



Costruire architetture esagonali su AWS

AWS Guida prescrittiva



AWS Guida prescrittiva: Costruire architetture esagonali su AWS

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

I marchi e l'immagine commerciale di Amazon non possono essere utilizzati in relazione a prodotti o servizi che non siano di Amazon, in una qualsiasi modalità che possa causare confusione tra i clienti o in una qualsiasi modalità che denigri o discrediti Amazon. Tutti gli altri marchi non di proprietà di Amazon sono di proprietà dei rispettivi proprietari, che possono o meno essere affiliati, collegati o sponsorizzati da Amazon.

Table of Contents

Introduzione	1
Panoramica	3
Progettazione basata sul dominio (DDD)	3
Architettura esagonale	3
Obiettivi aziendali specifici	5
Migliorare il ciclo di sviluppo	6
Test nel cloud	6
Test in locale	6
Parallelizzazione dello sviluppo	7
Tempi di commercializzazione del prodotto	7
Qualità fin dal design	8
Modifiche localizzate e maggiore leggibilità	8
Testare innanzitutto la logica aziendale	8
Manutenibilità	9
Adattarsi al cambiamento	10
Adattamento a nuovi requisiti non funzionali utilizzando porte e adattatori	10
Adattamento ai nuovi requisiti aziendali utilizzando comandi e gestori di comandi	10
Disaccoppiamento dei componenti utilizzando la facciata di servizio o il pattern CQRS	11
Scalabilità organizzativa	12
Best practice	14
Modella il dominio aziendale	14
Scrivi ed esegui test fin dall'inizio	14
Definisci il comportamento del dominio	15
Automatizza i test e l'implementazione	15
Scala il tuo prodotto utilizzando microservizi e CQRS	15
Progetta una struttura di progetto che si adatti ai concetti di architettura esagonale	16
Esempi di infrastrutture	18
Inizia in modo semplice	18
Applica il pattern CQRS	19
Fai evolvere l'architettura aggiungendo contenitori, un database relazionale e un'API esterna	20
Aggiungi altri domini (rimpicciolisci)	21
Domande frequenti	23
Perché dovrei usare un'architettura esagonale?	23
Perché utilizzare la progettazione basata sul dominio?	23

Posso praticare lo sviluppo basato sui test senza un'architettura esagonale?	23
Posso scalare il mio prodotto senza un'architettura esagonale e una progettazione basata sul dominio?	23
Quali tecnologie devo usare per implementare l'architettura esagonale?	23
Sto sviluppando un prodotto minimo valido. Ha senso dedicare del tempo a pensare all'architettura del software?	24
Sto sviluppando un prodotto minimamente funzionante e non ho tempo per scrivere test.	24
Quali modelli di progettazione aggiuntivi posso utilizzare con l'architettura esagonale?	24
Passaggi successivi	25
Resources	26
Cronologia dei documenti	28
Glossario	29
#	29
A	30
B	33
C	35
D	38
E	42
F	44
G	46
H	47
I	48
L	51
M	52
O	56
P	59
Q	62
R	62
S	65
T	69
U	70
V	71
W	71
Z	72
.....	lxxiv

Costruire architetture esagonali su AWS

Furkan Oruc, Dominik Goby, Darius Kuncze e Michal Ploski, Amazon Web Services (AWS)

Giugno 2022 ([cronologia dei documenti](#))

Questa guida descrive un modello mentale e una raccolta di modelli per lo sviluppo di architetture software. Queste architetture sono facili da mantenere, estendere e scalare in tutta l'organizzazione man mano che cresce l'adozione dei prodotti. Gli hyperscaler cloud come Amazon Web Services (AWS) forniscono elementi costitutivi per le piccole e grandi imprese per innovare e creare nuovi prodotti software. La rapida introduzione di questi nuovi servizi e funzionalità induce gli stakeholder aziendali ad aspettarsi che i propri team di sviluppo prototipino nuovi prodotti minimi validi (MVPs) più rapidamente, in modo che le nuove idee possano essere testate e verificate il prima possibile. Spesso MVPs vengono adottate e diventano parte dell'ecosistema software aziendale. Nel processo di produzione MVPs, i team a volte abbandonano le regole e le migliori pratiche di sviluppo del software, come [i principi SOLID](#) e [i test unitari](#). Ritengono che questo approccio accelererà lo sviluppo e ridurrà i tempi di commercializzazione. Tuttavia, se non riusciranno a creare un modello fondamentale e un framework per l'architettura software a tutti i livelli, sarà difficile o addirittura impossibile sviluppare nuove funzionalità per il prodotto. La mancanza di certezza e il cambiamento dei requisiti possono inoltre rallentare il team durante il processo di sviluppo.

Questa guida illustra l'architettura software proposta, da un'architettura esagonale di basso livello a una scomposizione architettonica e organizzativa di alto livello, che utilizza la progettazione basata sul dominio (DDD) per affrontare queste sfide. DDD aiuta a gestire la complessità aziendale e a scalare il team di progettazione man mano che vengono sviluppate nuove funzionalità. Allinea gli stakeholder aziendali e tecnici ai problemi aziendali, chiamati domini, utilizzando un linguaggio onnipresente. L'architettura esagonale è un fattore abilitante tecnico di questo approccio in un dominio molto specifico, chiamato contesto limitato. Un contesto limitato è una sottoarea del problema aziendale altamente coesiva e poco connessa. Si consiglia di adottare un'architettura esagonale per tutti i progetti software aziendali indipendentemente dalla loro complessità.

L'architettura esagonale incoraggia il team di progettazione a risolvere innanzitutto il problema aziendale, mentre l'architettura classica a più livelli sposta l'attenzione della progettazione dal dominio alla risoluzione innanzitutto dei problemi tecnici. Inoltre, se il software segue un'architettura esagonale, è più facile adottare un [approccio di sviluppo basato sui test, che riduce il ciclo](#) di feedback necessario agli sviluppatori per testare i requisiti aziendali. Infine, l'utilizzo di [comandi e gestori di comandi](#) è un modo per applicare la responsabilità unica e i principi aperti e chiusi di

SOLID. L'adesione a questi principi produce una base di codice che gli sviluppatori e gli architetti che lavorano al progetto possono facilmente navigare e comprendere, e riduce il rischio di introdurre modifiche sostanziali alle funzionalità esistenti.

Questa guida è destinata agli architetti e agli sviluppatori di software interessati a comprendere i vantaggi dell'adozione dell'architettura esagonale e del DDD per i loro progetti di sviluppo software. Include un esempio di progettazione di un'infrastruttura per l'applicazione AWS che supporti l'architettura esagonale. Per un esempio di implementazione, consulta [Strutturare un progetto Python in architettura esagonale utilizzando](#) il sito Web AWS Prescriptive AWS Lambda Guidance.

Panoramica

Progettazione basata sul dominio (DDD)

Nella [progettazione basata sul dominio \(DDD\)](#), un dominio è il nucleo del sistema software. Il modello di dominio viene definito prima di sviluppare qualsiasi altro modulo e non dipende da altri moduli di basso livello. Invece, moduli come i database, il livello di presentazione e i moduli esterni dipendono APIs tutti dal dominio.

In DDD, gli architetti scompongono la soluzione in contesti limitati utilizzando la scomposizione basata sulla logica aziendale anziché la scomposizione tecnica. I vantaggi di questo approccio sono descritti nella sezione. [Obiettivi aziendali specifici](#)

DDD è più facile da implementare quando i team utilizzano un'architettura esagonale. Nell'architettura esagonale, il core dell'applicazione è il centro dell'applicazione. È disaccoppiato dagli altri moduli tramite porte e adattatori e non dipende da altri moduli. Ciò si allinea perfettamente con la DDD, in cui il dominio è il fulcro dell'applicazione che risolve un problema aziendale. Questa guida propone un approccio in cui si modella il nucleo dell'architettura esagonale come modello di dominio di un contesto limitato. La sezione successiva descrive l'architettura esagonale in modo più dettagliato.

Questa guida non copre tutti gli aspetti del DDD, che è un argomento molto ampio. Per una migliore comprensione, puoi consultare le risorse DDD elencate sul sito web di [Domain Language](#).

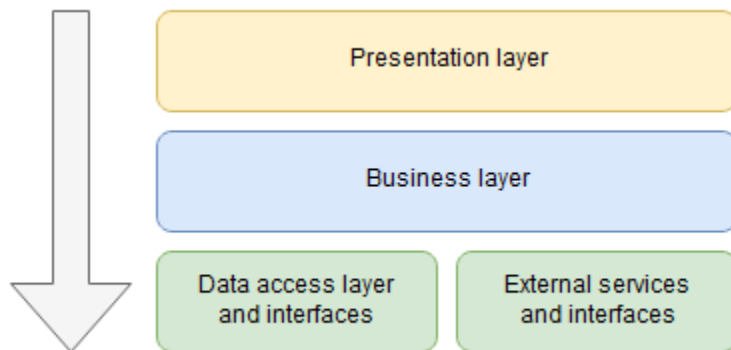
Architettura esagonale

L'architettura esagonale, nota anche come porte e adattatori o architettura a cipolla, è un principio di gestione dell'inversione delle dipendenze nei progetti software. L'architettura esagonale promuove una forte attenzione alla logica aziendale del dominio principale durante lo sviluppo del software e considera i punti di integrazione esterni come secondari. L'architettura esagonale aiuta gli ingegneri del software ad adottare buone pratiche come lo sviluppo basato sui test (TDD), che a sua volta promuove l'[evoluzione dell'architettura](#) e aiuta a gestire domini complessi a lungo termine.

Confrontiamo l'architettura esagonale con l'architettura classica a più livelli, che è la scelta più popolare per la modellazione di progetti software strutturati. Esistono differenze sottili ma potenti tra i due approcci.

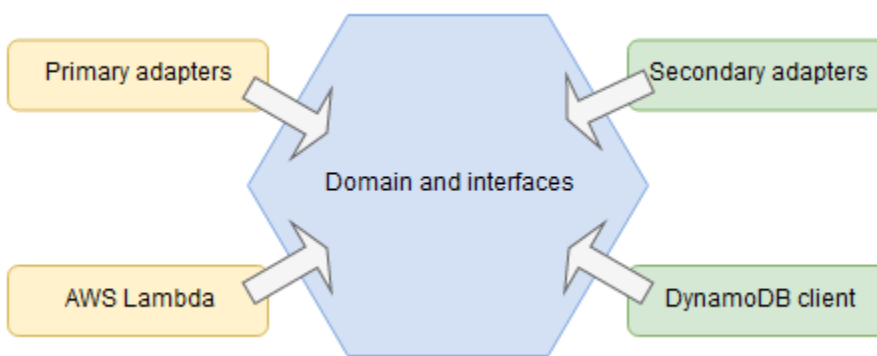
Nell'architettura a più livelli, i progetti software sono strutturati in livelli, che rappresentano aspetti generali come la logica aziendale o la logica di presentazione. Questa architettura utilizza una

gerarchia di dipendenze, in cui i livelli superiori dipendono dai livelli sottostanti, ma non viceversa. Nel diagramma seguente, il livello di presentazione è responsabile delle interazioni con l'utente, quindi include l'interfaccia utente APIs, le interfacce a riga di comando e componenti simili. Il livello di presentazione dipende dal livello aziendale, che implementa la logica di dominio. Il livello aziendale, a sua volta, dipende dal livello di accesso ai dati e da più servizi esterni.



Il principale svantaggio di questa configurazione è la struttura delle dipendenze. Ad esempio, se il modello di memorizzazione dei dati nel database cambia, ciò influisce sull'interfaccia di accesso ai dati. Qualsiasi modifica al modello di dati influisce anche sul livello aziendale, che dipende dall'interfaccia di accesso ai dati. Di conseguenza, gli ingegneri del software non possono apportare modifiche all'infrastruttura senza influire sulla logica del dominio. Ciò, a sua volta, aumenta la probabilità di bug di regressione.

L'architettura esagonale definisce le relazioni di dipendenza in un modo diverso, come illustrato nel diagramma seguente. Concentra il processo decisionale sulla logica aziendale del dominio, che definisce tutte le interfacce. I componenti esterni interagiscono con la logica aziendale tramite interfacce chiamate porte. Le porte sono astrazioni che definiscono le interazioni del dominio con il mondo esterno. Ogni componente dell'infrastruttura deve implementare tali porte, in modo che le modifiche apportate a tali componenti non influiscano più sulla logica del dominio principale.



I componenti circostanti sono chiamati adattatori. Un adattatore è un proxy tra il mondo esterno e il mondo interno e implementa una porta definita nel dominio. Gli adattatori possono essere classificati in due gruppi: primari e secondari. Gli adattatori primari sono i punti di accesso al componente software. Consentono ad attori, utenti e servizi esterni di interagire con la logica di base. AWS Lambda è un buon esempio di adattatore primario. Si integra con più AWS servizi in grado di richiamare le funzioni Lambda come punti di ingresso. Gli adattatori secondari sono wrapper di librerie di servizi esterni che gestiscono le comunicazioni con il mondo esterno. Un buon esempio di adattatore secondario è un client Amazon DynamoDB per l'accesso ai dati.

Obiettivi aziendali specifici

L'architettura esagonale discussa in questa guida ti aiuta a raggiungere i seguenti obiettivi:

- [Riduci il time-to-market migliorando il ciclo di sviluppo](#)
- [Migliora la qualità del software](#)
- [Adattati più facilmente ai cambiamenti](#)

Questi processi sono descritti in dettaglio nelle sezioni seguenti.

Migliorare il ciclo di sviluppo

Lo sviluppo di software per il cloud introduce nuove sfide per gli ingegneri del software, poiché è molto difficile replicare l'ambiente di runtime localmente sulla macchina di sviluppo. Un modo semplice per convalidare il software consiste nel distribuirlo nel cloud e testarlo lì. Tuttavia, questo approccio comporta un lungo ciclo di feedback, soprattutto quando l'architettura software contiene più implementazioni serverless. Il miglioramento di questo ciclo di feedback riduce i tempi di sviluppo delle funzionalità, il che riduce significativamente il time-to-market.

Test nel cloud

Il test diretto nel cloud è l'unico modo per assicurarsi che i componenti architetturici, come i gateway in Amazon API Gateway, AWS Lambda le funzioni, le tabelle Amazon DynamoDB e le autorizzazioni (IAM) AWS Identity and Access Management, siano configurati correttamente. Potrebbe anche essere l'unico modo affidabile per testare le integrazioni dei componenti. Sebbene alcuni AWS servizi (come [DynamoDB](#)) possano essere distribuiti localmente, la maggior parte di essi non può essere replicata in una configurazione locale. Allo stesso tempo, strumenti di terze parti come [Moto](#) e [LocalStack](#) quei AWS servizi fittizi a scopo di test potrebbero non rispecchiare fedelmente i contratti con le API di servizio reali o il numero di funzionalità potrebbe essere limitato.

Tuttavia, la parte più complessa del software aziendale risiede nella logica aziendale, non nell'architettura cloud. L'architettura cambia meno spesso rispetto al dominio, che deve soddisfare i nuovi requisiti aziendali. Pertanto, testare la logica aziendale nel cloud diventa un processo intenso di modifica del codice, avvio di un'implementazione, attesa che l'ambiente sia pronto e convalida della modifica. Se un'implementazione richiede solo 5 minuti, apportare e testare 10 modifiche alla logica aziendale richiederà un'ora o più. Se la logica aziendale è più complessa, i test potrebbero richiedere giorni di attesa solo per il completamento delle implementazioni. Se nel team sono presenti più funzionalità e più tecnici, il periodo prolungato diventa subito evidente per l'azienda.

Test in locale

Un'architettura esagonale aiuta gli sviluppatori a concentrarsi sul dominio anziché sugli aspetti tecnici dell'infrastruttura. Questo approccio utilizza test locali (gli strumenti di unit testing nel framework di sviluppo prescelto) per coprire i requisiti logici di dominio. Non dovrai perdere tempo a risolvere problemi di integrazione tecnica o distribuire il software nel cloud per testare la logica aziendale. È possibile eseguire test unitari localmente e ridurre il ciclo di feedback da minuti a secondi. Se

un'implementazione richiede 5 minuti ma gli unit test vengono completati in 5 secondi, si tratta di una riduzione significativa del tempo necessario per rilevare gli errori. La [Testare innanzitutto la logica aziendale](#) sezione successiva di questa guida tratta questo approccio in modo più dettagliato.

Parallelizzazione dello sviluppo

L'approccio all'architettura esagonale consente ai team di sviluppo di parallelizzare le attività di sviluppo. Gli sviluppatori possono progettare e implementare diversi componenti del servizio individualmente. Questa parallelizzazione è possibile attraverso l'isolamento di ogni componente e le interfacce definite tra ciascun componente.

Tempi di commercializzazione del prodotto

I test delle unità locali migliorano il ciclo di feedback sullo sviluppo e riducono i tempi di commercializzazione di nuovi prodotti o funzionalità, soprattutto quando questi contengono una logica aziendale complessa, come spiegato in precedenza. Inoltre, una maggiore copertura del codice mediante i test unitari riduce significativamente il rischio di introdurre bug di regressione quando si aggiorna o si rifattorizza la base di codice. La copertura degli unit test consente inoltre di rifattorizzare continuamente la base di codice per mantenerla ben organizzata, il che accelera il processo di onboarding per i nuovi ingegneri. Questo è discusso ulteriormente nella sezione. [Qualità fin dal design](#) Infine, se la logica aziendale è ben isolata e testata, consente agli sviluppatori di adattarsi rapidamente ai mutevoli requisiti funzionali e non funzionali. Questo è spiegato ulteriormente nella sezione. [Adattarsi al cambiamento](#)

Qualità fin dal design

L'adozione di un'architettura esagonale aiuta a promuovere la qualità della base di codice sin dall'inizio del progetto. È importante creare un processo che consenta di soddisfare i requisiti di qualità previsti sin dall'inizio, senza rallentare il processo di sviluppo.

Modifiche localizzate e maggiore leggibilità

L'utilizzo dell'approccio dell'architettura esagonale consente agli sviluppatori di modificare il codice in una classe o componente senza influire su altre classi o componenti. Questo design promuove la coesione dei componenti sviluppati. Disaccoppiando il dominio dagli adattatori e utilizzando interfacce note, è possibile aumentare la leggibilità del codice. Diventa più facile identificare problemi e casi isolati.

Questo approccio facilita anche la revisione del codice durante lo sviluppo e limita l'introduzione di modifiche o debiti tecnici non rilevati.

Testare innanzitutto la logica aziendale

I test locali possono essere eseguiti introducendo end-to-end, integrando e testando le unità nel progetto. End-to-end test coprono l'intero ciclo di vita delle richieste in entrata. In genere richiamano un punto di ingresso dell'applicazione e verificano se ha soddisfatto i requisiti aziendali. Ogni progetto software dovrebbe avere almeno uno scenario di test che utilizzi input noti e produca gli output previsti. Tuttavia, l'aggiunta di altri scenari specifici può diventare complessa, poiché ogni test deve essere configurato per inviare una richiesta tramite un punto di ingresso (ad esempio, tramite un'API REST o delle code), passare attraverso tutti i punti di integrazione richiesti dall'azione aziendale e quindi affermare il risultato. La configurazione dell'ambiente per lo scenario di test e l'affermazione dei risultati possono richiedere molto tempo agli sviluppatori.

Nell'architettura esagonale, si verifica la logica aziendale in modo isolato e si utilizzano i test di integrazione per testare gli adattatori secondari. È possibile utilizzare adattatori fittizi o falsi nei test di logica aziendale. Puoi anche combinare i test per i casi d'uso aziendale con i test unitari per il tuo modello di dominio per mantenere una copertura elevata con un accoppiamento basso. Come buona prassi, i test di integrazione non dovrebbero convalidare la logica aziendale. Dovrebbero invece verificare che l'adattatore secondario chiami correttamente i servizi esterni.

Idealmente, è possibile utilizzare lo sviluppo basato su test (TDD) e iniziare a definire entità di dominio o casi d'uso aziendali con test appropriati all'inizio dello sviluppo. Scrivere innanzitutto i

test consente di creare implementazioni fittizie delle interfacce richieste dal dominio. Quando i test hanno esito positivo e le regole logiche del dominio sono soddisfatte, è possibile implementare gli adattatori effettivi e distribuire il software nell'ambiente di test. A questo punto, l'implementazione della logica di dominio potrebbe non essere ideale. È quindi possibile lavorare sul refactoring dell'architettura esistente per farla evolvere introducendo modelli di progettazione o riorganizzando il codice in generale. Utilizzando questo approccio, è possibile evitare di introdurre bug di regressione e migliorare l'architettura man mano che il progetto cresce. Combinando questo approccio con i test automatici eseguiti nel processo di integrazione continua, è possibile ridurre il numero di potenziali bug prima che entrino in produzione.

Se utilizzi implementazioni serverless, puoi fornire rapidamente un'istanza dell'applicazione nel tuo AWS account per l'integrazione e il test manuali. end-to-end Dopo questi passaggi di implementazione, ti consigliamo di automatizzare i test con ogni nuova modifica inserita nel repository.

Manutenibilità

La manutenibilità si riferisce al funzionamento e al monitoraggio di un'applicazione per garantire che soddisfi tutti i requisiti e ridurre al minimo la probabilità di un guasto del sistema. Per rendere il sistema operativo, è necessario adattarlo al traffico o ai requisiti operativi futuri. È inoltre necessario assicurarsi che sia disponibile e facile da implementare con un impatto minimo o nullo sui client.

Per comprendere lo stato attuale e storico del sistema, è necessario renderlo osservabile. È possibile farlo fornendo metriche, registri e tracce specifici che gli operatori possono utilizzare per garantire che il sistema funzioni come previsto e per tenere traccia dei bug. Questi meccanismi dovrebbero inoltre consentire agli operatori di condurre l'analisi delle cause principali senza dover accedere alla macchina e leggere il codice.

Un'architettura esagonale mira ad aumentare la manutenibilità delle applicazioni Web in modo che il codice richieda complessivamente meno lavoro. Separando i moduli, localizzando le modifiche e separando la logica di business delle applicazioni dall'implementazione degli adattatori, è possibile produrre metriche e log che aiutano gli operatori a comprendere a fondo il sistema e a comprendere la portata delle modifiche specifiche apportate agli adattatori primari o secondari.

Adattarsi al cambiamento

I sistemi software tendono a complicarsi. Uno dei motivi potrebbe essere la frequente modifica dei requisiti aziendali e il poco tempo a disposizione per adattare di conseguenza l'architettura software. Un altro motivo potrebbe essere l'insufficiente investimento nella configurazione dell'architettura software all'inizio del progetto, in modo da adattarla ai cambiamenti frequenti. Qualunque sia la ragione, un sistema software potrebbe complicarsi al punto da rendere quasi impossibile apportare modifiche. Pertanto, è importante creare un'architettura software gestibile sin dall'inizio del progetto. Una buona architettura software può adattarsi facilmente alle modifiche.

Questa sezione spiega come progettare applicazioni gestibili utilizzando un'architettura esagonale che si adatta facilmente a requisiti non funzionali o aziendali.

Adattamento a nuovi requisiti non funzionali utilizzando porte e adattatori

In quanto elemento centrale dell'applicazione, il modello di dominio definisce le azioni richieste dal mondo esterno per soddisfare i requisiti aziendali. Queste azioni sono definite tramite astrazioni, chiamate porte. Queste porte sono implementate da adattatori separati. Ogni adattatore è responsabile dell'interazione con un altro sistema. Ad esempio, potresti avere un adattatore per l'archivio del database e un altro adattatore per interagire con un'API di terze parti. Il dominio non è a conoscenza dell'implementazione dell'adattatore, quindi è facile sostituire un adattatore con un altro. Ad esempio, l'applicazione potrebbe passare da un database SQL a un database NoSQL. In questo caso, è necessario sviluppare un nuovo adattatore per implementare le porte definite dal modello di dominio. Il dominio non ha dipendenze dal repository del database e utilizza le astrazioni per interagire, quindi non sarebbe necessario modificare nulla nel modello di dominio. Pertanto, l'architettura esagonale si adatta facilmente ai requisiti non funzionali.

Adattamento ai nuovi requisiti aziendali utilizzando comandi e gestori di comandi

Nell'architettura classica a più livelli, il dominio dipende dal livello di persistenza. Se si desidera modificare il dominio, è necessario modificare anche il livello di persistenza. In confronto, nell'architettura esagonale, il dominio non dipende da altri moduli del software. Il dominio è il cuore dell'applicazione e tutti gli altri moduli (porte e adattatori) dipendono dal modello di dominio. Il dominio utilizza il [principio di inversione delle dipendenze](#) per comunicare con il mondo esterno attraverso

le porte. Il vantaggio dell'inversione delle dipendenze è che puoi modificare liberamente il modello di dominio senza aver paura di violare altre parti del codice. Poiché il modello di dominio riflette il problema aziendale che si sta cercando di risolvere, aggiornare il modello di dominio per adattarlo alle mutevoli esigenze aziendali non è un problema.

Quando si sviluppa software, la separazione delle preoccupazioni è un principio importante da seguire. Per ottenere questa separazione, è possibile utilizzare uno [schema di comandi leggermente modificato](#). Si tratta di un modello di progettazione comportamentale in cui tutte le informazioni necessarie per completare un'operazione sono incapsulate in un oggetto di comando. Queste operazioni vengono quindi elaborate dai gestori di comandi. I gestori di comandi sono metodi che ricevono un comando, modificano lo stato del dominio e quindi restituiscono una risposta al chiamante. È possibile utilizzare client diversi, ad esempio code sincrone APIs o asincrone, per eseguire i comandi. È consigliabile utilizzare comandi e gestori di comandi per ogni operazione sul dominio. Seguendo questo approccio, è possibile aggiungere nuove funzionalità introducendo nuovi comandi e gestori di comandi, senza modificare la logica di business esistente. Pertanto, l'utilizzo di un modello di comandi semplifica l'adattamento ai nuovi requisiti aziendali.

Disaccoppiamento dei componenti utilizzando la facciata di servizio o il pattern CQRS

Nell'architettura esagonale, gli adattatori primari hanno la responsabilità di accoppiare liberamente le richieste di lettura e scrittura in entrata dai client al dominio. Esistono due modi per ottenere questo accoppiamento libero: utilizzando un modello di facciata di servizio o utilizzando il modello CQRS (Command Query Responsibility Segregation).

Il [service façade pattern](#) fornisce un'interfaccia frontale per servire client come il livello di presentazione o un microservizio. Una facciata di servizio offre ai clienti diverse operazioni di lettura e scrittura. È responsabile del trasferimento delle richieste in entrata al dominio e della mappatura della risposta ricevuta dal dominio ai client. L'utilizzo di una facciata di servizio è semplice per i microservizi che hanno un'unica responsabilità con diverse operazioni. Tuttavia, quando si utilizza la facciata di servizio, è più difficile seguire una [singola](#) responsabilità e principi aperti e chiusi. Il principio di responsabilità unica stabilisce che ogni modulo dovrebbe avere la responsabilità solo su una singola funzionalità del software. Il principio aperto-chiuso afferma che il codice deve essere aperto all'estensione e chiuso alla modifica. Man mano che la facciata dei servizi si estende, tutte le operazioni vengono raccolte in un'unica interfaccia, in essa vengono incorporate più dipendenze e sempre più sviluppatori iniziano a modificare la stessa facciata. Pertanto, consigliamo di utilizzare una facciata di servizio solo se è chiaro che il servizio non si estenderebbe molto durante lo sviluppo.

Un altro modo per implementare gli adattatori primari nell'architettura esagonale consiste nell'utilizzare il [pattern CQRS](#), che separa le operazioni di lettura e scrittura utilizzando query e comandi. Come spiegato in precedenza, i comandi sono oggetti che contengono tutte le informazioni necessarie per modificare lo stato del dominio. I comandi vengono eseguiti tramite i metodi del gestore dei comandi. Le interrogazioni, invece, non alterano lo stato del sistema. Il loro unico scopo è restituire i dati ai clienti. Nel modello CQRS, i comandi e le interrogazioni sono implementati in moduli separati. Ciò è particolarmente vantaggioso per i progetti che seguono un'[architettura basata sugli eventi](#), perché un comando può essere implementato come evento elaborato in modo asincrono, mentre una query può essere eseguita in modo sincrono utilizzando un'API. Una query può anche utilizzare un database diverso ottimizzato per essa. Lo svantaggio del pattern CQRS è che l'implementazione richiede più tempo rispetto a una facciata di servizio. Si consiglia di utilizzare il modello CQRS per i progetti che si prevede di scalare e mantenere a lungo termine. I comandi e le interrogazioni forniscono un meccanismo efficace per applicare il principio di responsabilità unica e sviluppare software liberamente accoppiati, specialmente in progetti su larga scala.

CQRS offre grandi vantaggi a lungo termine, ma richiede un investimento iniziale. Per questo motivo, ti consigliamo di valutare attentamente il tuo progetto prima di decidere di utilizzare il pattern CQRS. Tuttavia, è possibile strutturare l'applicazione utilizzando comandi e gestori di comandi fin dall'inizio senza separare le operazioni di lettura/scrittura. Questo vi aiuterà a rifattorizzare facilmente il vostro progetto per CQRS se deciderete di adottare tale approccio in un secondo momento.

Scalabilità organizzativa

Una combinazione di architettura esagonale, progettazione basata sul dominio e (facoltativamente) CQRS consente all'organizzazione di scalare rapidamente il prodotto. Secondo la [legge di Conway](#), le architetture software tendono a evolversi per riflettere le strutture di comunicazione di un'azienda. Questa osservazione ha storicamente avuto connotazioni negative, perché le grandi organizzazioni spesso strutturano i propri team sulla base di competenze tecniche come database, service bus aziendali e così via. Il problema di questo approccio è che lo sviluppo di prodotti e funzionalità implica sempre preoccupazioni trasversali, come la sicurezza e la scalabilità, che richiedono una comunicazione costante tra i team. La strutturazione dei team in base alle caratteristiche tecniche crea inutili silos all'interno dell'organizzazione, che si traducono in comunicazioni scadenti, mancanza di titolarità e perdita di vista del quadro generale. Alla fine, questi problemi organizzativi si riflettono nell'architettura del software.

L'[Inverse Conway Maneuver](#), d'altra parte, definisce la struttura organizzativa basata su domini che promuovono l'architettura software. [Ad esempio, ai team interfunzionali viene affidata la](#)

responsabilità di una serie specifica di contesti limitati, che vengono identificati utilizzando DDD e event storming.

Questi contesti limitati potrebbero riflettere caratteristiche molto specifiche del prodotto. Ad esempio, il team addetto all'account potrebbe essere responsabile del contesto di pagamento. Ogni nuova funzionalità viene assegnata a un nuovo team con responsabilità altamente coese e vagamente collegate, in modo che possa concentrarsi solo sulla fornitura di tale funzionalità e ridurre i tempi di commercializzazione. I team possono essere ridimensionati in base alla complessità delle funzionalità, in modo da assegnare funzionalità complesse a più ingegneri.

Best practice

Modella il dominio aziendale

Passa dal dominio aziendale alla progettazione del software per assicurarti che il software che stai scrivendo soddisfi le esigenze aziendali.

Utilizza metodologie di progettazione basata sul dominio (DDD) come l'[event storming](#) per modellare il dominio aziendale. Event storming ha un formato di workshop flessibile. Durante il workshop, esperti di dominio e software esplorano in modo collaborativo la complessità del settore aziendale. Gli esperti di software utilizzano i risultati del workshop per avviare il processo di progettazione e sviluppo dei componenti software.

Scrivi ed esegui test fin dall'inizio

Utilizzate lo sviluppo basato sui test (TDD) per verificare la correttezza del software che state sviluppando. TDD funziona meglio a livello di test unitario. Lo sviluppatore progetta un componente software scrivendo prima un test, che richiama quel componente. Tale componente non ha alcuna implementazione all'inizio, quindi il test fallisce. Come passaggio successivo, lo sviluppatore implementa la funzionalità del componente, utilizzando dispositivi di test con oggetti fittizi per simulare il comportamento di dipendenze o porte esterne. Quando il test ha esito positivo, lo sviluppatore può continuare implementando adattatori reali. Questo approccio migliora la qualità del software e produce codice più leggibile, perché gli sviluppatori comprendono come gli utenti utilizzerebbero i componenti. L'architettura esagonale supporta la metodologia TDD separando il core dell'applicazione. Gli sviluppatori scrivono test unitari incentrati sul comportamento principale del dominio. Non devono scrivere adattatori complessi per eseguire i test; possono invece utilizzare semplici oggetti e dispositivi fittizi.

Utilizzate lo sviluppo basato sul comportamento (BDD) per garantire end-to-end l'accettazione a livello di funzionalità. In BDD, gli sviluppatori definiscono gli scenari per le funzionalità e li verificano con gli stakeholder aziendali. I test BDD utilizzano quanto più linguaggio naturale possibile per raggiungere questo obiettivo. L'architettura esagonale supporta la metodologia BDD con il suo concetto di adattatori primari e secondari. Gli sviluppatori possono creare adattatori primari e secondari che possono essere eseguiti localmente senza chiamare servizi esterni. Configurano la suite di test BDD per utilizzare l'adattatore primario locale per eseguire l'applicazione.

Esegui automaticamente ogni test nella pipeline di integrazione continua per valutare costantemente la qualità del sistema.

Definisci il comportamento del dominio

Scomponi il dominio in entità, oggetti di valore e aggregati (leggi come [implementare la progettazione basata sul dominio](#)) e [definisci](#) il loro comportamento. Implementa il comportamento del dominio in modo che i test scritti all'inizio del progetto abbiano esito positivo. Definisci i comandi che richiamano il comportamento degli oggetti di dominio. Definisci gli eventi che gli oggetti di dominio emettono dopo il completamento di un comportamento.

Definite le interfacce che gli adattatori possono utilizzare per interagire con il dominio.

Automatizza i test e l'implementazione

Dopo una prima dimostrazione concettuale, ti consigliamo di dedicare tempo all'implementazione DevOps delle pratiche. Ad esempio, le pipeline di integrazione e distribuzione continua (CI/CD) e gli ambienti di test dinamici aiutano a mantenere la qualità del codice ed evitare errori durante la distribuzione.

- Eseguite i test unitari all'interno del processo CI e testate il codice prima che venga unito.
- Create un processo CD per distribuire l'applicazione in un ambiente di sviluppo/test statico o in ambienti creati dinamicamente che supportano l'integrazione e il test automatici. end-to-end
- Automatizza il processo di distribuzione per ambienti dedicati.

Scala il tuo prodotto utilizzando microservizi e CQRS

Se il tuo prodotto ha successo, scalalo scomponendo il progetto software in microservizi. Utilizzate la portabilità offerta dall'architettura esagonale per migliorare le prestazioni. Suddividi i servizi di interrogazione e i gestori di comandi in sincroni e asincroni separati. APIs Prendi in considerazione l'adozione del modello CQRS (Command Query Responsibility Segregation) e dell'architettura basata sugli eventi.

Se ricevi molte nuove richieste di funzionalità, valuta la possibilità di ridimensionare la tua organizzazione sulla base di modelli DDD. Strutturate i vostri team in modo che possiedano una

o più funzionalità come contesti delimitati, come discusso in precedenza nella sezione. [Scalabilità organizzativa](#) Questi team possono quindi implementare la logica aziendale utilizzando l'architettura esagonale.

Progetta una struttura di progetto che si adatti ai concetti di architettura esagonale

L'infrastruttura come codice (IaC) è una pratica ampiamente adottata nello sviluppo del cloud. Consente di definire e gestire le risorse dell'infrastruttura (come reti, sistemi di bilanciamento del carico, macchine virtuali e gateway) come codice sorgente. In questo modo, è possibile tenere traccia di tutte le modifiche all'architettura utilizzando un sistema di controllo della versione. Inoltre, è possibile creare e spostare facilmente l'infrastruttura a scopo di test. Ti consigliamo di conservare il codice dell'applicazione e il codice dell'infrastruttura nello stesso repository quando sviluppi le tue applicazioni cloud. Questo approccio semplifica la manutenzione dell'infrastruttura per l'applicazione.

Si consiglia di dividere l'applicazione in tre cartelle o progetti che corrispondono ai concetti di architettura esagonale: `entrypoints` (adattatori primari), (dominio e interfacce) e `domain` (adattatori secondari). `adapters`

La seguente struttura di progetto fornisce un esempio di questo approccio per la progettazione di un'API su AWS. Il progetto mantiene il codice dell'applicazione (`app`) e il codice dell'infrastruttura (`infra`) nello stesso repository, come consigliato in precedenza.

```
app/ # application code
|--- adapters/ # implementation of the ports defined in the domain
|   |--- tests/ # adapter unit tests
|--- entrypoints/ # primary adapters, entry points
|   |--- api/ # api entry point
|       |--- model/ # api model
|       |--- tests/ # end to end api tests
|--- domain/ # domain to implement business logic using hexagonal architecture
|   |--- command_handlers/ # handlers used to run commands on the domain
|   |--- commands/ # commands on the domain
|   |--- events/ # events emitted by the domain
|   |--- exceptions/ # exceptions defined on the domain
|   |--- model/ # domain model
|   |--- ports/ # abstractions used for external communication
|   |--- tests/ # domain tests
infra/ # infrastructure code
```

Come discusso in precedenza, il dominio è il cuore dell'applicazione e non dipende da nessun altro modulo. Si consiglia di strutturare la `domain` cartella in modo da includere le seguenti sottocartelle:

- `command_handlers` contiene i metodi o le classi che eseguono i comandi nel dominio.
- `commands` contiene gli oggetti di comando che definiscono le informazioni necessarie per eseguire un'operazione sul dominio.
- `events` contiene gli eventi che vengono emessi attraverso il dominio e quindi indirizzati ad altri microservizi.
- `exceptions` contiene gli errori noti definiti all'interno del dominio.
- `model` contiene le entità di dominio, gli oggetti di valore e i servizi di dominio.
- `ports` contiene le astrazioni attraverso le quali il dominio comunica con i database o altri componenti esterni. APIs
- `tests` contiene i metodi di test (come i test di logica aziendale) eseguiti sul dominio.

Gli adattatori principali sono i punti di ingresso all'applicazione, rappresentati dalla `entrypoints` cartella. Questo esempio utilizza la `api` cartella come adattatore principale. Questa cartella contiene un'API `model` che definisce l'interfaccia richiesta dall'adattatore principale per comunicare con i client. La `tests` cartella contiene end-to-end i test per l'API. Si tratta di test superficiali che confermano che i componenti dell'applicazione sono integrati e funzionano in armonia.

Gli adattatori secondari, rappresentati dalla `adapters` cartella, implementano le integrazioni esterne richieste dalle porte di dominio. Un archivio di database è un ottimo esempio di adattatore secondario. Quando il sistema di database cambia, puoi scrivere un nuovo adattatore utilizzando l'implementazione definita dal dominio. Non è necessario modificare il dominio o la logica aziendale. La `tests` sottocartella contiene test di integrazione esterni per ogni adattatore.

Esempi di infrastrutture su AWS

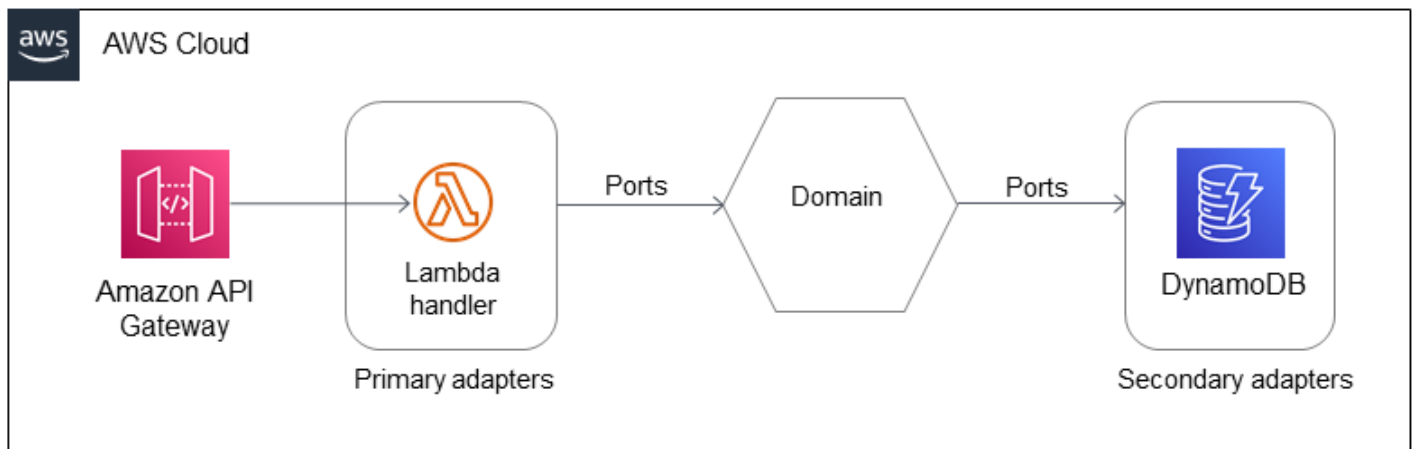
Questa sezione fornisce esempi per la progettazione di un'infrastruttura per l'applicazione da utilizzare per implementare un'architettura esagonale. AWS Ti consigliamo di iniziare con un'architettura semplice per creare un prodotto minimo valido (MVP). La maggior parte dei microservizi necessita di un unico punto di ingresso per gestire le richieste dei client, un livello di elaborazione per eseguire il codice e un livello di persistenza per archiviare i dati. I seguenti AWS servizi sono ottimi candidati per essere utilizzati come client, adattatori primari e adattatori secondari in architettura esagonale:

- Clienti: Amazon API Gateway, Amazon Simple Queue Service (Amazon SQS), Elastic Load Balancing, Amazon EventBridge
- Adattatori principali: Amazon Elastic Container Service (Amazon ECS) AWS Lambda, Amazon Elastic Kubernetes Service (Amazon EKS), Amazon Elastic Compute Cloud (Amazon) EC2
- Adattatori secondari: Amazon DynamoDB, Amazon Relational Database Service (Amazon RDS), Amazon Aurora, API Gateway, Amazon SQS, Elastic Load Balancing EventBridge, Amazon Simple Notification Service (Amazon SNS)

Le sezioni seguenti illustrano questi servizi nel contesto dell'architettura esagonale in modo più dettagliato.

Inizia in modo semplice

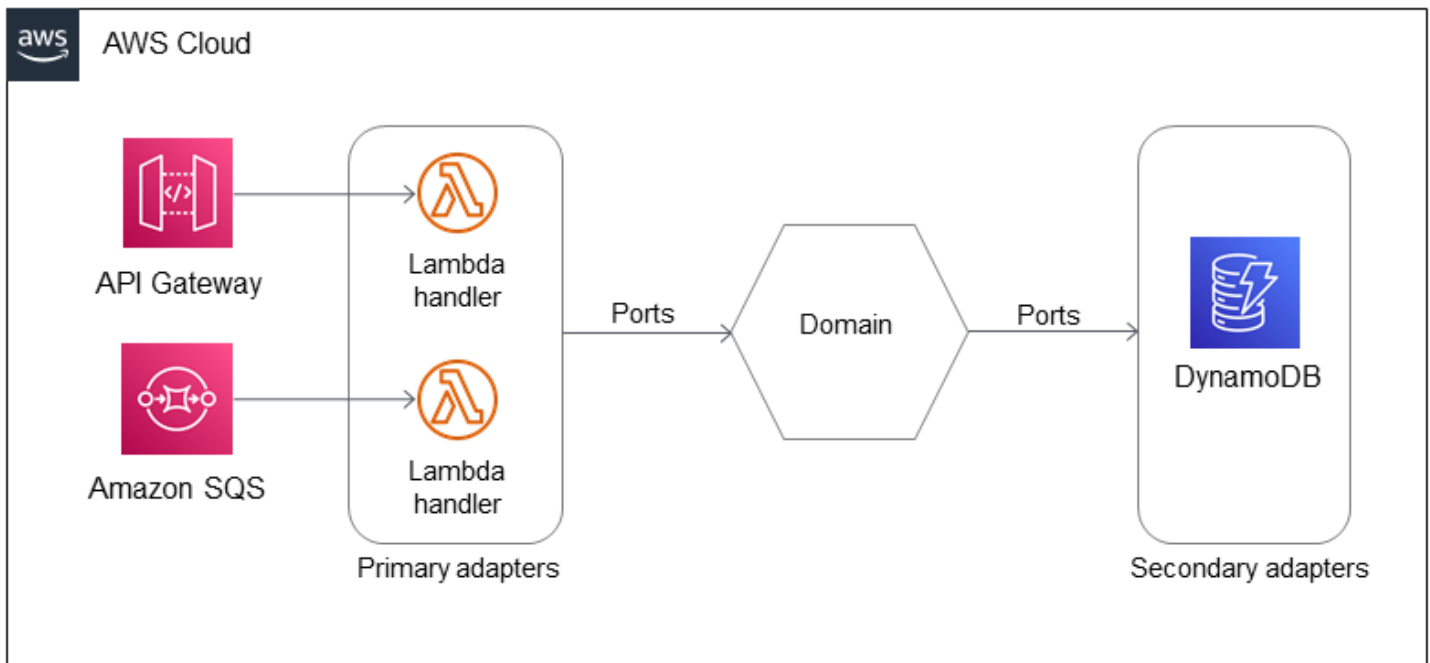
Si consiglia di iniziare in modo semplice quando si progetta un'applicazione utilizzando l'architettura esagonale. In questo esempio, API Gateway viene utilizzato come client (API REST), Lambda come adattatore principale (elaborazione) e DynamoDB viene utilizzato come adattatore secondario (persistenza). Il client gateway chiama il punto di ingresso, che, in questo caso, è un gestore Lambda.



Questa architettura è completamente serverless e offre all'architetto un buon punto di partenza. Si consiglia di utilizzare lo schema di comandi nel dominio perché semplifica la gestione del codice e si adatta ai nuovi requisiti aziendali e non funzionali. Questa architettura potrebbe essere sufficiente per creare microservizi semplici con poche operazioni.

Applica il pattern CQRS

Ti consigliamo di passare al pattern CQRS se il numero di operazioni sul dominio sta per aumentare. È possibile applicare il pattern CQRS come architettura completamente serverless AWS utilizzando l'esempio seguente.

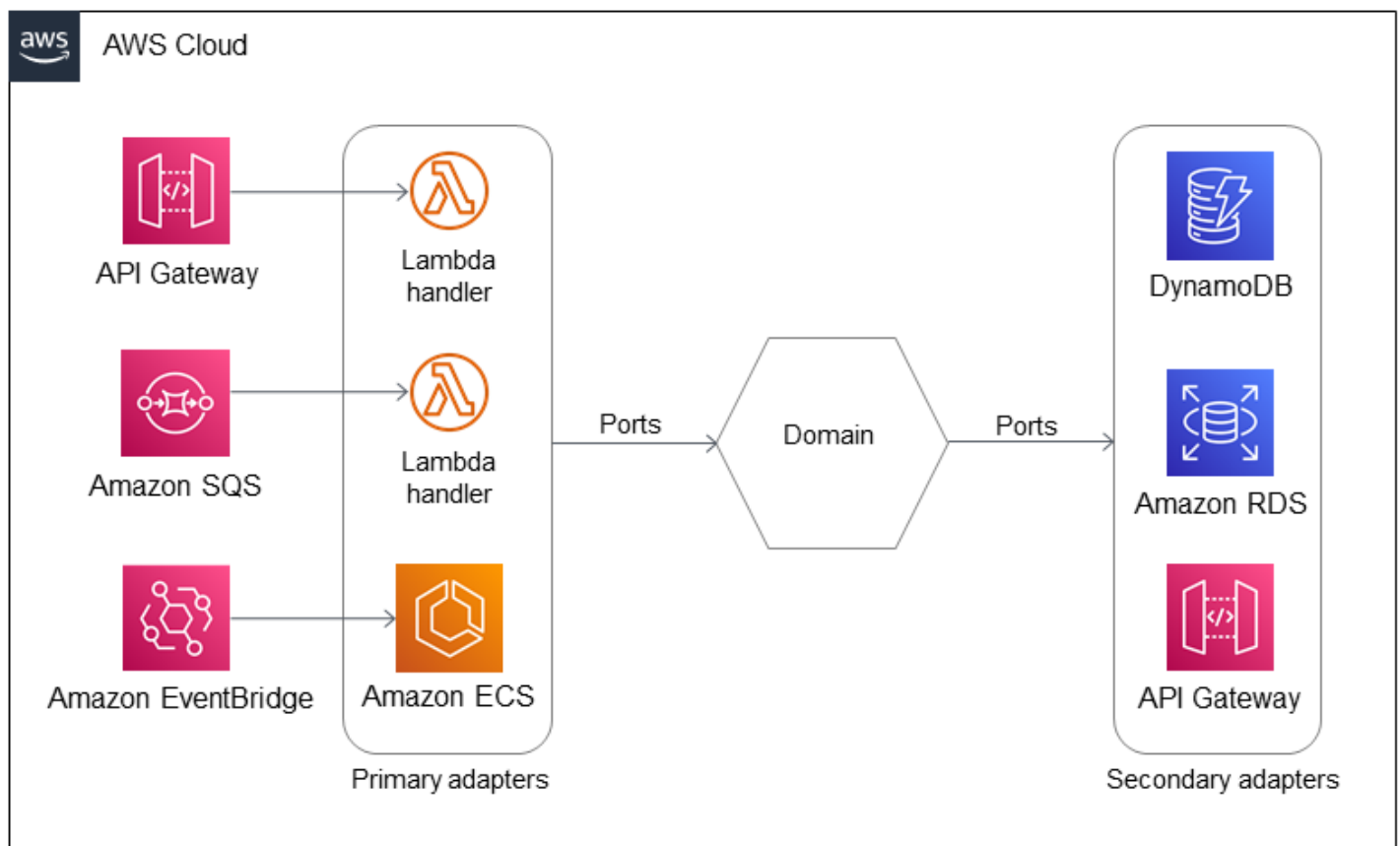


Questo esempio utilizza due gestori Lambda, uno per le query e uno per i comandi. Le query vengono eseguite in modo sincrono utilizzando un gateway API come client. I comandi vengono eseguiti in modo asincrono utilizzando Amazon SQS come client.

Questa architettura include più client (API Gateway e Amazon SQS) e più adattatori primari (Lambda), che vengono richiamati dai punti di ingresso corrispondenti (gestori Lambda). Tutti i componenti appartengono allo stesso contesto limitato, quindi si trovano all'interno dello stesso dominio.

Fai evolvere l'architettura aggiungendo contenitori, un database relazionale e un'API esterna

I contenitori sono una buona opzione per le attività di lunga durata. Potresti anche voler utilizzare un database relazionale se disponi di uno schema di dati predefinito e desideri sfruttare la potenza del linguaggio SQL. Inoltre, il dominio dovrebbe comunicare con l'esterno. APIs È possibile evolvere l'esempio di architettura per supportare questi requisiti, come illustrato nel diagramma seguente.

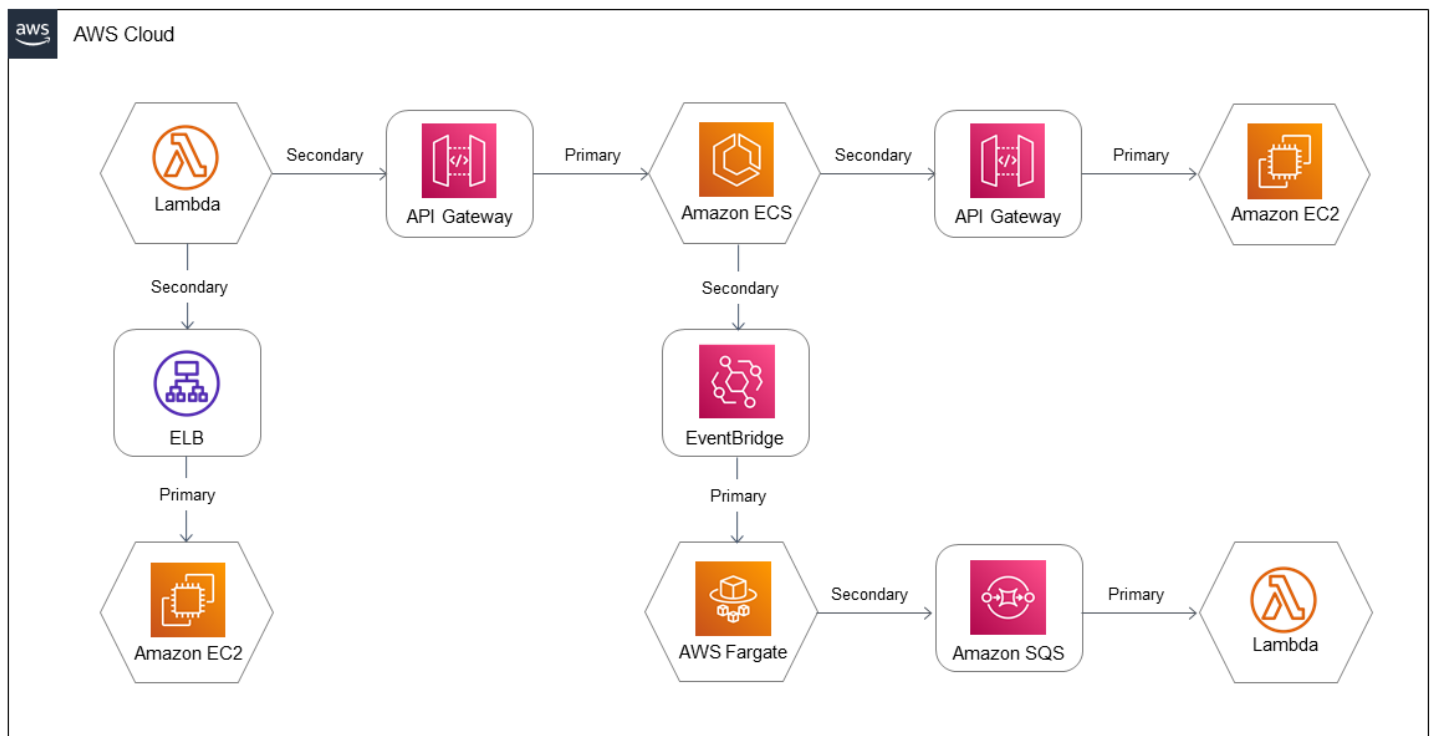


Questo esempio utilizza Amazon ECS come adattatore principale per l'avvio di attività di lunga durata nel dominio. Amazon EventBridge (client) avvia un'attività Amazon ECS (punto di ingresso) quando si verifica un evento specifico. L'architettura include Amazon RDS come altro adattatore secondario per l'archiviazione di dati relazionali. Aggiunge inoltre un altro gateway API come adattatore secondario per richiamare una chiamata API esterna. Di conseguenza, l'architettura utilizza più adattatori primari e secondari che si basano su diversi livelli di elaborazione sottostanti in un unico dominio aziendale.

Il dominio è sempre associato in modo lasco a tutti gli adattatori primari e secondari tramite astrazioni chiamate porte. Il dominio definisce ciò che richiede dal mondo esterno utilizzando le porte. Poiché è responsabilità dell'adattatore implementare la porta, il passaggio da un adattatore all'altro non influisce sul dominio. Ad esempio, puoi passare da Amazon DynamoDB ad Amazon RDS scrivendo un nuovo adattatore, senza influire sul dominio.

Aggiungi altri domini (rimpicciolisci)

L'architettura esagonale si allinea bene con i principi di un'architettura di microservizi. Gli esempi di architettura mostrati finora contenevano un singolo dominio (o contesto limitato). Le applicazioni includono in genere più domini, che devono comunicare tramite adattatori primari e secondari. Ogni dominio rappresenta un microservizio ed è associato in modo approssimativo ad altri domini.



In questa architettura, ogni dominio utilizza un set diverso di ambienti di calcolo. (Ogni dominio può anche avere più ambienti di calcolo, come nell'esempio precedente). Ogni dominio definisce le interfacce richieste per comunicare con altri domini tramite porte. Le porte vengono implementate utilizzando adattatori primari e secondari. In questo modo, il dominio non viene modificato in caso di modifica dell'adattatore. Inoltre, i domini sono disaccoppiati l'uno dall'altro.

Nell'esempio di architettura mostrato nel diagramma precedente, Lambda, Amazon EC2, Amazon ECS AWS Fargate e vengono utilizzati come adattatori principali. API Gateway, Elastic Load Balancing e Amazon SQS vengono utilizzati come adattatori secondari. EventBridge

Domande frequenti

Perché dovrei usare un'architettura esagonale?

L'architettura esagonale sposta l'attenzione degli sviluppatori sulla logica di dominio, semplifica l'automazione dei test e migliora la qualità e l'adattabilità del codice. Questi miglioramenti si traducono in un time-to-market più rapido e in una scalabilità tecnica e organizzativa più semplice.

Perché utilizzare la progettazione basata sul dominio?

La progettazione basata sul dominio (DDD) consente di creare componenti e costrutti software utilizzando un linguaggio comune tra gli stakeholder aziendali e gli ingegneri. La DDD aiuta a gestire la complessità del software ed è una strategia efficace per la manutenzione dei prodotti software a lungo termine.

Posso praticare lo sviluppo basato sui test senza un'architettura esagonale?

Sì. Lo sviluppo basato sui test (TDD) non si limita a modelli di progettazione software specifici. Tuttavia, l'architettura esagonale semplifica la pratica del TDD.

Posso scalare il mio prodotto senza un'architettura esagonale e una progettazione basata sul dominio?

Sì. La scalabilità tecnica e organizzativa del prodotto può essere ottenuta con la maggior parte dei modelli di progettazione. Tuttavia, l'architettura esagonale e il DDD facilitano la scalabilità e sono più efficaci per progetti di grandi dimensioni a lungo termine.

Quali tecnologie devo usare per implementare l'architettura esagonale?

L'architettura esagonale non si limita a uno stack tecnologico specifico. Ti consigliamo di scegliere una tecnologia che supporti l'inversione delle dipendenze e il test unitario.

Sto sviluppando un prodotto minimo valido. Ha senso dedicare del tempo a pensare all'architettura del software?

Sì. Ti consigliamo di utilizzare modelli di progettazione che ti sono familiari MVPs. Ti invitiamo a provare a praticare l'architettura esagonale finché i tuoi ingegneri non si sentiranno a proprio agio. La creazione di un'architettura esagonale per nuovi progetti non richiede un investimento di tempo significativamente maggiore rispetto all'avvio senza alcuna architettura.

Sto sviluppando un prodotto minimamente funzionante e non ho tempo per scrivere test.

Se il tuo MVP contiene una logica aziendale, ti consigliamo vivamente di scrivere test automatici al riguardo. Ciò ridurrà il ciclo di feedback e farà risparmiare tempo.

Quali modelli di progettazione aggiuntivi posso utilizzare con l'architettura esagonale?

Utilizzate il [pattern CQRS](#) per supportare il ridimensionamento dell'intero sistema. Usa il [modello di repository](#) per archiviare e ripristinare il tuo modello di dominio. Utilizza il modello Unit of Work per gestire le fasi del processo transazionale. Usa la composizione rispetto all'ereditarietà per modellare aggregati di dominio, entità e oggetti di valore. Non create gerarchie di oggetti complesse.

Passaggi successivi

- Acquisisci maggiore familiarità con i concetti di progettazione basati sul dominio leggendo i link raccolti nella sezione. [Resources](#)
- Se state implementando un nuovo progetto, utilizzate il [modello di struttura del progetto](#) fornito in questa guida e implementate alcune funzionalità.
- Se stai implementando un progetto esistente, identifica il codice che può essere suddiviso tra operazioni di sola lettura e di sola scrittura. Astrae il codice di sola lettura nei servizi di interrogazione e inserite il codice di sola scrittura nei gestori di comandi.
- Una volta stabilita la struttura del progetto di base, scrivete test unitari, stabilite l'integrazione continua (CI) con l'automazione dei test e seguite le pratiche di sviluppo basate sui test (TDD).

Resources

Riferimenti

- [Struttura un progetto Python in architettura esagonale usando AWS Lambda\(modello Prescriptive Guidance\)](#)AWS
- [Agile Teams](#) (sito web Scaled Agile Framework)
- [Architecture patterns with Python](#), di Harry Percival e Bob Gregory (O'Reilly Media, 31 marzo 2020), in particolare i seguenti capitoli:
 - [Comandi e gestore di comandi](#)
 - [Segregazione della responsabilità delle interrogazioni di comando \(CQRS\)](#)
 - [Modello di repository](#)
- [Event storming: l'approccio più intelligente alla collaborazione oltre i confini dei silo](#), di Alberto Brandolini (sito Event Storming)
- [Demystifying Conway's Law](#), di Sam Newman (sito web di Thoughtworks, 30 giugno 2014)
- [Sviluppo di un'architettura evolutiva con AWS Lambda](#), di James Beswick (Compute Blog, 8 luglio 2021)AWS
- [Domain Language: affrontare la complessità nel cuore del software](#) (sito web Domain Language)
- [Facade](#), tratto da Dive Into Design Patterns di Alexander Shvets (ebook, 5 dicembre 2018)
- [GivenWhenThen](#), di Martin Fowler (21 agosto 2013)
- [Implementazione della progettazione basata sul dominio](#), di Vaughn Vernon (Addison-Wesley Professional, febbraio 2013)
- [Inverse Conway Maneuver](#) (sito web Thoughtworks, 8 luglio 2014)
- [Modello del mese: Red Green Refactor](#) (sito web, 2 giugno 2017) DZone
- [Spiegazione dei principi di progettazione di SOLID: principio di inversione delle dipendenze con esempi di codice](#), di Thorben Janssen (sito web Stackify, 7 maggio 2018)
- [Principi SOLID: spiegazione ed esempi](#), di Simon Hoiberg (sito web ITNEXT, 1 gennaio 2019)
- [L'arte dello sviluppo agile: sviluppo basato sui test, di](#) James Shore e Shane Warden (O'Reilly Media, 25 marzo 2010)
- [I solidi principi della programmazione orientata agli oggetti spiegati in inglese semplice](#), da Yigit Kemal Erinc (post sulla programmazione orientata agli oggetti, 20 agosto 2020) freeCodeCamp
- [Cos'è un'architettura basata sugli eventi?](#) (sito web)AWS

AWS servizi

- [Gateway Amazon API](#)
- [Amazon Aurora](#)
- [Amazon DynamoDB](#)
- [Amazon Elastic Compute Cloud \(Amazon EC2\)](#)
- [Amazon Elastic Container Service \(Amazon ECS\)](#)
- [Amazon Elastic Kubernetes Service \(Amazon EKS\)](#)
- [Elastic Load Balancing](#)
- [Amazon EventBridge](#)
- [AWS Fargate](#)
- [AWS Lambda](#)
- [Amazon Relational Database Service \(Amazon RDS\)](#)
- [Amazon Simple Notification Service \(Amazon SNS\)](#)
- [Amazon Simple Queue Service \(Amazon SQS\)](#)

Altri strumenti

- [Moto](#)
- [LocalStack](#)

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 giugno 2022

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 PostgreSQL.
- **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 (Amazon RDS) 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 tuo database Oracle locale a Oracle su un'istanza EC2 in 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. Esegui la migrazione dei server da una piattaforma locale a un servizio cloud per la stessa piattaforma. Esempio: migra un'applicazione su Microsoft Hyper-V. 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](#).

ACIDO

Vedi [atomicità, consistenza, isolamento, 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 del 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 di aggregazione

Una funzione SQL 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 frequentemente 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 utilizzato 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, durabilità (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 su attributi (ABAC)

La pratica di creare autorizzazioni dettagliate basate su attributi utente, come reparto, ruolo professionale e nome del team. Per ulteriori informazioni, consulta [ABAC 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 Cloud Adoption Framework (CAF)AWS

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 CAF organizza 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 delle persone, la formazione e le comunicazioni per aiutare a preparare l'organizzazione all'adozione del cloud di successo. Per ulteriori informazioni, consulta il [sito web di AWS CAF](#) e il [white paper AWS CAF](#).

AWS Workload Qualification Framework (WQF)AWS

Uno strumento che valuta i carichi di lavoro di migrazione dei database, consiglia strategie di migrazione e fornisce stime del lavoro. AWS WQF è 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 non riusciti, chiamate API 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.

implementazione blu/verde

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 [Cloud Adoption AWS Framework](#).

implementazione canaria

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

CCoE

Vedi [Cloud Center of Excellence](#).

CDC

Vedi [Change Data Capture](#).

Change Data Capture (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 utilizzare CDC per vari scopi, ad esempio 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 gli [CCoE post](#) 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 zone, definizione di una CCo E, 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 più comuni includono GitHub o Bitbucket Cloud. Ogni versione del codice è denominata ramo. 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, Amazon SageMaker AI 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 di gestione della configurazione (CMDB)

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 di un CMDB nella fase di individuazione e analisi del portafoglio della 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, compilazione, test, gestione temporanea e produzione del processo di rilascio del software. CI/CD viene comunemente descritto come una pipeline. CI/CD può aiutarvi 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 architetturico 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 dello 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. DVSM estende 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 il modello del fico strangolatore (Strangler Fig), consulta la sezione [Modernizzazione incrementale dei servizi Web Microsoft ASP.NET \(ASMX\) legacy utilizzando container e il Gateway Amazon API](#).

DOTT.

Vedi [disaster recovery](#).

rilevamento della deriva

Tracciamento delle deviazioni da una configurazione di base. Ad esempio, è possibile AWS CloudFormation utilizzarlo per [rilevare deviazioni nelle risorse di sistema](#) oppure AWS Control Tower per [rilevare cambiamenti nella 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](#).

MODIFICA

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 aziendali 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) da condividere con altri utenti. Puoi 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 principali possono connettersi al servizio endpoint in privato creando endpoint VPC di interfaccia. Per ulteriori informazioni, consulta [Creazione di un servizio endpoint](#) nella documentazione di Amazon Virtual Private Cloud (Amazon VPC).

pianificazione delle risorse aziendali (ERP)

Un sistema che automatizza e gestisce i processi aziendali chiave (come contabilità, [MES](#) e 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 CI/CD pipeline, l'ambiente di produzione è l'ultimo ambiente di distribuzione.

- 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 della sicurezza AWS CAF 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 [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. L'EDA viene eseguita calcolando statistiche di riepilogo 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, consulta [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 a un [LLM](#) un numero 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 il 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 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. FMs sono 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

IA generativa

Un sottoinsieme di modelli di [intelligenza artificiale](#) che sono stati addestrati su grandi quantità di dati e che possono utilizzare un semplice messaggio 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 implementati utilizzando le policy di controllo dei servizi e i limiti delle autorizzazioni IAM. 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 CSPM 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 per 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 policy associata a uno o più principi IAM che definisce le relative autorizzazioni all'interno dell'Cloud AWS ambiente.

applicazione inattiva

Un'applicazione che prevede un uso di CPU e memoria medio 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](#).

VPC in ingresso (ingress)

In un'architettura AWS multi-account, un 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.

IIoInternet delle cose industriale (T)

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 [Creazione di una strategia di trasformazione digitale per l'Internet of Things \(IIoT\) industriale](#).

VPC di ispezione

In un'architettura AWS multi-account, un VPC centralizzato 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 informazioni 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. ITIL fornisce le basi per ITSM.

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 gli strumenti ITSM, 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 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. Un LLM può svolgere 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.

BIANCO

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 [Applicazione delle autorizzazioni del privilegio minimo](#) nella documentazione di 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.

MEH

Vedi [sistema di esecuzione della produzione](#).

Message Queuing Telemetry Transport (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 di accelerazione della migrazione (MAP)

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. MAP include 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 AWS con Application Migration Service.

Valutazione del portfolio di migrazione (MPA)

Uno strumento online che fornisce informazioni per la convalida del business case per la migrazione a. Cloud AWS MPA offre una valutazione dettagliata del portfolio (dimensionamento corretto dei server, prezzi, confronto del TCO, 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 strumento MPA](#) (richiede l'accesso) è disponibile gratuitamente per tutti i AWS consulenti e i consulenti dei partner 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 CAF. AWS Per ulteriori informazioni, consulta la [guida di preparazione alla migrazione](#). MRA è 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](#).

MAPPA

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.

QUERCIA

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 sono gli impegni reciproci tra i gruppi IT funzionali, a supporto di un accordo sul livello 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 Well-Architected AWS 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 del cambiamento organizzativo (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. OCM aiuta le organizzazioni a prepararsi e passare a nuovi sistemi e strategie accelerando l'adozione del cambiamento, affrontando i problemi di transizione 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, consultare la [Guida OCM](#).

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). OAC supporta tutti i bucket S3 in generale Regioni AWS, la crittografia lato server con AWS KMS (SSE-KMS) e le richieste dinamiche e dirette al bucket S3. PUT DELETE

identità di accesso origine (OAI)

Nel CloudFront, un'opzione per limitare l'accesso per proteggere i tuoi contenuti Amazon S3. Quando usi OAI, 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 fornisce un controllo degli accessi più granulare e avanzato.

ORR

[Vedi la revisione della prontezza operativa.](#)

NON

Vedi la [tecnologia operativa](#).

VPC in uscita (egress)

In un'architettura AWS multi-account, un 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 policy di gestione IAM collegata ai principali IAM per impostare le autorizzazioni massime che l'utente o il ruolo possono avere. Per ulteriori informazioni, consulta [Limiti delle autorizzazioni](#) nella documentazione di 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. Esempi di informazioni personali includono nomi, indirizzi e informazioni di contatto.

Informazioni che consentono l'identificazione personale degli utenti

Visualizza le [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 \(vedi politica basata sulle risorse\) 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.

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 ruolo IAM o un utente. Per ulteriori informazioni, consulta Principali in [Termini e concetti dei ruoli](#) nella documentazione di IAM.

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 query DNS per un dominio e i 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 prompt [LLM](#) come input per il prompt successivo per generare risposte migliori. Questa tecnica viene utilizzata per suddividere un'attività complessa in sottoattività o per perfezionare 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 [MES](#) basato su microservizi, 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 relazionale SQL.

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

Matrice RACI

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.

Matrice RASCI

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.](#)

Region

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 alle interruzioni o di ripristinarle. [L'elevata disponibilità e il disaster recovery](#) sono considerazioni comuni quando si pianifica la resilienza in Cloud AWS. [Per 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 di assegnazione di responsabilità (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 matrice RASCI e, se la escludi, viene chiamata matrice RACI.

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.](#)

Retrieval Augmented Generation (RAG)

Una tecnologia di [intelligenza artificiale generativa](#) in cui un [LLM](#) fa riferimento a una fonte di dati autorevole esterna alle sue fonti di dati di formazione prima di generare una risposta. Ad esempio, un modello RAG potrebbe eseguire una ricerca semantica nella knowledge base o nei dati personalizzati di un'organizzazione. Per ulteriori informazioni, consulta [Cos'è il 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 espressioni SQL di base e flessibili con regole di accesso definite. RCAC è costituito da autorizzazioni di riga e maschere di colonna.

RPO

Vedi [obiettivo del punto di ripristino](#).

VERSO

Vedi [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

SAML 2.0

Uno standard aperto utilizzato da molti provider di identità (IdPs). Questa funzionalità abilita il single sign-on (SSO) federato, in modo che gli utenti possano accedere Console di gestione AWS o chiamare le operazioni AWS API senza che tu debba creare un utente in IAM per tutti i membri dell'organizzazione. Per ulteriori informazioni sulla federazione basata su SAML 2.0, consulta [Informazioni sulla federazione basata su SAML 2.0](#) nella documentazione di IAM.

SCADA

Vedi [controllo di supervisione e acquisizione dati](#).

SCP

Vedi la [politica di controllo del servizio](#).

Secret

In Gestione dei segreti AWS, 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 sistemi di gestione degli eventi di sicurezza (SEM). Un sistema SIEM 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 alla 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 sicurezza VPC, l'applicazione di patch a un'istanza Amazon EC2 o la rotazione delle credenziali.

Crittografia lato server

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

Policy di controllo dei servizi (SCP)

Una politica che fornisce il controllo centralizzato sulle autorizzazioni per tutti gli account di un'organizzazione in. AWS Organizations SCPs definire barriere o fissare 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

L'URL del punto 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](#).

punto di errore singolo (SPOF)

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

SLAM

Vedi il contratto sul [livello di servizio](#).

SLI

Vedi l'indicatore del [livello di servizio](#).

LENTA

Vedi 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

Vedi [punto di errore singolo](#).

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, consulta [Modernizzazione incrementale dei servizi Web legacy di Microsoft ASP.NET \(ASMX\) mediante container e Gateway Amazon API](#).

sottorete

Un intervallo di indirizzi IP nel VPC. 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

Una tecnica per fornire contesto, istruzioni o linee guida a un [LLM](#) per indirizzarne il comportamento. I prompt di sistema aiutano a impostare il contesto e stabilire regole per le interazioni con gli utenti.

T

tag

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.

Peering VPC

Una connessione tra due VPCs che consente di indirizzare il traffico utilizzando indirizzi IP privati. Per ulteriori informazioni, consulta [Che cos'è il peering VPC?](#) nella documentazione di Amazon VPC.

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 funzione SQL 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.

VERME

Vedi [scrivere una volta, leggere molti](#).

WQF

Vedi [AWS Workload Qualification Framework](#).

scrivi una volta, leggi molte (WORM)

Un modello di storage 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

Fornire a un [LLM](#) le istruzioni per eseguire un'attività ma non esempi (immagini) che possano aiutarla. Il LLM deve utilizzare le sue 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 che prevede un utilizzo CPU e 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.