migrazione.AWS

strategia di migrazione

L'approccio utilizzato per migrare un carico di lavoro verso. Cloud AWS Per ulteriori informazioni, consulta la voce 7 R in questo glossario e consulta Mobilita la tua organizzazione per accelerare le migrazioni su larga scala.

ML

Vedi machine learning.

modernizzazione

Trasformazione di un'applicazione obsoleta (legacy o monolitica) e della relativa infrastruttura in un sistema agile, elastico e altamente disponibile nel cloud per ridurre i costi, aumentare

l'efficienza e sfruttare le innovazioni. Per ulteriori informazioni, vedere Strategia per la modernizzazione delle applicazioni in. Cloud AWS

valutazione della preparazione alla modernizzazione

Una valutazione che aiuta a determinare la preparazione alla modernizzazione delle applicazioni di un'organizzazione, identifica vantaggi, rischi e dipendenze e determina in che misura l'organizzazione può supportare lo stato futuro di tali applicazioni. Il risultato della valutazione è uno schema dell'architettura di destinazione, una tabella di marcia che descrive in dettaglio le fasi di sviluppo e le tappe fondamentali del processo di modernizzazione e un piano d'azione per colmare le lacune identificate. Per ulteriori informazioni, vedere Valutazione della preparazione alla modernizzazione per le applicazioni in. Cloud AWS

applicazioni monolitiche (monoliti)

Applicazioni eseguite come un unico servizio con processi strettamente collegati. Le applicazioni monolitiche presentano diversi inconvenienti. Se una funzionalità dell'applicazione registra un picco di domanda, l'intera architettura deve essere dimensionata. L'aggiunta o il miglioramento delle funzionalità di un'applicazione monolitica diventa inoltre più complessa man mano che la base di codice cresce. Per risolvere questi problemi, puoi utilizzare un'architettura di microservizi. Per ulteriori informazioni, consulta la sezione Scomposizione dei monoliti in microservizi.

MPA

Vedi Migration Portfolio Assessment.

MQTT

Vedi Message Queuing Telemetry Transport.

classificazione multiclasse

Un processo che aiuta a generare previsioni per più classi (prevedendo uno o più di due risultati). Ad esempio, un modello di machine learning potrebbe chiedere "Questo prodotto è un libro, un'auto o un telefono?" oppure "Quale categoria di prodotti è più interessante per questo cliente?"

infrastruttura mutabile

Un modello che aggiorna e modifica l'infrastruttura esistente per i carichi di lavoro di produzione. Per migliorare la coerenza, l'affidabilità e la prevedibilità, il AWS Well-Architected Framework consiglia l'uso di un'infrastruttura immutabile come best practice.

O

OAC

Vedi Origin Access Control.

OAI

Vedi Origin Access Identity.

OCM

Vedi gestione delle modifiche organizzative.

migrazione offline

Un metodo di migrazione in cui il carico di lavoro di origine viene eliminato durante il processo di migrazione. Questo metodo prevede tempi di inattività prolungati e viene in genere utilizzato per carichi di lavoro piccoli e non critici.

OI

Vedi l'integrazione delle operazioni.

OLA

Vedi accordo a livello operativo.

migrazione online

Un metodo di migrazione in cui il carico di lavoro di origine viene copiato sul sistema di destinazione senza essere messo offline. Le applicazioni connesse al carico di lavoro possono continuare a funzionare durante la migrazione. Questo metodo comporta tempi di inattività pari a zero o comunque minimi e viene in genere utilizzato per carichi di lavoro di produzione critici.

OPC-UA

Vedi Open Process Communications - Unified Architecture.

Comunicazioni a processo aperto - Architettura unificata (OPC-UA)

Un protocollo di comunicazione machine-to-machine (M2M) per l'automazione industriale. OPC-UA fornisce uno standard di interoperabilità con schemi di crittografia, autenticazione e autorizzazione dei dati.

accordo a livello operativo () OLA

Un accordo che chiarisce quali accordi tra i gruppi IT funzionali si impegnano a fornire i risultati reciproci, a supporto di un accordo sui livelli di servizio (). SLA

revisione della prontezza operativa () ORR

Un elenco di domande e best practice associate che aiutano a comprendere, valutare, prevenire o ridurre la portata degli incidenti e dei possibili guasti. Per ulteriori informazioni, vedere Operational Readiness Reviews (ORR) nel AWS Well-Architected Framework.

tecnologia operativa (OT)

Sistemi hardware e software che interagiscono con l'ambiente fisico per controllare le operazioni, le apparecchiature e le infrastrutture industriali. Nella produzione, l'integrazione di sistemi OT e di tecnologia dell'informazione (IT) è un obiettivo chiave per le trasformazioni dell'Industria 4.0.

integrazione delle operazioni (OI)

Il processo di modernizzazione delle operazioni nel cloud, che prevede la pianificazione, l'automazione e l'integrazione della disponibilità. Per ulteriori informazioni, consulta la guida all'integrazione delle operazioni.

trail organizzativo

Un percorso creato da noi AWS CloudTrail che registra tutti gli eventi di un'organizzazione per tutti Account AWS . AWS Organizations Questo percorso viene creato in ogni Account AWS che fa parte dell'organizzazione e tiene traccia dell'attività in ogni account. Per ulteriori informazioni, consulta Creazione di un percorso per un'organizzazione nella CloudTrail documentazione.

gestione delle modifiche organizzative (OCM)

Un framework per la gestione di trasformazioni aziendali importanti e che comportano l'interruzione delle attività dal punto di vista delle persone, della cultura e della leadership. OCMaiuta le organizzazioni a prepararsi e a passare a nuovi sistemi e strategie accelerando l'adozione del cambiamento, affrontando le questioni transitorie e promuovendo cambiamenti culturali e organizzativi. Nella strategia di AWS migrazione, questo framework si chiama accelerazione delle persone, a causa della velocità di cambiamento richiesta nei progetti di adozione del cloud. Per ulteriori informazioni, consulta la OCMguida.

controllo dell'accesso all'origine (OAC)

In CloudFront, un'opzione avanzata per limitare l'accesso per proteggere i contenuti di Amazon Simple Storage Service (Amazon S3). OACsupporta tutti i bucket S3 in generale Regioni AWS,

la crittografia lato server con AWS KMS (SSE-KMS) e la crittografia dinamica `PUT` e `DELETE` le richieste al bucket S3.

identità OAI di accesso all'origine ()

In CloudFront, un'opzione per limitare l'accesso per proteggere i tuoi contenuti Amazon S3. Quando lo usiOAI, CloudFront crea un principale con cui Amazon S3 può autenticarsi. I principali autenticati possono accedere ai contenuti in un bucket S3 solo tramite una distribuzione specifica. CloudFront Vedi anche OAC, che offre un controllo degli accessi più granulare e avanzato.

ORR

Vedi la revisione della prontezza operativa.

- NON

Vedi la tecnologia operativa.

in uscita (uscita) VPC

In un'architettura AWS multi-account, VPC che gestisce le connessioni di rete avviate dall'interno di un'applicazione. La AWS Security Reference Architecture consiglia di configurare l'account di rete con funzionalità in entrata, in uscita e di ispezione VPCs per proteggere l'interfaccia bidirezionale tra l'applicazione e Internet in generale.

P

limite delle autorizzazioni

Una politica di IAM gestione associata ai IAM principali per impostare le autorizzazioni massime che l'utente o il ruolo possono avere. Per ulteriori informazioni, consulta Limiti delle autorizzazioni nella documentazione. IAM

informazioni di identificazione personale () PII

Informazioni che, se visualizzate direttamente o abbinate ad altri dati correlati, possono essere utilizzate per dedurre ragionevolmente l'identità di un individuo. Alcuni esempi PII includono nomi, indirizzi e informazioni di contatto.

PII

Visualizza informazioni di identificazione personale.

playbook

Una serie di passaggi predefiniti che raccolgono il lavoro associato alle migrazioni, come l'erogazione delle funzioni operative principali nel cloud. Un playbook può assumere la forma di script, runbook automatici o un riepilogo dei processi o dei passaggi necessari per gestire un ambiente modernizzato.

PLC

Vedi controllore logico programmabile.

PLM

Vedi la gestione del ciclo di vita del prodotto.

policy

Un oggetto in grado di definire le autorizzazioni (vedi politica basata sull'identità), specificare le condizioni di accesso (vedi politicabasata sulle risorse) o definire le autorizzazioni massime per tutti gli account di un'organizzazione in (vedi politica di controllo dei servizi). AWS Organizations

persistenza poliglotta

Scelta indipendente della tecnologia di archiviazione di dati di un microservizio in base ai modelli di accesso ai dati e ad altri requisiti. Se i microservizi utilizzano la stessa tecnologia di archiviazione di dati, possono incontrare problemi di implementazione o registrare prestazioni scadenti. I microservizi vengono implementati più facilmente e ottengono prestazioni e scalabilità migliori se utilizzano l'archivio dati più adatto alle loro esigenze. Per ulteriori informazioni, consulta la sezione Abilitazione della persistenza dei dati nei microservizi.

valutazione del portfolio

Un processo di scoperta, analisi e definizione delle priorità del portfolio di applicazioni per pianificare la migrazione. Per ulteriori informazioni, consulta la pagina Valutazione della preparazione alla migrazione.

predicate

Una condizione di interrogazione che restituisce o, in genere, si trova in una clausola`true`. `false` `WHERE`

predicato pushdown

Una tecnica di ottimizzazione delle query del database che filtra i dati della query prima del trasferimento. Ciò riduce la quantità di dati che devono essere recuperati ed elaborati dal database relazionale e migliora le prestazioni delle query.

controllo preventivo

Un controllo di sicurezza progettato per impedire il verificarsi di un evento. Questi controlli sono la prima linea di difesa per impedire accessi non autorizzati o modifiche indesiderate alla rete. Per ulteriori informazioni, consulta Controlli preventivi in Implementazione dei controlli di sicurezza in AWS.

principale

Un'entità in AWS grado di eseguire azioni e accedere alle risorse. Questa entità è in genere un utente root per un Account AWS, un IAM ruolo o un utente. Per ulteriori informazioni, consulta i termini e i concetti di Principal in Roles nella IAM documentazione.

privacy fin dalla progettazione

Un approccio di ingegneria dei sistemi che tiene conto della privacy durante l'intero processo di sviluppo.

zone ospitate private

Un contenitore che contiene informazioni su come desideri che Amazon Route 53 risponda alle DNS richieste relative a un dominio e ai relativi sottodomini all'interno di uno o più. VPCs Per ulteriori informazioni, consulta Utilizzo delle zone ospitate private nella documentazione di Route 53.

controllo proattivo

Un controllo di sicurezza progettato per impedire l'implementazione di risorse non conformi. Questi controlli analizzano le risorse prima del loro provisioning. Se la risorsa non è conforme al controllo, non viene fornita. Per ulteriori informazioni, consulta la guida di riferimento sui controlli nella AWS Control Tower documentazione e consulta Controlli proattivi in Implementazione dei controlli di sicurezza su. AWS

gestione del ciclo di vita del prodotto () PLM

La gestione dei dati e dei processi di un prodotto durante l'intero ciclo di vita, dalla progettazione, sviluppo e lancio, attraverso la crescita e la maturità, fino al declino e alla rimozione.

Ambiente di produzione

Vedi ambiente.

controllore logico programmabile () PLC

Nella produzione, un computer altamente affidabile e adattabile che monitora le macchine e automatizza i processi di produzione.

concatenamento rapido

Utilizzo dell'output di un LLMprompt come input per il prompt successivo per generare risposte migliori. Questa tecnica viene utilizzata per suddividere un'attività complessa in sottoattività o per rifinire o espandere iterativamente una risposta preliminare. Aiuta a migliorare l'accuratezza e la pertinenza delle risposte di un modello e consente risultati più granulari e personalizzati.

pseudonimizzazione

Il processo di sostituzione degli identificatori personali in un set di dati con valori segnaposto. La pseudonimizzazione può aiutare a proteggere la privacy personale. I dati pseudonimizzati sono ancora considerati dati personali.

publish/subscribe (pub/sub)

Un modello che consente comunicazioni asincrone tra microservizi per migliorare la scalabilità e la reattività. Ad esempio, in un sistema basato su microservizi MES, un microservizio può pubblicare messaggi di eventi su un canale a cui altri microservizi possono abbonarsi. Il sistema può aggiungere nuovi microservizi senza modificare il servizio di pubblicazione.

Q

Piano di query

Una serie di passaggi, come le istruzioni, utilizzati per accedere ai dati in un sistema di database SQL relazionale.

regressione del piano di query

Quando un ottimizzatore del servizio di database sceglie un piano non ottimale rispetto a prima di una determinata modifica all'ambiente di database. Questo può essere causato da modifiche a statistiche, vincoli, impostazioni dell'ambiente, associazioni dei parametri di query e aggiornamenti al motore di database.

R

RACImatrice

Vedi responsabile, responsabile, consultato, informato () RACI.

RAG

Vedi Retrieval Augmented Generation.

ransomware

Un software dannoso progettato per bloccare l'accesso a un sistema informatico o ai dati fino a quando non viene effettuato un pagamento.

RASCImatrice

Vedi responsabile, responsabile, consultato, informato () RACI.

RCAC

Vedi controllo dell'accesso a righe e colonne.

replica di lettura

Una copia di un database utilizzata per scopi di sola lettura. È possibile indirizzare le query alla replica di lettura per ridurre il carico sul database principale.

riprogettare

Vedi 7 Rs.

obiettivo del punto di ripristino (RPO)

Il periodo di tempo massimo accettabile dall'ultimo punto di ripristino dei dati. Questo determina ciò che si considera una perdita di dati accettabile tra l'ultimo punto di ripristino e l'interruzione del servizio.

obiettivo del tempo di ripristino (RTO)

Il ritardo massimo accettabile tra l'interruzione del servizio e il ripristino del servizio.

rifattorizzare

Vedi 7 R.

Regione

Una raccolta di AWS risorse in un'area geografica. Ciascuna Regione AWS è isolata e indipendente dalle altre per fornire tolleranza agli errori, stabilità e resilienza. Per ulteriori informazioni, consulta Specificare cosa può utilizzare Regioni AWS il proprio account.

regressione

Una tecnica di ML che prevede un valore numerico. Ad esempio, per risolvere il problema "A che prezzo verrà venduta questa casa?" un modello di ML potrebbe utilizzare un modello di regressione lineare per prevedere il prezzo di vendita di una casa sulla base di dati noti sulla casa (ad esempio, la metratura).

riospitare

Vedi 7 R.

rilascio

In un processo di implementazione, l'atto di promuovere modifiche a un ambiente di produzione.

trasferisco

Vedi 7 Rs.

ripiattaforma

Vedi 7 Rs.

riacquisto

Vedi 7 Rs.

resilienza

La capacità di un'applicazione di resistere o ripristinare le interruzioni. L'elevata disponibilità e il disaster recovery sono considerazioni comuni quando si pianifica la resilienza in. Cloud AWSPer ulteriori informazioni, vedere Cloud AWS Resilience.

policy basata su risorse

Una policy associata a una risorsa, ad esempio un bucket Amazon S3, un endpoint o una chiave di crittografia. Questo tipo di policy specifica a quali principali è consentito l'accesso, le azioni supportate e qualsiasi altra condizione che deve essere soddisfatta.

matrice responsabile, responsabile, consultata, informata () RACI

Una matrice che definisce i ruoli e le responsabilità di tutte le parti coinvolte nelle attività di migrazione e nelle operazioni cloud. Il nome della matrice deriva dai tipi di responsabilità definiti nella matrice: responsabile (R), responsabile (A), consultato (C) e informato (I). Il tipo di supporto (S) è facoltativo. Se includi il supporto, la matrice viene chiamata RASCImatrice e se la escludi, viene chiamata RACImatrice.

controllo reattivo

Un controllo di sicurezza progettato per favorire la correzione di eventi avversi o deviazioni dalla baseline di sicurezza. Per ulteriori informazioni, consulta Controlli reattivi in Implementazione dei controlli di sicurezza in AWS.

retain

Vedi 7 R.

andare in pensione

Vedi 7 Rs.

Generazione aumentata di recupero () RAG

Una tecnologia di intelligenza artificiale generativa in cui un LLMfa riferimento a una fonte di dati autorevole esterna alle sue fonti di dati di addestramento prima di generare una risposta. Ad esempio, un RAG modello potrebbe eseguire una ricerca semantica nella knowledge base o nei dati personalizzati di un'organizzazione. Per ulteriori informazioni, consulta Cos'è RAG.

rotazione

Processo di aggiornamento periodico di un segreto per rendere più difficile l'accesso alle credenziali da parte di un utente malintenzionato.

controllo dell'accesso a righe e colonne () RCAC

L'uso di SQL espressioni di base e flessibili con regole di accesso definite. RCACè costituito da permessi di riga e maschere di colonna.

RPO

Vedi obiettivo del punto di ripristino.

RTO

Vedi l'obiettivo del tempo di ripristino.

runbook

Un insieme di procedure manuali o automatizzate necessarie per eseguire un'attività specifica. In genere sono progettati per semplificare operazioni o procedure ripetitive con tassi di errore elevati.

S

SAML2.0

Uno standard aperto utilizzato da molti provider di identità (IdPs). Questa funzionalità abilita il single sign-on federato (SSO), in modo che gli utenti possano accedere AWS Management Console o richiamare le AWS API operazioni senza che sia necessario creare un account utente IAM per tutti i membri dell'organizzazione. Per ulteriori informazioni sulla federazione SAML basata sulla versione 2.0, vedere Informazioni sulla federazione basata sulla versione SAML 2.0 nella documentazione. IAM

SCADA

Vedi controllo di supervisione e acquisizione dati.

SCP

Vedi la politica di controllo del servizio.

Secret

In AWS Secrets Manager, informazioni riservate o riservate, come una password o le credenziali utente, archiviate in forma crittografata. È costituito dal valore segreto e dai relativi metadati. Il valore segreto può essere binario, una stringa singola o più stringhe. Per ulteriori informazioni, consulta Cosa c'è in un segreto di Secrets Manager? nella documentazione di Secrets Manager.

sicurezza fin dalla progettazione

Un approccio di ingegneria dei sistemi che tiene conto della sicurezza durante l'intero processo di sviluppo.

controllo di sicurezza

Un guardrail tecnico o amministrativo che impedisce, rileva o riduce la capacità di un autore di minacce di sfruttare una vulnerabilità di sicurezza. Esistono quattro tipi principali di controlli di sicurezza: preventivi, investigativi, reattivi e proattivi.

rafforzamento della sicurezza

Il processo di riduzione della superficie di attacco per renderla più resistente agli attacchi. Può includere azioni come la rimozione di risorse che non sono più necessarie, l'implementazione di best practice di sicurezza che prevedono la concessione del privilegio minimo o la disattivazione di funzionalità non necessarie nei file di configurazione.

sistema di gestione delle informazioni e degli eventi di sicurezza () SIEM

Strumenti e servizi che combinano sistemi di gestione delle informazioni di sicurezza (SIM) e di gestione degli eventi di sicurezza (SEM). Un SIEM sistema raccoglie, monitora e analizza i dati da server, reti, dispositivi e altre fonti per rilevare minacce e violazioni della sicurezza e generare avvisi.

automazione della risposta di sicurezza

Un'azione predefinita e programmata progettata per rispondere o porre rimedio automaticamente a un evento di sicurezza. Queste automazioni fungono da controlli di sicurezza investigativi o reattivi che aiutano a implementare le migliori pratiche di sicurezza. AWS Esempi di azioni di risposta automatizzate includono la modifica di un gruppo di VPC sicurezza, l'applicazione di patch a un'EC2istanza Amazon o la rotazione delle credenziali.

Crittografia lato server

Crittografia dei dati a destinazione, da parte di chi li riceve. Servizio AWS

politica di controllo del servizio (SCP)

Una policy che fornisce il controllo centralizzato sulle autorizzazioni per tutti gli account di un'organizzazione in AWS Organizations. SCPsdefinisci barriere o imposta limiti alle azioni che un amministratore può delegare a utenti o ruoli. È possibile utilizzarli SCPs come elenchi consentiti o elenchi di rifiuto, per specificare quali servizi o azioni sono consentiti o proibiti. Per ulteriori informazioni, consulta le politiche di controllo del servizio nella AWS Organizations documentazione.

endpoint del servizio

Il punto URL di ingresso per un Servizio AWS. Puoi utilizzare l'endpoint per connetterti a livello di programmazione al servizio di destinazione. Per ulteriori informazioni, consulta Endpoint del Servizio AWS nei Riferimenti generali di AWS.

accordo sul livello di servizio () SLA

Un accordo che chiarisce ciò che un team IT promette di offrire ai propri clienti, ad esempio l'operatività e le prestazioni del servizio.

indicatore del livello di servizio () SLI

Misurazione di un aspetto prestazionale di un servizio, ad esempio il tasso di errore, la disponibilità o la velocità effettiva.

obiettivo a livello di servizio () SLO

Una metrica target che rappresenta lo stato di un servizio, misurato da un indicatore del livello di servizio.

Modello di responsabilità condivisa

Un modello che descrive la responsabilità condivisa AWS per la sicurezza e la conformità del cloud. AWS è responsabile della sicurezza del cloud, mentre tu sei responsabile della sicurezza nel cloud. Per ulteriori informazioni, consulta Modello di responsabilità condivisa.

SIEM

Vedi il sistema di gestione delle informazioni e degli eventi sulla sicurezza.

singolo punto di errore (SPOF)

Un guasto in un singolo componente critico di un'applicazione che può disturbare il sistema.

SLA

Vedi il contratto sui livelli di servizio.

SLI

Vedi l'indicatore del livello di servizio.

SLO

Vedi l'obiettivo del livello di servizio.

split-and-seed modello

Un modello per dimensionare e accelerare i progetti di modernizzazione. Man mano che vengono definite nuove funzionalità e versioni dei prodotti, il team principale si divide per creare nuovi team di prodotto. Questo aiuta a dimensionare le capacità e i servizi dell'organizzazione, migliora la produttività degli sviluppatori e supporta una rapida innovazione. Per ulteriori informazioni, vedere Approccio graduale alla modernizzazione delle applicazioni in. Cloud AWS

SPOF

Vedere Single Point of Failure.

schema a stella

Una struttura organizzativa di database che utilizza un'unica tabella dei fatti di grandi dimensioni per archiviare i dati transazionali o misurati e utilizza una o più tabelle dimensionali più piccole per memorizzare gli attributi dei dati. Questa struttura è progettata per l'uso in un data warehouse o per scopi di business intelligence.

modello del fico strangolatore

Un approccio alla modernizzazione dei sistemi monolitici mediante la riscrittura e la sostituzione incrementali delle funzionalità del sistema fino alla disattivazione del sistema legacy. Questo modello utilizza l'analogia di una pianta di fico che cresce fino a diventare un albero robusto e alla fine annienta e sostituisce il suo ospite. Il modello è stato introdotto da Martin Fowler come metodo per gestire il rischio durante la riscrittura di sistemi monolitici. Per un esempio di come applicare questo modello, vedi Modernizing legacy MicrosoftASP. NET(ASMX) servizi web in modo incrementale utilizzando contenitori e Amazon API Gateway.

sottorete

Una gamma di indirizzi IP nel tuoVPC. Una sottorete deve risiedere in una singola zona di disponibilità.

controllo di supervisione e acquisizione dati () SCADA

Nella produzione, un sistema che utilizza hardware e software per monitorare gli asset fisici e le operazioni di produzione.

crittografia simmetrica

Un algoritmo di crittografia che utilizza la stessa chiave per crittografare e decrittografare i dati.

test sintetici

Test di un sistema in modo da simulare le interazioni degli utenti per rilevare potenziali problemi o monitorare le prestazioni. Puoi usare Amazon CloudWatch Synthetics per creare questi test.

prompt di sistema

Tecnica per fornire contesto, istruzioni o linee guida a un utente per LLMindirizzarne il comportamento. I prompt di sistema aiutano a definire il contesto e stabilire regole per le interazioni con gli utenti.

T

tags

Coppie chiave-valore che fungono da metadati per l'organizzazione delle risorse. AWS Con i tag è possibile a gestire, identificare, organizzare, cercare e filtrare le risorse. Per ulteriori informazioni, consulta Tagging delle risorse AWS.

variabile di destinazione

Il valore che stai cercando di prevedere nel machine learning supervisionato. Questo è indicato anche come variabile di risultato. Ad esempio, in un ambiente di produzione la variabile di destinazione potrebbe essere un difetto del prodotto.

elenco di attività

Uno strumento che viene utilizzato per tenere traccia dei progressi tramite un runbook. Un elenco di attività contiene una panoramica del runbook e un elenco di attività generali da completare. Per ogni attività generale, include la quantità stimata di tempo richiesta, il proprietario e lo stato di avanzamento.

Ambiente di test

Vedi ambiente.

training

Fornire dati da cui trarre ispirazione dal modello di machine learning. I dati di training devono contenere la risposta corretta. L'algoritmo di apprendimento trova nei dati di addestramento i pattern che mappano gli attributi dei dati di input al target (la risposta che si desidera prevedere). Produce un modello di ML che acquisisce questi modelli. Puoi quindi utilizzare il modello di ML per creare previsioni su nuovi dati di cui non si conosce il target.

Transit Gateway

Un hub di transito di rete che puoi utilizzare per interconnettere le tue reti VPCs e quelle locali. Per ulteriori informazioni, consulta Cos'è un gateway di transito nella AWS Transit Gateway documentazione.

flusso di lavoro basato su trunk

Un approccio in cui gli sviluppatori creano e testano le funzionalità localmente in un ramo di funzionalità e quindi uniscono tali modifiche al ramo principale. Il ramo principale viene quindi integrato negli ambienti di sviluppo, preproduzione e produzione, in sequenza.

Accesso attendibile

Concessione delle autorizzazioni a un servizio specificato dall'utente per eseguire attività all'interno dell'organizzazione AWS Organizations e nei suoi account per conto dell'utente. Il servizio attendibile crea un ruolo collegato al servizio in ogni account, quando tale ruolo è necessario, per eseguire attività di gestione per conto dell'utente. Per ulteriori informazioni, consulta Utilizzo AWS Organizations con altri AWS servizi nella AWS Organizations documentazione.

regolazione

Modificare alcuni aspetti del processo di training per migliorare la precisione del modello di ML. Ad esempio, puoi addestrare il modello di ML generando un set di etichette, aggiungendo etichette e quindi ripetendo questi passaggi più volte con impostazioni diverse per ottimizzare il modello.

team da due pizze

Una piccola DevOps squadra che puoi sfamare con due pizze. Un team composto da due persone garantisce la migliore opportunità possibile di collaborazione nello sviluppo del software.

U

incertezza

Un concetto che si riferisce a informazioni imprecise, incomplete o sconosciute che possono minare l'affidabilità dei modelli di machine learning predittivi. Esistono due tipi di incertezza: l'incertezza epistemica, che è causata da dati limitati e incompleti, mentre l'incertezza aleatoria è causata dal rumore e dalla casualità insiti nei dati. Per ulteriori informazioni, consulta la guida Quantificazione dell'incertezza nei sistemi di deep learning.

compiti indifferenziati

Conosciuto anche come sollevamento di carichi pesanti, è un lavoro necessario per creare e far funzionare un'applicazione, ma che non apporta valore diretto all'utente finale né offre vantaggi competitivi. Esempi di attività indifferenziate includono l'approvvigionamento, la manutenzione e la pianificazione della capacità.

ambienti superiori

Vedi ambiente.

V

vacuum

Un'operazione di manutenzione del database che prevede la pulizia dopo aggiornamenti incrementali per recuperare lo spazio di archiviazione e migliorare le prestazioni.

controllo delle versioni

Processi e strumenti che tengono traccia delle modifiche, ad esempio le modifiche al codice di origine in un repository.

VPCscrutando

Una connessione tra due VPCs che consente di indirizzare il traffico utilizzando indirizzi IP privati. Per ulteriori informazioni, consulta What is VPC peering nella VPC documentazione di Amazon.

vulnerabilità

Un difetto software o hardware che compromette la sicurezza del sistema.

W

cache calda

Una cache del buffer che contiene dati correnti e pertinenti a cui si accede frequentemente. L'istanza di database può leggere dalla cache del buffer, il che richiede meno tempo rispetto alla lettura dalla memoria dal disco principale.

dati caldi

Dati a cui si accede raramente. Quando si eseguono interrogazioni di questo tipo di dati, in genere sono accettabili query moderatamente lente.

funzione finestra

Una SQL funzione che esegue un calcolo su un gruppo di righe che si riferiscono in qualche modo al record corrente. Le funzioni della finestra sono utili per l'elaborazione di attività, come il calcolo di una media mobile o l'accesso al valore delle righe in base alla posizione relativa della riga corrente.

Carico di lavoro

Una raccolta di risorse e codice che fornisce valore aziendale, ad esempio un'applicazione rivolta ai clienti o un processo back-end.

flusso di lavoro

Gruppi funzionali in un progetto di migrazione responsabili di una serie specifica di attività. Ogni flusso di lavoro è indipendente ma supporta gli altri flussi di lavoro del progetto. Ad esempio, il flusso di lavoro del portfolio è responsabile della definizione delle priorità delle applicazioni, della pianificazione delle ondate e della raccolta dei metadati di migrazione. Il flusso di lavoro del portfolio fornisce queste risorse al flusso di lavoro di migrazione, che quindi migra i server e le applicazioni.

WORM

Vedi write once, read many.

WQF

Vedi AWSWorkload Qualification Framework.

scrivi una volta, leggi molte () WORM

Un modello di archiviazione che scrive i dati una sola volta e ne impedisce l'eliminazione o la modifica. Gli utenti autorizzati possono leggere i dati tutte le volte che è necessario, ma non possono modificarli. Questa infrastruttura di archiviazione dei dati è considerata immutabile.

Z

exploit zero-day

Un attacco, in genere malware, che sfrutta una vulnerabilità zero-day.

vulnerabilità zero-day

Un difetto o una vulnerabilità assoluta in un sistema di produzione. Gli autori delle minacce possono utilizzare questo tipo di vulnerabilità per attaccare il sistema. Gli sviluppatori vengono spesso a conoscenza della vulnerabilità causata dall'attacco.

prompt zero-shot

Fornisce istruzioni per eseguire un'LLMattività, ma non fornisce esempi (immagini) che possano aiutarla. LLMDeve utilizzare le proprie conoscenze pre-addestrate per gestire l'attività. L'efficacia

del prompt zero-shot dipende dalla complessità dell'attività e dalla qualità del prompt. Vedi anche few-shot prompting.

applicazione zombie

Un'applicazione con un utilizzo medio CPU e della memoria inferiore al 5%. In un progetto di migrazione, è normale ritirare queste applicazioni.

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.